The Pennsylvania State University
The Graduate School
College of Engineering

MODELING, MONITORING AND SCHEDULING TECHNIQUES FOR NETWORK RECOVERY FROM MASSIVE FAILURES

A Comprehensive Document in
School of Electrical Engineering and Computer Science
by
Diman Zad Tootaghaj

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2017

The comprehensive document of Diman Zad Tootaghaj was reviewed and approved* by the following:

Thomas La Porta
Professor of the School of Electrical Engineering and Computer Science
Dissertation Academic Advisor, Chair of Committee

Ting He
Associate Professor of the School of Electrical Engineering and Computer Science
Dissertation Academic Co-Advisor

Nilanjan Ray Chaudhuri
Assistant Professor of the School of Electrical Engineering and Computer Science
Major Program Member

Marek Flaska
Assistant Professor of the school of Mechanical and Nuclear Engineering
Outside Field Member

Novella Bartolini
Associate Professor at the Computer Science Department of Sapienza University of Rome
Outside Unit Member

*Signatures are on file in the Graduate School.

Abstract

This proposal explores modeling, monitoring and scheduling techniques for network recovery from massive failures, with a focus on optimization methods under uncertain knowledge of failures.

Large-scale failures in communication networks due to natural disasters or malicious attacks can severely affect critical communications and threaten lives of people in the affected area. In 2005, Hurricane Katrina led to outage of over 2.5 million lines in the BellSouth (now AT&T) network. 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. 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 begin by modeling large-scale failures in a communication network. In particular, we propose a new recovery approach under uncertain knowledge of failures. We propose a progressive multi-stage recovery approach that uses the incomplete knowledge of failure to find a feasible recovery schedule. From the elements of this solution, we select a node with highest centrality at each iteration step to repair and exploit as a monitor to increase our knowledge of network state, until all critical services are restored. The recovery problem can be addressed by giving different priority to three performance aspects such as: 1) demand loss, 2) execution time and 3) number of repairs (cost). These aspects are in conflict with each other and we study the trade-off among them.

Next, we focus on the vulnerability due to inter-connectivity of multiple networks.

In particular, we focus on the interaction between a power grid and a communication network. We model the cascading failures in a power gird using a DC power flow model. We tackle the problem of mitigating an ongoing cascade by formulating the minimum cost flow assignment as a linear programming optimization. The optimization aims at finding a minimum cost DC power flow setting that stops the cascading failure, where the total cost is defined as the total weighted amount of reduced power due to the re-distribution of the power in the generators and loads without violating the overload constraint at each line.

Finally, we focus on network monitoring techniques that can be used for diagnosing the performance of individual links or localizing the failures in the network. We study the optimal selection of the monitoring paths to balance identifiability and cost. We considered four closely related optimization problems: (1) *Max-IL-Cost* that maximizes the number of identifiable links under a probing budget, (2) *Max-Rank-Cost* that maximizes the rank of selected paths under a probing budget, (3) *Min-Cost-IL* that minimizes the probing cost while preserving identifiability, and (4) *Min-Cost-Rank* that minimizes the probing cost while preserving rank. We show that while (1) and (3) are hard to solve, (2) and (4) possess desirable properties that allow efficient computation while providing good approximation to (1) and (3). We propose an optimal greedy-based approach for (4) and propose a $(1 - 1/e)$-approximation algorithm for (2). Our experimental analysis reveals that, compared to several greedy approaches, our rank-based optimization performs better in terms of identifiability and probing cost.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to express my deepest gratitude to my advisor Professor Thomas La Porta for the continuous support of my Ph.D study and research and for his inspiration and motivation. He has been a distinguished advisor and a true friend. I never would have made my Ph.D at Penn State without his support.

My sincere thanks also goes to my co-advisor, Prof. Ting He, my mentors and collaborators Prof. Novella Bartolini and Hana Khamfroush who have provided me with valuable help. I am forever indebted to them for promoting a stimulating research environment.

Besides my advisors, I would like to thank the members of my dissertation committee, Dr. Nilanjan Ray Chaudhuri and Dr. Marek Flaska for their time, reviewing my dissertation and comments. In particular, I want to thank Dr. Nilanjan Ray Chaudhuri for his feedback on Chapter 3 of this dissertation. Without his insightful feedback, I was not able to model the inter-dependency between the power grid and the communication network. I would also like to thank Vittorio Rosato for providing the power grid datasets. The code implementation included in Chapter 2 of this proposal was done in collaboration with Stefano Ciavarella and Seamus Hayes and I would like to thank them for their help.

I was extremely lucky to meet and work with Prof. Kathryn McKinley during my graduate studies. Kathryn was my mentor in Microsoft Research during my internship in 2015, and thanks to her enthusiasm, mentoring, support, valuable career and life advice, I have courage and confidence to continue doing research. I have also benefited greatly from collaborations with Dr. Todd Mytkowicz, Dr. Yuxiong He and Dr. Adrian Sampson during my internship at Microsoft.

I am thankful to my lab-mates Noor Felemban, Stefan Achleitner, Nicolas Papernot, Injung Kim and Berkay Celik for their help and friendship.

I also would like to thank the wonderful friends who became my second family in State College. My special thanks goes to Shelembazan: Sanam Sanei and Amirali Kani, Sahar Zarmehri and Farid Farrokhi, Caspian, Azadeh Keivani and Ashkan Balouchi, Nima Zolghadr, Ardalan Mirshani, Ashkan Jasour, Shahrzad Fadaei. I would also like to thank Laleh Sarkarat and Amirreza Aref, Hana Khamfroush and Sam Heshmati, Neda Nasiriani and Alireza Haghighat, Roozbeh Kermani and Farima Fooladi, Amin Jadidi, Narges Shahidi and Amir Meimand, Farnaz Tehranchi and Mehdi Kiani.

I would also like to thank my family for the support they provided me through my entire life and in particular, I must acknowledge my husband and best friend, Farshid Farhat, without whose love, encouragement and editing assistance, I would not have finished this thesis.

In conclusion, I recognize that this research would not have been possible without the financial assistance provided 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 and express my gratitude to them.

Dedication

This thesis is dedicated to my parents, and my husband, Farshid, for their endless love, support and encouragement.

Chapter 1

Introduction

Needless to say, large-scale failures due to natural disasters or malicious attacks can severely affect operation of critical infrastructures and cause catastrophic economic and social disruptions. Communication networks and power grids are examples of such critical infrastructures that are highly vulnerable to such failures. In 2005, Hurricane Katrina led to outage of over 2.5 million lines in the BellSouth (now AT&T) network [1]. In 2003, a large cascading blackout, in northeast of the United States, led to over 50 million people losing power, some for several days. The overall cascade propagation lasted approximately four hours, during which a cascade prevention mechanism could have stopped further propagation of the failure and lowered cost of recovery.

The leading causes of these failures has been reported to be inadequate training, planning and operations studies to respond to the emergency situations [2,3], which highlights the necessity for a holistic control and recovery approach that has the ability to use the real-time data taken from a monitoring network to predict and prevent possible failures. Furthermore, it is crucial to have a strategic recovery plan that effectively utilizes the available resources and maximizes the total operation of the disrupted services during the recovery time.

In this proposal, we first study large-scale failures in (i) communication networks in Chapter 2 and (ii) an interdependent power grid and it's monitoring network in Chapter 3. We then focus on network monitoring techniques that can be used for diagnosing individual links' performance or localizing the failures. In Chapter 4,

we study the optimal selection of the monitoring paths to balance identifiability and cost.

1.1 Motivation and Challenges

Despite considerable research in the past few decades leading to multi-fold improvements on large-scale failure detection and mitigation approaches, the problem has become more interesting and challenging for three main reasons: (1) Lack of complete knowledge, (2) Interdependency between multiple networks, and (3) Progressive Recovery. In the following subsections, we summarize the three reasons.

1.1.1 Lack of Complete Knowledge

Almost all failure recovery and prevention algorithms assume complete information about the failures and ignore the uncertainty caused by failure of the dependent monitoring systems. In a real system, one needs to assume that only incomplete data may arrive at control centers due to failures in the underlying communication and monitoring network. Network recovery and failure prevention is a challenging problem under uncertainty of the exact location of the disrupted network components.

To clarify the discussion, consider Deltacom topology taken from the Internet Topology Zoo shown in Figure 1.1 [4, 5]. After a large-scale failure occurs in the network, the state of the entire network is not visible to the network manager or control centers. Instead, the network manager knows that some nodes and links have failed, some continue to work and the fate of others is uncertain. The working nodes and links are shown with green color in Figure 1.1, the broken nodes and links are shown in red and the uncertain nodes and links are shown in grey. The main challenge is that the status of grey nodes and links is unknown to the network manager and therefore, current recovery techniques that assume complete and accurate information is accessible, might not work as they should. To this end, we propose an itrative stochastic recovery approach (ISR) in Chapter 2, that runs a multi-stage stochastic optimization algorithm. At each iteration step, ISR finds

Figure 1.1: Large-scale failure in Deltacom topology.

a feasible solution set and selects a candidate node to repair and exploits it as a monitor to discover the surrounding network. The procedure is repeated until all critical services are restored.

1.1.2 Interdependency between Multiple Networks

Most of the research on large-scale failure management has concentrated on the recovery of a single network [6–8]. Many man-made or natural systems can be modeled as an interconnection of multiple networks, where the nodes are the system components and the edges show the interaction or dependency between different components. Because of the dependency between different components in multiple networks, perturbations caused by physical attacks or natural disasters in one node can cascade and affect other nodes in the system. The cascaded failure can repeat multiple times, feeding on itself and accelerating, eventually resulting in a total failure of the whole system.

Most of the work on the recovery and cascade prevention algorithms, concentrate on one network, while today's critical infrastructures are highly interconnected and mutually dependent on each other. For example, the operation and reliability of the electric power networks relies on the operation of the control communication network that provides the required information about the power lines being overloaded. In addition to the power grids and communication networks, other critical infrastructures are also coupled together, such as food supply and water systems, financial transactions and power grids, transportation systems and food

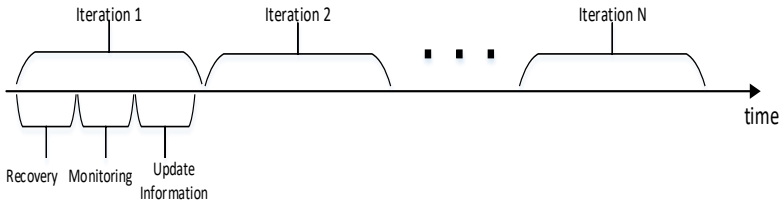


Figure 1.2: Different steps of our progressive recovery approach.

supply. Today, critical infrastructures are becoming increasingly correlated and interdependent. Therefore, modeling and understanding the interactions between multiple networks and designing failure resilient infrastructures is crucial for the reliability and availability of many applications and services.

1.1.3 Progressive Recovery

Restoring critical services after a large-scale disruption or a cascaded failure is not a one shot operation. The amount of repair resources required to restore damaged network elements may vary over time. Also, each network element might require a different amount of resources to be restored. Therefore, finding the optimal assignment of resources to maximize the recovery over time is a challenging problem. To the best of our knowledge, our progressive recovery approach is the first work that studies progressive recovery of a disrupted network under uncertainty. Figure 1.2 shows different steps of our recovery approach. At each iteration step, based on the available resources, 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.

1.2 Our Contributions

In this proposal, we aim to provide comprehensive solutions for accurately modeling, monitoring and scheduling the recovery of the network from large-scale failures under uncertain knowledge of failures. Specifically, we focus on four main goals: (1)

Minimizing the number of repaired elements, (2) Minimizing the amount of demand loss, (3) Minimizing the execution time and (4) Minimizing the cost of monitoring probes. We briefly explain these goals in the following three subsections.

1.2.1 Network Recovery from massive Failures under Uncertain Knowledge of damages

In Chapter 2, 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. The proposed iterative stochastic recovery (ISR) approach recovers the network in a progressive manner while satisfying the critical service demands [9]. At each iteration step, ISR makes a decision to repair a part of the network and gathers more information by putting a monitor on the selected node. We propose several algorithms to find a feasible solution set at each iteration of the algorithm. Our results show that ISR outperforms the state-of-the-art ISP algorithm while we can configure our choice of trade-off between the execution time, number of repairs and demand loss.

1.2.2 Controlling Cascading Failures in Interdependent Networks under Incomplete Knowledge

In Chapter 3, we show that the inter-connectivity and dependency between different elements makes complex networks more vulnerable to failure [10]. We study the inter-dependency between a power grid and a communication network. We propose a failure mitigation strategy that first detects the failure and limits further propagation of the disruption by re-distributing the generator and load's power. We then formulate a recovery plan to maximize the total amount of power delivered to the demand loads during the recovery intervention. The cascade mitigation problem is formulated as a linear programming optimization that minimizes the

cost of new flow assignment (*Min-CFA*) and aims at finding a DC power flow setting that stops the cascading failure at minimum cost. At the recovery phase, we aim at maximizing the restored accumulative flow. We show that the recovery problem (*Max-R*) is NP-Hard and propose heuristic recovery strategies that work under partial knowledge of damage locations. We propose a consistent failure set algorithm (*CFS*) to locate the failures.

1.2.3 Optimizing Cost-Identifiability Trade-off for Probing-based Network Monitoring

In Chapter 4, we study the optimal selection of monitoring paths to balance identifiablity and cost [11]. To this end, we considered four closely related optimization problems: (1) *Max-IL-Cost* that maximizes the number of identifiable links under a probing budget, (2) *Max-Rank-Cost* that maximizes the rank of selected paths under a probing budget, (3) *Min-Cost-IL* that minimizes the probing cost while preserving identifiability, and (4) *Min-Cost-Rank* that minimizes the probing cost while preserving rank. We show that while (1) and (3) are hard to solve, (2) and (4) posses desirable properties that allow efficient computation while providing good approximation to (1) and (3). We proposed an optimal greedy-based approach for (4) and proposed a $(1 - 1/e)$-approximation algorithm for (2). Our experimental analysis reveals that, compared to several greedy approaches, our rank-based optimization performs better in terms of identifiability and probing cost.

1.3 Organization

The remainder of the proposal is organized as follows. Chapter 2 presents our progressive recovery approach that iteratively restores critical services and monitors the network to detect the failures [9]. Chapter 3 studies the inter-dependency between a power grid and a communication network [10]. We propose a failure mitigation strategy that first detects the failures and limits further propagation of the disruption by re-distributing the generator and load's power. We then formulate

a recovery plan to maximize the total amount of power delivered to the demand loads during the recovery intervention. In Chapter 4, we study the optimal selection of monitoring paths to balance identifiability and cost [11]. Finally, we conclude the dissertation proposal and discuss the future work in Chapter 5.

Chapter 2

Network Recovery from Massive Failures under Uncertain Knowledge of Damages

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. The aggregate monetary cost of Hurricane Katrina was 108 billion dollars [12]. The storm led to the loss of service over 2.5 million lines on 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., [6, 8]. 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.

In this Chapter, 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.

2.1 Introduction

Prior works on network recovery assume having complete information and ignore the uncertainty caused by lack of information about the exact location of failures. Network recovery is a challenging problem under uncertainty of the exact location of the disrupted network components.

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 our proposed recovery algorithms, we modified the state-of-the-art *iterative split and prune* (ISP) algorithm [6], presented as *progressive ISP* to work 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.

2.2 Background and Motivation

2.2.1 Background

Large-scale network failure detection and recovery has been studied when full knowledge of the failure pattern is available in the system [7, 13–15]. 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 [8]. While both Wang et al.'s work and our work aim to design a progressive recovery approach, the objective is different. In [8], 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. [6]. In addition, both [8] and [6], assume that full knowledge of failure is available in the system while our work does not make this assumption. Further, Ciavarella et al. [16] propose a polynomial time heuristic, called Centrality based Damage Assessment and Recovery (CeDAR) that progressively recovers and monitors the damaged network elements. Our formulation differs from CeDAR in that we try to minimize the number of repaired elements while CeDAR maximizes the accumulative flow over time. Also, we assume availability of a probabilistic estimate of the failure scenario, while CeDAR does not have 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 [6]. 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 2.5 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.

2.2.2 Motivation

In this section, we show the gap between optimal recovery and ISP [6] 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, meaning 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 network topologies which are more fully described in Section 4.5 to illustrate the gap in performance between ISP run 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 the whole network is gray,

Table 2.1: 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

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 2.1 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 more fully described in Section 2.5, to run in iterations, where in each iteration a repair is made. Information about network state is then gathered about 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.

2.3 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. 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 [17], seismography analysis in case of an earthquake [18], or stability analysis of different parts of the network. The network elements in the gray area might have geographical or independent correlation of failures [19]. For example, if we are certain that a router in a specific building is failed, 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 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 $G = (V, E)$, where V represents the set of nodes and E is the set of links, 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 $G = (V, E)$ 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.

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:

$$\underset{\delta_i^v, \delta_{i,j}^e}{\text{minimize}} E_\zeta \sum_{(i,j)\in E_U \cup E_B} k_{ij}^e(\zeta_{ij}^e(n))\zeta_{ij}^e(n)\delta_{ij}^e + \sum_{i\in V_U \cup V_B} k_i^v(\zeta_i^v(n))\zeta_i^v(n)\delta_i^v$$

$$\text{subject to} \quad c_{ij}.\delta_{ij}^v \geqslant \sum_{h=1}^{|E_H|} f_{ij}^h(n) + f_{ji}^h(n) \qquad \forall (i,j) \in E \tag{2.1a}$$

$$\delta_i^v.\eta_{max} \geqslant \sum\nolimits_{(i,j)\in E_B} \delta_{ij}^e \qquad \forall i \in V \tag{2.1b}$$

$$\sum\nolimits_{j\in V} f_{ij}^h(n) = \sum\nolimits_{k\in V} f_{ki}^h(n) + b_i^h(n) \qquad \forall (i,h) \in V \times E_H \tag{2.1c}$$

$$f_{ij}^h(n) \geqslant 0 \qquad \forall (i,j) \in E, h \in E_H \tag{2.1d}$$

$$\delta_i^v, \delta_{i,j}^e \in \{0,1\} \tag{2.1e}$$

where the binary variables δ_{ij}^e and δ_i^v represent the decision to repair link $(i,j) \in E$ and node $i \in V$, c_{ij} is the capacity of edge (i,j), $f_{ij}^h(n)$ is the fraction of flow h that will be routed through link (i,j), η_{max} is the maximum degree of the network, and $b_i^h(n)$ 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; $k_{ij}^e(\zeta_{ij}^e(n))$ and $k_i^v(\zeta_i^v(n))$ 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 5.1a 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; 5.1b specifies that the degree of each node is smaller or equal than the maximum degree of the network; 5.1c 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; 5.1d states that we consider non-negative

assignment of flows and finally 5.1e 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 [20–22], is a special case of our optimization problem. The deterministic version of our problem is shown to be NP-Hard in [6]. 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.

2.4 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 2.1 and Algorithm 1.

Initially, ISR starts by estimating the probability distribution 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, $S_t = \{(i \in$

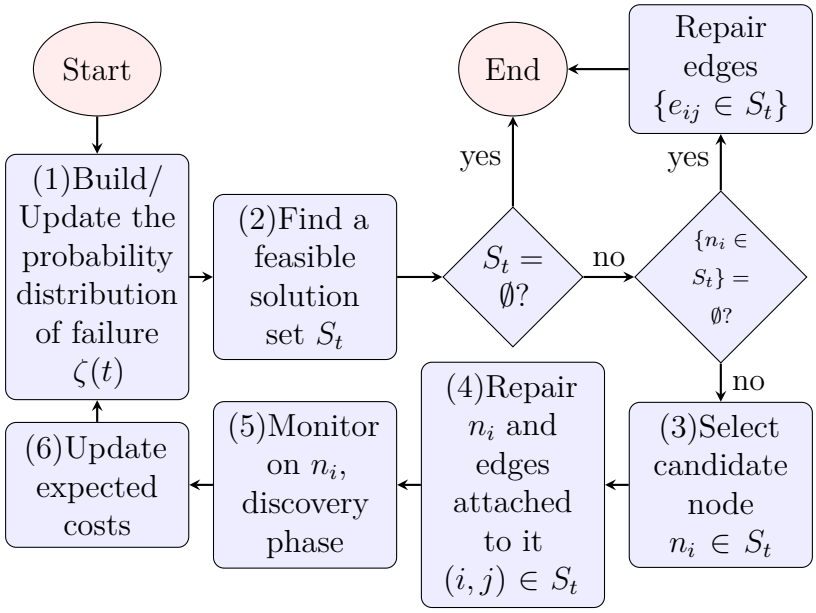


Figure 2.1: Different steps of our iterative stochastic recovery (ISR).

$V_U \cup V_B \quad |\delta_i = 1), ((i,j) \in E_U \cup E_B \quad |\delta_{ij} = 1)\}$ (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 2.4.1 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.

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 m-hop discovery phase;
        t = t + 1 ;
        Update our belief ζ(t) from failure probability distribution from the discovered nodes/edges ;
return Solution
```

2.4.1 An approximate feasible solution

This section describes our three different approaches to find an approximate feasible solution, step (2) in Figure 2.1, 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 with respect to the iterative branch and bound solution.

2.4.1.1 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.

2.4.1.2 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 [23]. 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 [24]. 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.

2.4.1.3 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 of the algorithm, we repair a node with maximum flow. If there exist more than one node, which has the same amount of flow in the current solution, we select the one with maximum failure probability.

2.4.2 Best candidate node selection

The choice of the best candidate node, step (3) in Figure 2.1, 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:

$$N^*_i = argmax_{n_i \in S_t} \frac{\sum_{p \in P^*_{n_i}} f(p)}{\sum_{p \in P^*} f(p)} \tag{2.2}$$

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 used maximum failure probability to select the candidate node. 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 chose centrality as the main metric and 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.

2.4.3 Monitoring nodes

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.

2.4.4 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 1 *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.*

2.5 Progressive ISP

This section describes progressive ISP which is our extension of the state-of-the-art iterative split and prune (ISP) [6]. 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 2.1. 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 2.4.1.1 (ISR-SRT), and c_{ij} is capacity of e_{ij}. The edge cost is

Table 2.2: 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

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 4.5, 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 chapter we use the terms "progressive ISP" and "ISP" interchangeably.

2.6 Evaluation

In this section, we compare ISR algorithms, presented in 4.4, to the modified version of ISP introduced in Sections 2.5. We use different network topologies including planar and non-planar real topologies taken from the Internet Topology Zoo [4, 5]. Table 4.3 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 [25].

2.6.1 m-hop discovery impact

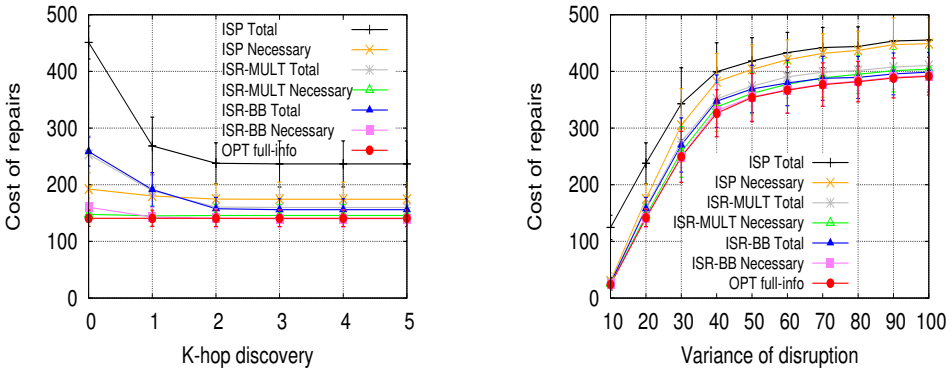
In this section, we investigate the impact of the depth of discovery phase on the performance of the proposed algorithm. We changed the number of discovered hops for the monitoring nodes from 1 to 5. We used the BellCanada topology from the Internet Topology Zoo [4, 5], with 10 demand pairs and 4 units 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 2.2a, 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 chapter, due to space limitations. These experiments highlighted that the impact of the parameter m varies significantly with the size of network topology.

2.6.2 Disruption variation

In this scenario, addressed in Figure 2.2b, 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 2.2b 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



(a) Increasing m-hop. (b) Increasing disruption, (m-hop=2).

Figure 2.2: m-hop discovery and disruption variance.

disruption variations and the discovery phase does not help to reduce the number of unnecessary repairs by a large amount.

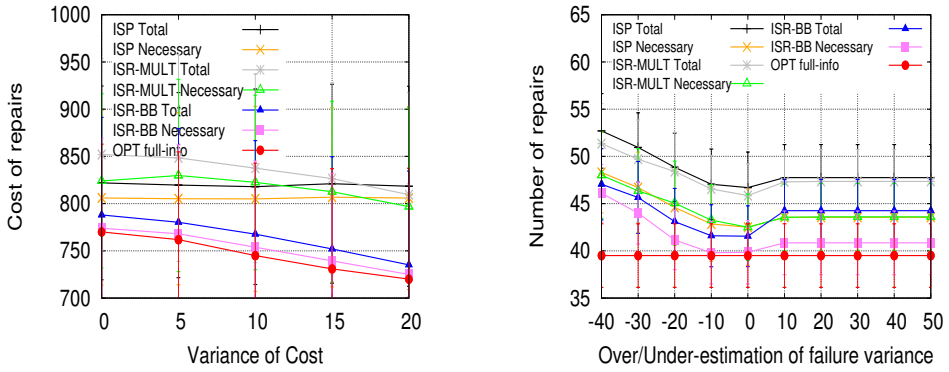
2.6.3 Heterogeneous repair cost

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 2.3a 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 ISP algorithm when the variance of heterogeneity is higher. Therefore, in the next set of experiments we consider homogenous repair cost.

2.6.4 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



(a) Heterogeneous repair cost.

(b) Sensitivity analysis.

Figure 2.3: 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).

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 2.3b 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 ISP.

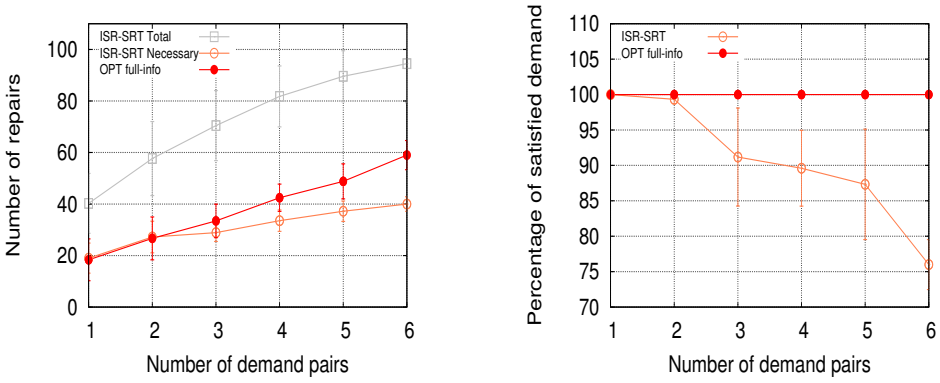
2.6.5 Trade-off on demand loss, time complexity, and number of repairs

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 2.4, we considered the Deltacom topology taken from the Internet Topology Zoo [4,5], 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 2.4a 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 2.4b 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



(a) repairs. (b) demand loss.

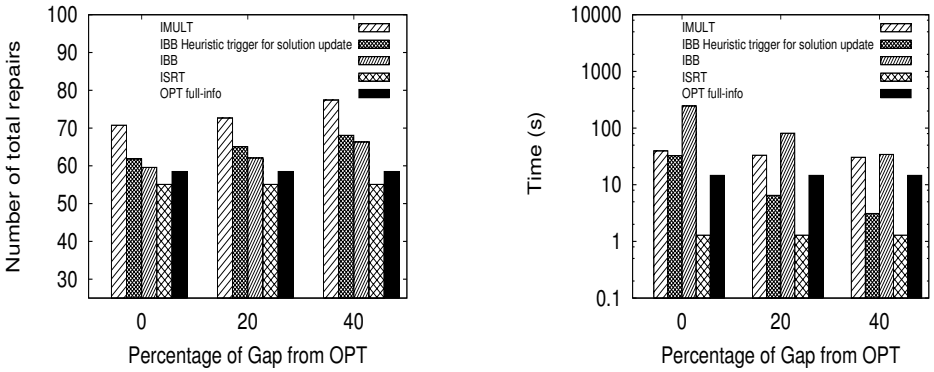
Figure 2.4: Trade-off between number of repairs and demand loss.

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 2.5 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.

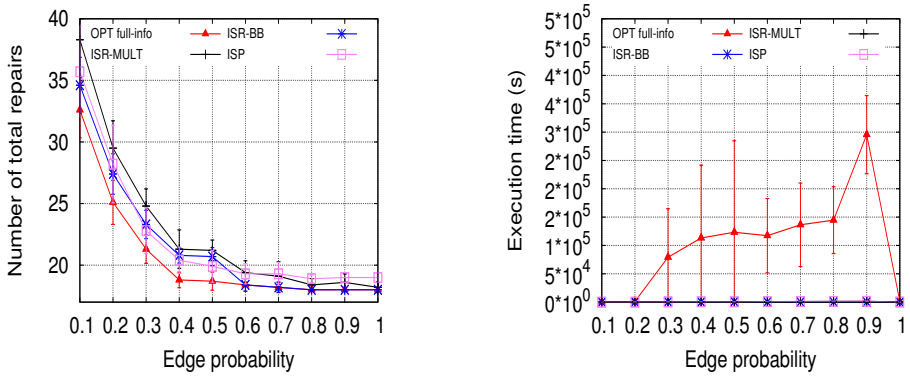
Figure 2.5a shows that, when we increase the gap from 0 to 40%, the difference between the total number of repairs of ISR-MULT with respect to optimal increases by a factor of 1.6, while this factor is 3.4 for ISR-BB. Furthermore, we observe that by running the Heuristic trigger for solution update, introduced in Section 2.4.4, 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 1. 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



(a) repairs. (b) time.

Figure 2.5: Trade-off between execution time and repairs.



(a) repairs. (b) time.

Figure 2.6: Synthetic Erdos-Renyi topology with 100 nodes.

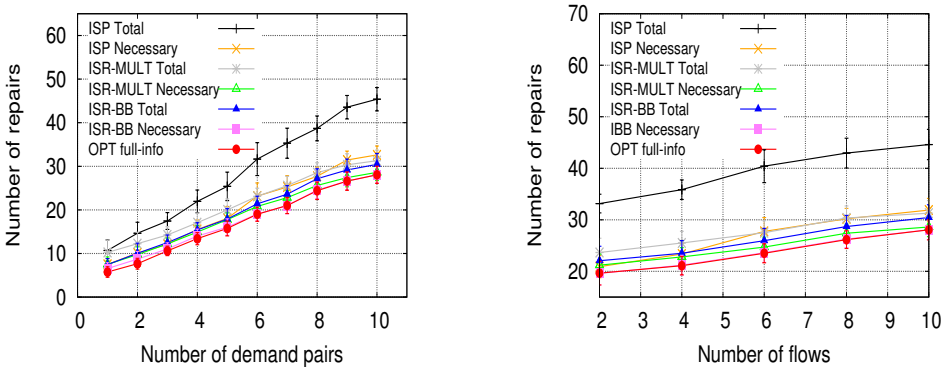
Erdos-Renyi graph, any two nodes are connected through an edge with probability p. We considered an Erdos-Renyi topology with 100 nodes where each link has a capacity of 1,000 units of flow. We set the number of critical demand pairs to 6, of one unit each. Notice that with this setting of demand flows and capacities, the problem reduces to establishing connectivity between the endpoints of the demand pairs. The complexity of the problem increases as 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 till all critical services are restored. Figure 2.6b 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.

2.6.6 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 2.7a 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 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 demand pairs to 5 and increased the units of flow per demand pair from 2 to 10. Figure 2.7b 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, ISP performs slightly better than the ISR-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 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



(a) Increasing demand.

(b) Increasing flows.

Figure 2.7: Increasing demand pairs and flows (BellCanada).

Table 2.3: 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.

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 2.3 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.

2.7 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 chapter considers, 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.

We aim to extend this work by comparing our stochastic optimization with Centrality based Damage Assessment and Recovery (CeDAR) [26]. Our recovery approach minimizes the total number of repairs while CeDAR aims at optimizing the accumulative flow over time under recovery resource constraints.

Chapter 3

Controlling Cascading Failures in Interdependent Networks under Incomplete Knowledge.

Vulnerability due to inter-connectivity of multiple networks has been observed in many complex networks. Previous works mainly focused on robust network design and on recovery strategies after sporadic or massive failures in the case of complete knowledge of failure location. We focus on cascading failures involving the power grid and its communication network with consequent imprecision in damage assessment. We tackle the problem of mitigating the ongoing cascading failure and providing a recovery strategy. We propose a failure mitigation strategy in two steps: 1) Once a cascading failure is detected, we limit further propagation by re-distributing the generator and load's power. 2) We formulate a recovery plan to maximize the total amount of power delivered to the demand loads during the recovery intervention. Our approach to cope with insufficient knowledge of damage locations is based on the use of a new algorithm to determine consistent failure sets (*CFS*). We show that, given knowledge of the system state before the disruption, the *CFS* algorithm can find all consistent sets of unknown failures in polynomial time provided that, each connected component of the disrupted graph has at least one line whose failure status is known to the controller [10].

3.1 Introduction

Power grids are one of the most critical infrastructure in our everyday lives. Large-scale blackouts in the power grid due to propagating failures, natural disasters or malicious attacks, can severely affect the operation of other interconnected critical infrastructures and cause catastrophic economic and social disruptions.

In September 2003, a large cascading blackout, in Italy, led to the shortage of 6400 MW of power, which caused a complete system collapse. The cascade began when a tree flashover caused a 380-kV line to fail between Italy and Switzerland [2]. The cascade lasted approximately several minutes, a time sufficient for enabling countermeasures, which could have mitigated and limited the blackout propagation. The main cause of most cascading failures including 2003 Italian and Northeast US-Canada blackout is reported to be inadequate training, planning and operations studies to respond to the emergency [2, 3]. This highlights the necessity of a holistic power control strategy that utilizes real-time monitoring to detect, predict and prevent possible failures. Furthermore, it is crucial to have a strategic recovery plan that ensures effective use of the available resources during the recovery process.

The functionality of the electric power grid and its damage assessment rely on the operation of a monitoring system. Such a monitoring system utilizes communication lines to interact with power grid controllers, to notify them of detected damage involving overloaded power lines. When a cascading failure affects the power grid, the monitoring system and the communication network are also likely to fail, inevitably compromising the completeness and reliability of damage detection and assessment.

Previous works addressed the problem of cascading failures involving the power grid and the communication network. The majority of these works aimed at characterizing the residual functionality of the networks subject to failure, on the basis of network topology, size and location of the initial damage which caused the cascading phenomenon. Recovery was mostly considered only in the unrealistic case of complete knowledge of the damage, and with interventions aimed at restoring network functionality under the assumption that failure propagation has ended.

In this chapter, we address, for the first time, the study of mitigating an ongoing cascade of failures in a power grid and maximizing the provided energy by recovering

damaged network elements while the cascade is still in progress and knowledge of the network damages is only partial. Uncertainty of the exact location of the disrupted network components poses a new challenge that has never been successfully tackled.

We study the impact of cascading failures in power grids and propose a mitigation strategy in two phases that (1) stops the cascade when the system is still in transient state, and (2) provides a recovery schedule that maximizes the total amount of power delivered to demand loads over all the steps of the recovery process.
In the following, we summarize the most important **contributions**:

- We tackle the problem of mitigating an ongoing cascade (first phase) by formulating the minimum cost flow assignment (*Min-CFA*) problem as a linear programming optimization. *Min-CFA* aims at finding a DC power flow setting that stops the cascading failure at minimum cost. We define the total cost, the total weighted amount of reduced power due to the re-distribution of the power in the generators and loads without violating the overload constraint at each line.

- We study the interaction between two correlated networks: i) a power grid, and ii) a communication network. We show that, the inter-connectivity and dependency between the two networks makes the power grid more vulnerable to cascading failures because damage to the communication network may result in incomplete information. In the absence of complete knowledge of failure locations, classic cascade prevention approaches may not work as they should.

- We address the recovery phase (second phase) formulating the problem of maximizing the restored accumulative flow (*Max-R*). We show that *Max-R* is NP-hard and propose a heuristic recovery strategy which works under partial knowledge of damage locations by calculating consistent failure sets to locate failures.

- We performed an experimental evaluation, considering cascading failures in a power grid and its monitoring communication network. We use real data from the Italian high-voltage transmission grid (HVIET) and its communication network (GARR) [27–29]. The experiments show that when 60% of the power grid is disrupted, our cascade prevention approach (Min-CFA) finds

the optimal solution with 54.39% of the demand satisfied. While, without a cascade prevention algorithm, the whole system fails. Furthermore, our backward recovery approach on average delivers 20% more power to the loads with respect to a shadow-pricing approach inspired by the work in [16].

While our recovery approach is proposed for a case study of a power grid and a communication network, our approach invites further work on recovery of other interdependent networks.

The remainder of this chapter is organized as follows. Section 3.2 discusses the background and motivation behind this work. In section 3.3, we explain the *Min-CFA* and *Max-R* optimization problems and show that *Max-R* is NP-Hard. Section 3.4 describes our algorithms. Section 3.5 shows our evaluation methodology and experimental results and Section 3.6 concludes the chapter with a summary.

3.2 Background and Motivation

Most of the research on large-scale failure management has concentrated on the recovery of a single network. Bartolini et al. [6], Al Sabeh et al. [7], Tootaghaj et al. [9] and Wang et al. [8] jointly address the progressive recovery of a single data communication network after a large-scale disruption.

In complex networks however, multiple heterogeneous networks may be interconnected and interdependent. Because of the interdependency between different components, perturbations caused by failures, physical attacks or natural disasters may propagate across the different networks. To study the interactions in a complex network, graph-based models are typically used, where nodes are the system components and edges model the interactions or dependencies between different components of the same or of different networks. A cascading failure may propagate across the nodes of the complex network traversing the dependency edges across a same network or multiple networks, possibly accelerating and eventually resulting in a potentially total failure of the system.

Cascading failures in interdependent networks have been studied in several works [30–35]. The existing works on interdependent networks can be broadly classified into three categories: 1) those which study the interaction through perco-

lation theory [33–36], 2) works which try to identify most vulnerable nodes and design failure resilient networks [30, 37–40], 3) and the works which try to find the root cause of failures [41, 42]. To the best of our knowledge, the problem of mitigating and recovering from cascading failures, during the transient regime of the propagation process, has not been studied extensively. Percolation/epidemic-based approaches depend on having a prior knowledge about the probabilistic model of failure propagation, which is hard to obtain. In addition, real systems usually have a deterministic failure propagation. For example, if a power line fails, a certain number of communication routers will stop working. Finding the root cause of the propagating failure is shown to be NP-Hard [41], but is the key to design restoration algorithms. Identifying the most vulnerable nodes and root cause of failures helps to design failure-resilient systems but does not provide a mitigation solution when the failure happens in the system.

Cascading failures in power grids can be due to a permanent short circuit, e.g. a tree falls on a transmission line etc., or due to a to a temporary failure, e.g. a temporary short circuit in a transmission line. When a short circuit happens in one of the transmission lines, the controller sends a "trip" signal to the breakers and the breakers set open. The controller tries to connect the breaker multiple times before the line fails. In case of a permanent failure, the breaker stays open circuit. After a line fails in the system, the power re-distributes according to Kirchhoff's and Ohm's laws. This can cause other lines to be overloaded and trigger new failures. The cascaded failures can trigger multiple times and spread over the entire network. Unlike the approach proposed in [37], that re-distributes the power flow evenly over all transmission lines, we use the DC power flow model [43, 44] which is widely used in studies of cascading failures.

The operation and reliability of today's power grid is highly dependent on the operation of the communication network that provides the necessary information needed by the supervisory control and data acquisition (SCADA) system to respond to emergency situations. The required data is measured and gathered at the substations from the intelligent electronic devices (IEDs), control circuit breakers and phasor measurement units (PMUs) [45, 46]. While the security of the control system is itself an important challenge on the reliability of the power grids (e.g. a compromised controller can send a trip signal to disrupt the power grid) [47, 48], we focus on the inter-dependency between the operation of the monitoring system

and the controller to avoid the cascaded failure.

3.3 Problem Definition

We consider a complex system for which some failures are detected while the propagation is still in the transient regime. We propose a mitigation strategy to avoid further cascade and a recovery plan to maximize the total operability of the network during K steps of recovery. We define the power grid operability to be the accumulative amount of power delivered to satisfy the demand load over the K recovery steps. Our approach can be extended to the use of other operability measures such as the total number of working power lines etc. Table 4.2 shows the notation used in this chapter.

The power and communication networks are modeled as undirected graphs $G_p = (V_p, E_p)$ and $G_c = (V_c, E_c)$ respectively. Transmission lines in the power grid are monitored by several sensors deployed nearby that area. The aggregated data are then sent to closest communication node and to the control center. Also the control commands are sent to the closest communication node. Therefore, each power line is monitored and controlled through the closest communication node. Each node $i \in V_p$ in the power grid can be 1) a generator G_i, where the power is inserted, 2) a load L_i, where the power is extracted, or 3) a junction J_i where power flows by. As transformer and generator failures are extremely unlikely, we hereby assume that failures only occur in power lines (E_p). Further, we consider the inter-dependency between the power grid and the communication network such that failures in the communication network would lead to lack of information in the control center. We assume that the communication network gets power from an emergency source in case of failures in the power grid and ignore the ping-pong failures between the two networks.

The edges in the power grid graph G_p may be in three different states:

1. the set $E_p^{B,t} \subseteq E_p$ is the set of certain **broken** edges (hereby denoted as red edges) at time t [1].

[1]Notice that in order to be able to assess an edge damage, the edge must be connected to a working communication node in G_c. The working communication node provides the failure status of the edge to the central controller and can send power adjustment commands to the connected loads or generators.

Table 3.1: Summary of notations.

Notation	Explanation
$G_p = (V_p, E_p)$	undirected graph modeling the power grid. V_p is the set of nodes and E_p is the set of links
$G_c = (V_p, E_c)$	undirected graph modeling the communication network. V_c is the set of nodes and E_c is the set of links
$G_i \in V_p$	generator node $G_i \in V_p$ where the power is inserted
$L_i \in V_p$	load node $L_i \in V_p$ where the power is extracted
$J_i \in V_p$	junction node $J_i \in V_p$ where the power just flows by
$E_p^{B,t} \subseteq E_p$	set of broken edges in the red area
$E_p^{U,t} \subseteq E_p$	set of edges in the grey area whose failure patterns is unknown
$E_p^{W,t} \subseteq E_p$	set of edges in the green area which are known to be working correctly
F_{ij}^t	power flow in line (ij) at time t
θ_i^t	voltage angle of node i at time t
x_{ij}	series reactance of line (ij)
P_i^t	power generated/consumed at node i at time t
B^t	nodal admittance matrix at time t
w_{G_i}	weighted cost of increasing the power in generator G_i
w_{L_i}	weighted cost of shedding the power of load L_i
$P_{G_i}^{max}$	maximum power that can be generated in G_i
$P_{L_j}^{demand}$	demand load at L_j
F_{ij}^{max}	maximum capacity of the line (ij)
E_k^R	set of restored edges at iteration k
$\delta_{(ij),k}$	decision variable to repair $(ij) \in E_p^{B,t}$ at the kth iteration
r_{ij}	resources needed for repairing (ij)
R_k	available resource at iteration k of the recovery

2. the set $E_p^{U,t} \subseteq E_p$ is the set of edges of **unknown** working status (denoted as grey edges) at time t,

3. the set $E_p^{W,t} \subseteq E_p$ is the set of certain **working** edges (denoted as green edges) at time t.

3.3.1 2-phase Recovery approach: Power grid case study

In this section, we study the mitigation of cascading failure and related recovery process in a power grid. Figure 3.1 illustrates the two phases of this process: 1)

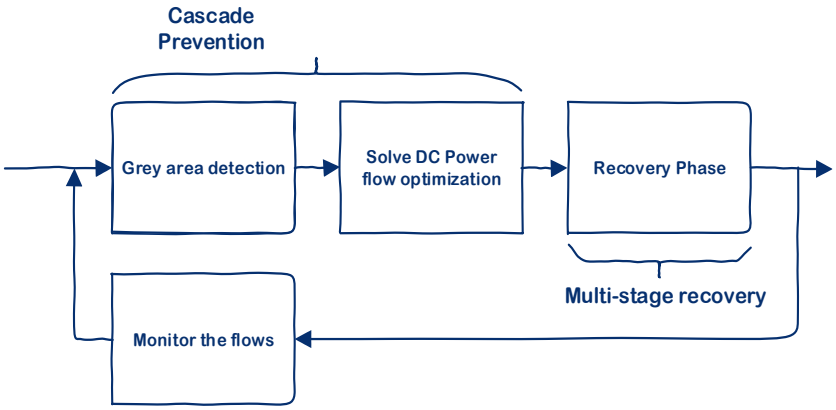


Figure 3.1: Recovery Process: 1) Re-distribution of power, 2) Recovery phase.

mitigation of the cascade using a combination of load shedding and adjustment of the generated power, and 2) recovery phase.

3.3.1.1 Cascade mitigation (Min-CFA)

We model the cascading failure in a power system using a DC load flow model [43]. The DC power flow model provides a linear relationship between the active power flowing through the lines and the power generated/consumed in the nodes, which can be formulated as follows:

$$F_{ij}^t = \frac{\theta_i^t - \theta_j^t}{x_{ij}} \ , \tag{3.1}$$

where, F_{ij}^t is the power flow in line (ij) at time t, x_{ij} is the series reactance of line (ij) and θ_i^t and θ_j^t are the voltage angles of node i and j at time t. The power flow of node i can be found by summing up the power flows of all its adjacent power lines:

$$P_i^t = \sum_j F_{ij}^t \tag{3.2}$$

We can re-write the power flow model as a linear system of equations as follows:

$$P^t = B^t \theta^t \tag{3.3}$$

where B^t is nodal admittance matrix at time t, $b_{ij}^t = -\frac{1}{x_{ij}^t}$ for $i \neq j$ and $b_{ii}^t = \sum_k \frac{1}{x_{ik}^t}$. Once a transmission line trips, the power is redistributed according to Equation (3.3) and if the power exceeds the maximum threshold on another line (ij), the transmission line (ij) will also disconnect unless we reduce the total load or re-distribute the generated power.

Theorem 1 *The power flow model (Equation 3.3) is always solvable for each connected component of the power graph.*

Proof In Section 1 (APPENDIX).

Once we detect an outage of the transmission line, we readjust power and load according to the optimization problem described in the following. The Minimum Cost Flow Assignment (*Min-CFA*) optimization problem minimizes the total cost of reducing the load or generator's power. Let w_{G_i} be the weighted cost of reducing the power in generator G_i and w_{L_i} be the weighted cost of decreasing the power of load L_i. The *Min-CFA* problem to avoid the cascaded failures can be formulated as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{G_i, L_j \in V_p} w_{G_i}(P_{G_i}^0 - P_{G_i}^t) - w_{L_j}(P_{L_j}^t - P_{L_j}^0) \\
\text{subject to} \quad & 0 \leqslant P_{G_i}^t \leqslant P_{G_i}^0, \ \ \forall G_i \in V_p^t \\
& 0 \leqslant P_{L_j}^t \leqslant P_{L_j}^{demand}, \ \ \forall L_j \in V_p^t \\
& -F_{ij}^{max} \leqslant F_{ij}^t \leqslant F_{ij}^{max}, \ \ \forall (ij) \in E_p^t \\
& \sum_{G_i, L_j \in V_p} P_{G_i}^t + P_{L_j}^t = 0. \\
& P_{G_i}^t = \sum_j F_{ij}^t, \ \ \forall G_i \in V_p^t, (ij) \in E_p^t \\
& P_{L_i}^t = \sum_j F_{ij}^t, \ \ \forall L_i \in V_p^t, (ij) \in E_p^t \\
& P_{G_i}^t = B^t \theta^t, \ \ \forall G_i \in V_p^t \\
& P_{L_j}^t = B^t \theta^t, \ \ \forall L_j \in V_p^t \\
& F_{ij}^t = \frac{(\theta_i^t - \theta_j^t)}{x_{ij}}, \ \ \forall (ij) \in E_p^t
\end{aligned}
\tag{3.4}
$$

The first constraint indicates that the power generated at each generator cannot exceed the initial power at each generator. If we had full knowledge about the

location of failures, we could have a more relaxed constraint to increase the power at some of the generators without violating a maximum threshold. However, under uncertain failure we reduce our solution space to decrease the possibility of consequent cascades due to unknown knowledge. The second constraint shows that the reduced load cannot exceed the demand. The third constraint shows that the power flowing through each line cannot exceed the maximum capacity of the line. The fourth constraint is the power conservation condition, i.e. the total power generated in the generators should be equal to the total power consumed in the loads. The fifth and sixth constraints show that the total power generated/consumed at each node should be equal to the total power flow through its edges. The last three constraints reflect the DC power flow model.

We next provide the sufficient condition to ensure the solution to our power re-distribution model under uncertain knowledge of the failure does not overload more lines in the system with respect to the case where we don't run the *Min-CFA* optimization. Suppose that the new power distribution for each node i, is reduced by a factor α_i, where $0 \leq \alpha_i \leq 1$. Also, using the DC power flow model, let the original reference angles be computed as follows:

$$\theta_i = \sum_{i=1}^{n-1} k_i P_i \qquad where \qquad k_i \geq 0 \tag{3.5}$$

Now, let k_i^+ and α_i^+ correspond to the positive power nodes P_i^+ (generators) and let k_i^- and α_i^- correspond to the negative power nodes P_i^- (loads). The sufficient condition for no cascade propagation is as follows.

Theorem 2 *Sufficient condition for no cascade propagation in Min-CFA problem is as follows:*

$$\frac{1 - min(\alpha_i^-)}{1 - max(\alpha_i^+)} \geq \frac{\sum k_i^+ P_i^+}{\sum k_i^- (-P_i^-)} \tag{3.6}$$

Proof In Section 1 (APPENDIX).

3.3.1.2 Recovery Phase (Max-R)

In the general cascading failure model, suppose that recovery of each failed power line $(ij) \in E_p^{B,t}$ leads to the restoration of $\sum P_{L_j}^k(Rep_k)$ power units in the

loads' demand. Where $Rep_k = \{(i,j) \in E_p^{W,k}\}$ is the set of restored and working power lines at iteration k. Also, suppose that at each iteration k of the recovery R_k resources are available and repairing (ij) needs r_{ij} resources. The maximum recovery (*Max-R*) optimization can be modeled as a mixed integer programming where we maximize the accumulative delivered power over K steps of the algorithm. Assuming that at each iteration we have enough resources to repair at least one disrupted edge, we set K to be the total number of disrupted edges. Let E_k^W be the set of lines which have been restored or are working up to time step k and let E_k^R be the set of restored edges up to iteration k. The *Max-R* recovery problem is formulated as follows:

$$\begin{aligned}
\text{maximize} \quad & \sum_{k=1}^{K} \sum_{L_j \in V_p} P_{L_j}^k(Rep_k) \ , \\
\text{subject to} \quad & \sum_{m=1}^{k} \sum_{(ij) \in E_k^R} \delta_{(ij),m}.r_{ij} \leq \sum_{m=1}^{k} R_m \quad k = 1, ..., K \ , \\
& \sum_{k=1}^{K} \delta_{(ij),k} \leq 1, \ \forall (ij) \in E_k^R \quad k = 1, ..., K \ , \\
& \delta_{(ij),k} \in \{0,1\}, \ \forall (ij) \in E_k^R \quad k = 1, ..., K \ ,
\end{aligned} \tag{3.7}$$

where $\delta_{(ij),k}$ is the decision variable to repair $(ij) \in E_p^B$ at the kth iteration of the algorithm. The first constraint indicates that at iteration k of the recovery, R_k resources are available; if the resources are not used in the k-th iteration of the recovery, the unused resources can be used in the following steps. The second constraint shows that each broken line can only be repaired once. Note that the total delivered power in the objective function changes with respect to the recovery schedule. The objective function is the accumulative power flow measured at the loads in the K steps of execution of the algorithm. With $P_{L_j}^k(Rep_k)$ we denote the power received by load L_j when the recovery decision $\delta_{(ij),k}$ is made up to step k leading to the restoration of the power lines Rep_k. One needs to re-solve the DC power flow optimization problem to find $\sum P_{L_j}^k(Rep_k)$ since the set of working lines, Rep_k, at time step k changes based on the current and previous decisions of the recovery schedule $\delta_{(ij),k}$. Note that in the recovery phase, we remove the generator's power reduction constraint and the generator and load's power increases gradually until all demand loads are satisfied.

Theorem 3 *The problem of Max-R is NP-Hard.*

Proof In Section 1 (APPENDIX).

As Max-R is NP-hard, we consider two polynomial time heuristics, (Max-R-shadow-pricing) and (Max-R-Backward) in Section 4.4.

Remark: Note that the maximum recovery problem is a combinatorial optimization and the total flow that each line can add to the final solution of the problem is unknown in advance and depends on the recovery schedule of other lines. The marginal flow that each line can add to the current solution of the problem can be found by solving the *Min-CFA* problem introduced in section 3.3.1.1 which itself is a linear programming optimization. We call the marginal utility (flow) added by recovery of each line the "*shadow price*" referring to the amount of flow assigned to the currently unknowable value of the flow that can be added to the final solution by repairing a broken line.

We now consider an example where the underlying communication network is disrupted and therefore, the controller fails to make appropriate decision to stop the cascade. We then propose a consistent failure set (*CFS*) algorithm in Section 3.4.1 to cope with lack of knowledge.

An illustrative example: Consider the network given in Figure 4.2, using the DC power flow model to calculate the power flows in the lines, where the reference angle is $\theta_1 = 0$, we have:

$$\begin{pmatrix} \theta_2^0 \\ \theta_3^0 \end{pmatrix} = \begin{pmatrix} 5 & -2 \\ -2 & 4 \end{pmatrix}^{-1} \begin{pmatrix} 1.5 \\ -2 \end{pmatrix} = \begin{pmatrix} 0.125 \\ -0.4375 \end{pmatrix} \tag{3.8}$$

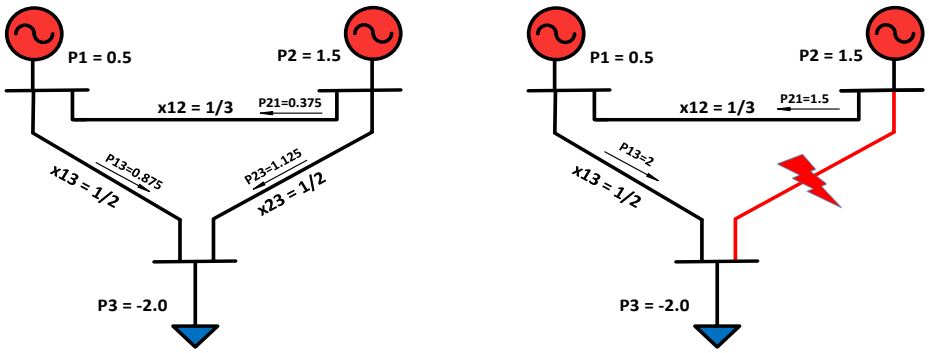
The power flow through each line is then computed as follows:

$$F_{12}^0 = \frac{\theta_{12}^0}{x_{12}} = 3 \times (0 - 0.125) = -0.3750, \tag{3.9}$$

$$F_{13}^0 = \frac{\theta_{13}^0}{x_{13}} = 2 \times (0 - (-0.4375)) = 0.875, \tag{3.10}$$

$$F_{23}^0 = \frac{\theta_{23}^0}{x_{23}} = 2 \times (0.125 - (-0.4375)) = 1.125. \tag{3.11}$$

If the power line 23 gets disrupted as in Figure 3.2b, the power redistributes



(a) All lines working. (b) Failure in line (23).

Figure 3.2: An example of a 3-bus network where active power and reactances are in pu.

according to DC power flow model, where $F_{21}^1 = 1.5$ and $F_{13}^1 = 2$. Suppose that the maximum power that each line can tolerate is $F_{ij}^{max} = 1.3$. Therefore, after the first line gets disrupted, the whole system collapses and the demand load cannot be satisfied. However, if we know the exact location of the failure, the controller may reduce the generator's power to satisfy a degraded quality of service. One trivial solution of *Min-CFA* to this problem is to reduce the second generator's power to $P_2^1 = 0.8$ and reduce the load to $P_3^1 = -1.3$ without violating the maximum power on each line. However, under the uncertainty of the exact location of the failure, the controller fails to make appropriate decisions and the whole network collapses.

3.4 Methodology

In this section, we first describe the consistent failure set approach to detect the status of grey lines. The grey lines, as descibed in Section 3.3 are the set of edges whose working status is unknown to the cotroller. Then, we describe two heuristic algorithms to solve Max-R. Inspired by the proposed approach in [16] that finds a progressive recovery schedule in a data communication network, we first propose a shadow price-based approach with polynomial time complexity and then propose a polynomial time backward approach that solves a single stage of the problem and traces back until it finds the recovery schedule for all stages.

3.4.1 Finding a Consistent Failure Set (CFS)

In order to detect the grey area, we use an algorithm, which starts with the nodes that have the smallest number of grey edges.

Lemma 4 *In the power grid graph G_p, if there exists a node $n_i \in V_p$ which has only one grey neighbor link $e = (n_i, n_j) \in E_p^t$, the exact status of the grey edge e can be discovered.*

Proof The exact status of a single grey edge attached to a node n_i can be determined using the power flow equation 3.2, i.e. the power generated/consumed at node n_i can be found by summing the power flow of all its adjacent power lines.

Lemma 5 *If the grey area does not contain a cycle and there exists at least one edge in the power grid graph G_p whose status is known, the exact status of all grey edges can be found in $O(|E_p^{U,t}|)$.*

Proof If the grey area does not contain any cycles, there exists at least one node that has only one grey neighbor link e and therefore, according to lemma 4, the exact status of e can be found. This procedure can be repeated to find the status of all grey edges in $O(|E_p^{U,t}|)$.

For the case study of a graph G_p which has one or multiple cycles in its grey area, we propose a consistent failure set rule to detect the exact status of unknown transmission lines. We assume the power generated/consumed in each generator/load or junction is known before the disruption. The algorithm starts by finding the status of grey edges, which are not within a cycle and are the only grey neighboring node of one of its end points. If all nodes have at least two grey edges in the graph, i.e. there exists a cycle in the grey area, and our algorithm picks a node within a cycle with the minimum number of adjacent grey edges and makes a decision tree. The algorithm tries to solve the unknown status of the grey edges by assuming one edge at each cycle to be working or not working, and solving the rest of DC power flow to see if the assumption is correct. If the assumption is not correct, the algorithm chooses another branch of the decision tree until it finds a consistent failure set. In cases where there exists multiple consistent failure sets, the algorithm performs a local inspection of an edge whose status is different from the possible solutions

Table 3.2: Average number of local inspections needed as the size of the grey area increases in the Italian power grid network.

Percentage of disrupted monitors	Average number of grey edges in the italian power grid	Average # of grey edges within a cycle
10	25.25	3.74
20	62.49	7.15
30	92.06	10.04
40	124.16	13.35
50	157.6	16.84
60	193.29	21.52
70	227.59	26.95
80	265.49	32.71
90	303.5	40.06

and picks the solution, which is consistent with the result of the local inspection. Algorithm 2 shows different steps of *CFS*.

Theorem 6 ***Complexity Analysis:*** *Assuming the grey area becomes a tree by removing C edges, CFS algorithm runs in $O(2^C|E_p^{U,t}|)$.*

Figure 3.3 shows an example of a network with 6 grey edges and shows different steps of the *CFS* algorithm. In the first step, the status of all edges with a single grey adjacent edge are designated. In the second step, the decision tree makes two branches to remove the cycle, and solves the DC flow optimization for each branch to find a consistent failure set. Assuming edge $(23) \in E_p^{B,t}$ was broken, we do not find a consistent feasible solution and therefore we assume $(23) \in E_p^{W,t}$ is working. The last graph shows a consistent failure set of broken and working edges.

In cases where we have multiple consistent failure sets, we perform a local inspection of the edges whose failure status is different from the possible consistent solutions, and pick the solution consistent with the local inspection.

Table 3.2 shows the average number of grey edges within a cycle, in the Italian power grid network [27–29] when the size of the disrupted communication network (garr) increases from 10% to 90% for 100 different random selection of disrupted communication nodes. Assuming all possible failures within a cycle are consistent with known information, we only need a maximum of 10% local inspection of the grey edges.

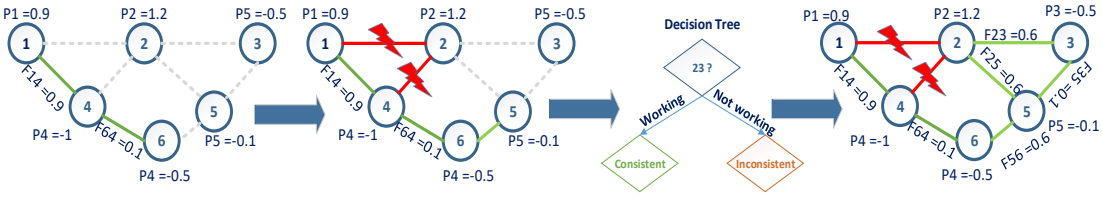


Figure 3.3: An example of a 6-bus network with 6 grey edges and different steps of CFS algorithm.

Algorithm 2: Consistent Failure Set (CFS) algorithm.

Data: A set of grey lines $(ij) \in E_p^{U,t}$ whose failure status is unknown, the graph of the network $G_p = (V_p^t, E_p^t)$, the power generated at each generator P_{G_i} $\forall G_i \in V_p^t$, the power consumed at each load P_{L_i} $\forall L_i \in V_p^t$

Result: The status of edges in the grey area $(ij) \in E_p^{U,t}$, which can be failure or working.

$C =$ `Number of edges in` $E_p^{U,t}$ `that need to be removed to make the grey area cycle-free` **if** $C > 0$ **then**

    pick an edge at each cycle to generate a cycle-free grey area

for all 2^C combination of the chosen edges at each cycle, run CFS-Cycle-Free$(E_p^{U,t}, G_p, P_{G_i}, P_{L_i})$ to find a consistent failure set **if** $C = 0$ **then**

    else

        run CFS-Cycle-Free$(E_p^{U,t}, G_p, P_{G_i}, P_{L_i})$.

    return $E_p^{B,t}, E_p^{W,t}$

3.4.2 Identifiability of voltage phasors

When the network is divided into a known and unknown part we can re-write the DC power flow equations as follows:

$$\begin{pmatrix} B_{known} \\ B_{unknown} \end{pmatrix} \times \begin{pmatrix} \theta_{known} \\ \theta_{unknown} \end{pmatrix} = \begin{pmatrix} P_{known} \\ P_{unknown} \end{pmatrix} \tag{3.12}$$

Therefore, the unknown voltage phasors can be found as follows:

$$B_{known} \times \begin{pmatrix} \theta_{known} \\ \theta_{unknown} \end{pmatrix} = P_{known} \tag{3.13}$$

Algorithm 3: CFS-Cycle-Free

Function *CFS-Cycle-Free* $(E_p^{U,t}, G_p, P_{G_i}, P_{L_i})$
  $greys = argmin|(n_i j) \in E_p^{U,t}|$;
  while $greys = 1$ **do**
    Select a node $i \in V_p^t$ with one grey neighbor
     $greys = argmin|(ij) \in E_p^{U,t}|$;
    detect whether $(ij) \in E_p^{U,t}$ is working or not using equation 3.2.;
    if *there exists no solution from equation 3.2* **then**
      return inconsistent;
      break ;
    if $(ij) \in E_p^{U,t}$ *is working* **then**
      $E_p^{W,t} = E_p^{W,t} \cup (ij)$ and $E_p^{U,t} = E_p^{U,t} \setminus (ij)$;
    else
      $E_p^{B,t} = E_p^{B,t} \cup (ij)$ and $E_p^{U,t} = E_p^{U,t} \setminus (ij)$;
  return consistent, $E_p^{B,t}, E_p^{W,t}$;

Let $N = Null(B_{known})$ denote the null space of B_{known}, i.e., for any vector $n \in Null(B_{known})$, $B_{known}.n = 0$.

Theorem 7 *Voltage phasor θ_i is identifiable, if and only if $\forall n \in N$ we have $n_i = 0$.*

Therefore, in order to find a set of identifiable voltage phasors $\theta_i \in \theta$, we can first compute the null space of B_{known} and find all indices with zero values in the null space. The null space of B_{known} gives the number of identifiable voltage angles. If the value of the voltage phasor of θ_i is not identifiable, we have to perform a local inspection to find the value of voltage angles for non-identifiable nodes.

3.4.3 Identifying the failures

After identifying all voltage phasors, one can identify the unknown admittance matrix if the grey area does not contain any cycles.

$$B_{unknown} \times \begin{pmatrix} \theta_{known} \\ \theta_{unknown} \end{pmatrix} = P_{unknown} \tag{3.14}$$

Note that the value of the $P_{unknown}$ is determined from the previous state of the disruption. We assume the powers at the generators and loads are only controlled through the central controller unit and therefore since the controller has not increased or reduced the power $P^{t}_{unknonw} = P^{t-1}_{known}$. Therefore, we can find the state of the network for all grey edges, which are not inside a cycle. In case of having a grey cycle we use the consistent failure set algorithm to remove the cycles and find a consistent set. If the consistent failure set algorithm finds multiple solutions, we pick one by performing a local inspection.

3.4.4 Max-R-shadow-pricing

Since the total value of the flow that each repaired line can add to the solution is not known in advance, we use a shadow pricing technique, which is used to assign values to the unknown value of repaired edges in the power grid graph. At each stage k, the shadow-pricing algorithm repairs the transmission line $(ij) \in E_p^{B,t}$, which adds the maximum to the total delivered power over the required resource, i.e. $argmax_{(ij)}(F_{ij}/r_{ij})$, until the total available resources for stage k are used. Algorithm 4 shows different steps of the *Max-R-shadow-pricing* algorithm. The algorithm starts with the disrupted network and computes the value of the flow added to the current state of the network divided by the total number of resources it needs, and repairs the power line that maximizes this value. This procedure repeats until there are no more resources left to repair additional lines for the current stage.

3.4.5 Max-R-Backward

As an alternative to compute a more accurate solution of the *Max-R* problem, we use *Max-R-Backward*. The algorithm starts by solving a single stage of the problem assuming $R = R_1 + ... + R_K$ resources are available. The solution of this single stage algorithm is added to the set Rep_K, showing the set of edges, which should be repaired up to stage k. Then, the single stage is solved assuming $R = R_1 + ... + R_{K-1}$ resources are available, which gives the solution set Rep_{K-1}. The repaired lines, which are in the solution set of Rep_K and not in Rep_{K-1} will

Algorithm 4: Max-R-shadow-pricing recovery algorithm.

Data: A set of failed lines $(ij) \in E_p^{B,t}$, A set of demand loads $L_j \in V_p$ and generators $G_i \in V_p$, limit on the tolerable power of each transmission line F_{ij}^{max}, the nodal admittance matrix B, the required resources to repair each line r_{ij}

Result: The recovery schedule of the failed transmission lines $\delta_{(ij),k}$

1: $R = 0$ **for** $k \in \{1, ..., K\}$ **do**

2: $R = R + R_k$ **while** $\exists (ij) \in E_p^{B,t}$ *that* $r_{ij} \leqslant R$ **do**

3: Select an un-repaired line $(ij)^* = argmax_{ij} \frac{F_{(ij)}}{r_{(ij)}}$

4: $\delta_{(ij),k} = 1$

5: $R = R - r_{(ij)^*}$ **return** $\delta_{(ij)^*,k}$

Algorithm 5: Max-R-Backward recovery algorithm.

Data: A set of failed lines $(ij) \in E_p^{B,t}$, A set of demand loads $L_j \in V_p$ and generators $G_i \in V_p$, limit on the tolerable power of each transmission line F_{ij}^{max}, the nodal admittance matrix B, the required resources to repair each line r_{ij}

Result: The recovery schedule of the failed transmission lines $\delta_{(ij),k}$

1: solve DC power flow model to find F_{ij}, assuming all lines are working

2: $Rep_K = E_p^{B,t}$ **for** $k = K - 1$ *downto* $k = 1$ **do**

3: $R = \sum_{m=1}^{k} R_m$

4: $Rep_k = Rep_{k+1}$ **while** $\sum_{(ij) \in Rep_{k+1}} r_{ij} > R$ **do**

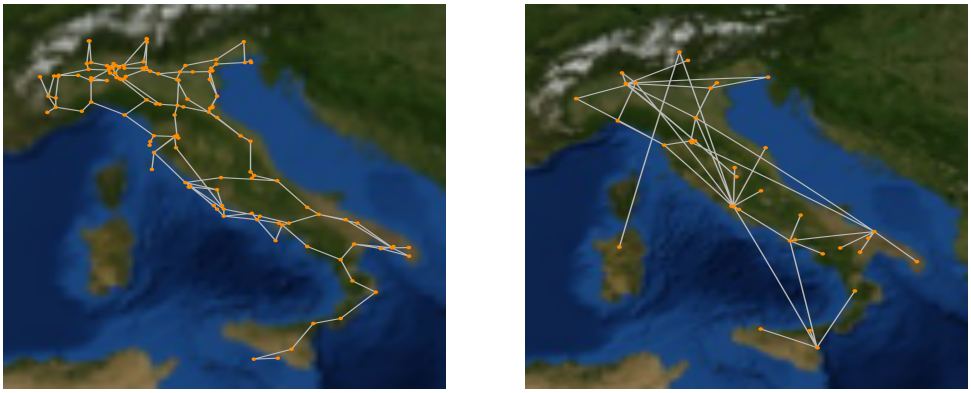
5: Select a line with minimum flow per cost $(ij)^* = argmin_{ij} \frac{F_{(ij)}}{r_{(ij)}}$

6: $\delta_{(ij),k+1} = 1$

7: $Rep_k = Rep_k \setminus (ij)^*$

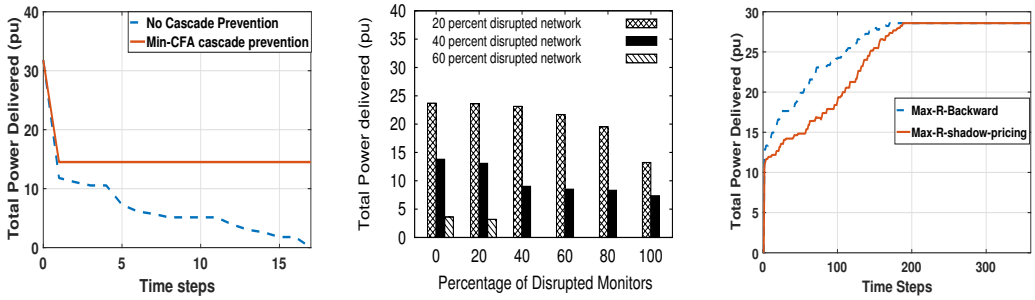
8: solve DC power flow model to find F_{ij}, assuming $(ij) \in Rep_k$ are working. **return** $\delta_{(ij)^*,k}$

be added to the repair schedule of stage K. This procedure repeats until the repair schedule of all stages is found.



(a) HVIET. (b) GARR.

Figure 3.4: a) The Italian high-voltage (380 kV) transmission grid (HVIET), and b) its communication network (GARR).

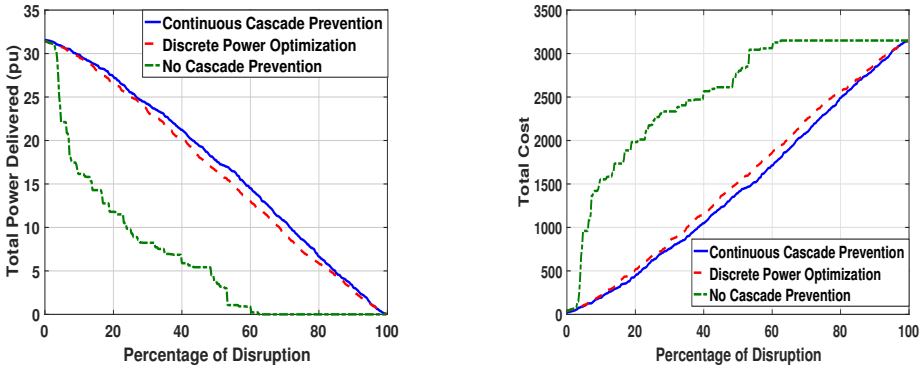


(a) Cascade prevention. (b) Sensitivity Analysis. (c) Recovery Phase.

Figure 3.5: a) Total delivered power (pu) during time when we use Min-CFA cascade prevention algorithm and without any cascade prevention in Italian power gird network, b) Total delivered power (pu) versus the percentage of failures on the monitoring nodes, c) Total delivered power (pu) flow over time for Max-R-Backward and Max-R-shadow-pricing in the Italian power gird.

3.5 Evaluation

In this section, we compare our *Min-CFA* cascade prevention approach presented in section 4.3, with a baseline algorithm, which does not include cascade prevention. We also compare the recovery performance of *Max-R-shadow-pricing* and *Max-R-backward* recovery approaches. We use the Italian power grid network shown in Figure 3.4a consisting of 310 nodes, 113 generators and 97 demand loads. The power grid has 361 power lines (edges). For the communication network we use the



(a) Total delivered power. (b) Total cost.

Figure 3.6: a) Total delivered power (pu), and b) total cost versus the percentage of disruption in the Italian power gird.

GARR network, shown in Figure 3.4b consisting of 39 nodes and 50 edges [28, 29]. Transmission lines in the power grid are monitored by several sensors deployed nearby. The aggregated data are then sent to the closest communication node and to the control center. Also, the control commands for power adjustment are sent to the closest communication node. Therefore, each node in the communication network might monitor and control several lines and nodes in the power grid. We implement our cascade prevention and recovery algorithms in python and used the Gurobi optimization toolkit, on a 120-core, 2.5 GHz, 4TB RAM cluster [25].

In the following experiments, we compare the total cost of failure and delivered power in cases where 1) there is no cascade prevention, 2) the cascade prevention can only turn a load on/off and 3) where we can reduce the load's demand continuously. For each scenario, we randomize the results running 10 different trials, where we vary the random selection of failed transmission lines.

3.5.1 Preventing the cascade (Min-CFA)

In the first set of simulations, we compare the performance of the *Min-CFA* cascade prevention algorithm with respect to the total cost and total delivered demand power. Similar to [43], we assume all loads have the same priority and give a high penalty for not being able to satisfy the demand. We assume the weighted cost of decreasing power of load L_j is 100, i.e. $w_{L_j} = 100 \quad \forall L_j \in L$, while the

normalized weighted cost of generators is 1, i.e. $w_{G_i} = 1 \quad \forall G_i \in G$.

In the first set of simulations we disrupt 60% of the transmission lines in the power grid and run Min-CFA to find the optimal flow assignment. The Min-CFA algorithm finds the optimal solution with 54.39% of the demand satisfied. On the other hand, if we do not run a cascade prevention algorithm, the failed transmission lines lead to more lines failing and this process continues until the whole system fails. Figure 3.5a shows the total delivered power during different time steps of the algorithms with Min-CFA cascade prevention and without it. As shown Min-CFA can save 45% of total power that could be delivered if the network was not disrupted.

In the next set of simulations, we use a continuous cascade prevention, meaning that $P^t_{L_j}$ in equation 5.1 can be decreased continuously. Then, we consider a discrete cascade prevention scenario, where each load's demand power should be satisfied or turned off; and finally, we consider a scenario, where there is no monitoring technique to reschedule the power flow or avoid the cascade and the failed transmission lines can trigger multiple cascade. Figures 3.6b and 3.6a show the simulation results for the 3 cases versus the percentage of disrupted network. As shown, the continuous cascade prevention approach saves more power compared to the discrete power optimization and when there is no information from the monitoring network no power can be delivered when 60% of the power lines are disrupted.

3.5.2 Sensitivity Analysis

In this section, we investigate the impact of incomplete knowledge about the exact location of failures. We consider a destroyed communication graph and make x% of the communication network destroyed (where we lose monitoring information). We then run the detection algorithm to remove the grey cycle-free edges. Next, we assume that the total grey area within the cycle is working (controller's belief about the grey area within the cycle which might not be correct), and then we run Min-CFA algorithm (to adjust the powers). Figure 3.5b shows the simulation results of this experiments. It is shown that when 20% of the power grid network get disrupted, the total delivered power can drop by 44.20% when all the monitors get disrupted. Assuming the maximum unitary profit of

26.6 €/MW according to [49], the total profit loss, due to uncertainty can be as high as 209076 € = $10.48pu \times 750MW/pu \times 26.6$ €/MW which could be avoided using a detection algorithm and a cascade prevention approach.

3.5.3 Recovery phase (Max-R)

In the next set of experiments, we compare the recovery performance of the proposed heuristics (Max-R-shadow-pricing and Max-R-Backward). Figure 3.5c shows the total delivered power flow over different steps of the algorithm when using the two algorithms. As shown, the shadow-pricing algorithm does not consider the correlation between different steps of the recovery approach and tries to maximize the added flow at each iteration step. On the other hand, the backward algorithm solves the problem using all repair resources in the beginning and removes the repair edges with less profit ($F_{ij}/r_{(ij)}$) from the schedule of previous stage until all repair schedules are determined. Therefore, Max-R-Backward performs better than the Max-R-shadow-pricing approach with larger total area behind the curve in Figure 3.5c.

3.6 Conclusion

This chapter studies the combined impact of large-scale failures on a power gird and its monitoring network. We propose a 2-phase mitigation strategy that 1) avoids further cascade while the system is in the transient state and 2) provide a maximum power flow recovery approach. We show that the maximum flow recovery problem (Max-R) is NP-Hard and intractable. Due to high complexity of the recovery problem, we propose two heuristic approaches (i) a shadow-pricing heuristic and (ii) a backward algorithm. It is shown that since the shadow-pricing heuristic does not consider the combined impact of repaired component, it performs poorly compared to the backward algorithm.

We also propose a consistent failure set (CFS) algorithm to cope with the uncertainty due to the failure of the dependent communication network that provides the information about the status of power lines being overloaded. We

show that *CFS* can find all failure sets given the information from the previous state of the network before the disruption and the incomplete information about the status of the lines. Our recovery approach and detection mechanism with incomplete information due to failure of the monitoring network is one of the first steps towards understanding the cascaded failures under uncertainty and opens up the area of power grid reliability approaches under incomplete or noisy information. We aim to extend our work to model the two way interdependency between the power grid and the communication network. Right now, our model assumes that the communication network gets power from an emergency source in case of failures in the power grid and ignores the dependency of the communication network to the power grid. We aim to modify this assumption and work on a more general dependency model.

Chapter 4

Parsimonious Tomography: Optimizing Cost-Identifiability Trade-off for Probing-based Network Monitoring

Network tomography using end-to-end probes provides a powerful tool for monitoring the performance of internal network elements. However, active probing can generate tremendous traffic, which degrades the overall network performance. Meanwhile, not all the probing paths contain useful information for identifying the link metrics of interest. This observation motivates us to study the optimal selection of monitoring paths to balance identifiability and probing cost. Assuming additive link metrics (e.g., delays), we consider four closely-related optimization problems: 1) *Max-IL-Cost* that maximizes the number of identifiable links under a probing budget, 2) *Max-Rank-Cost* that maximizes the rank of selected paths under a probing budget, 3) *Min-Cost-IL* that minimizes the probing cost while preserving identifiability, and 4) *Min-Cost-Rank* that minimizes the probing cost while preserving rank. While (1) and (3) are hard to solve, (2) and (4) are easy to solve, and the solutions give a good approximation for (1) and (3). Specifically, we provide an optimal algorithm for (4) and a $(1 - 1/e)$-approximation algorithm for (2). We prove that the solution for (4) provides tight upper/lower bounds on

the minimum cost of (3), and the solution for (2) provides upper/lower bounds on the maximum identifiability of (1). Our evaluations on real topologies show that solutions to the rank-based optimization (2, 4) have superior performance in terms of the objectives of the identifiability-based optimization (1, 3), and our solutions can reduce the total probing cost by an order of magnitude while achieving the same monitoring performance [11].

4.1 Introduction

Today's Internet traffic is massive, heterogeneous, and distributed, and continues to grow in these dimensions. Therefore, unlike small-scale networks, provisioning the desired services under an acceptable quality of service (QoS) for the ever-growing traffic sizes is extremely challenging and depends on continuous monitoring of the performance of individual links. Network monitoring provides the internal network state that is crucial for many network management functions such as traffic engineering, anomaly detection, and service provisioning. In cases where the important performance metrics are not directly observable (e.g., due to lack of access), network tomography [50, 51] provides a solution that infers these metrics from end-to-end probes. Compared to other monitoring techniques such as SNMP polling, *ping*, or *traceroute*, end-to-end probes does not need any special support from the routers [52–56] and is therefore a reliable tool for monitoring the Internet.

However, despite the considerable amount of research on estimating the individual link's performance metrics using given end-to-end measurements, the selection of which paths to probe, either to minimize probing cost or to satisfy a given bound (i.e., budget) on the probing cost, has not been thoroughly studied in prior works. Probing all possible paths between each pair of monitors can produce a tremendous amount of traffic in the network. Meanwhile, many paths contain redundant information due to shared links. In this chapter, we show that by carefully selecting the probing paths, we can significantly reduce the amount of probing traffic while achieving the same monitoring performance.

To this end, we consider the following closely-related problems under the assumption of additive performance metrics (e.g. delays): 1) the ***Max-IL-Cost*** problem that maximizes the number of identifiable links under a limited probing budget, 2)

the ***Max-Rank-Cost*** problem that maximizes the rank of probing paths under a probing budget, 3) the ***Min-Cost-IL*** problem that minimizes the probing cost while identifying all the identifiable links, 4) the ***Min-Cost-Rank*** problem that minimizes the probing cost while preserving the rank. Problems (1) and (3) are considered because they address, from different perspectives, the optimal trade-off between monitoring performance (measured by identifiability) and probing cost. Problems (2) and (4) are considered because they possess desirable properties that allow efficient computation while providing good approximations to (1) and (3).

Specifically, we make the following contributions:

1. Based on an existing algorithm that computes all the minimal sets of paths to identify each identifiable link, we convert (1) and (3) to problems similar to the max-coverage problem [57] and the set-cover problem [58], respectively. The conversion allows us to apply the greedy heuristic to these problems. We also propose an iterative branch-and-bound algorithm that treats our problems as integer linear programs (ILPs), and decomposes each problem into smaller meaningful sub-problems to exploit parallelism on a multi-core machine. Using our iterative branch-and-bound algorithm, we can configure the trade-off between the execution time and the optimality gap of the solution.

2. Using techniques from matroid optimization, we give polynomial-time solutions to (2) and (4) with guaranteed performance. The proposed solution for (4) is provably optimal, and the solution for (2) achieves a $(1 - 1/e)$-approximation.

3. We show that the solution for (4) provides tight upper/lower bounds for (3), and the solution for (2) provides upper/lower bounds for (1).

4. Our evaluations on real topologies show that in terms of the objectives of (1) and (3), the solutions proposed for (2) and (4) perform very close to the optimal and even outperform the solutions designed for (1) and (3). Compared to the baseline of probing all the candidate probing paths, our solutions can reduce the probing cost by an order of magnitude while achieving the same monitoring performance.

We first discuss the background and motivation behind this work in Section 4.2. In section 4.3, we formulate the four optimization problems. Section 4.4 contains our algorithms and their performance analysis. Section 4.5 shows our evaluation methodology and results. Finally, Section 5.3 concludes the chapter.

4.2 Background and Motivation

4.2.1 Background

The problem of designing the monitoring system to optimize the trade-off between cost and monitoring performance is a long-standing hard problem. If monitors cannot control the routing of probes, the problem is to place the minimum number of monitors (beacons) to identify all the links, which is proved to be NP-hard [59,60]. If monitors can control the probing paths (e.g., via source routing or software-defined networking), the problem is to both place the minimum number of monitors and construct the minimum number of probing paths to identify all the links, which is polynomial-time solvable [61–63]. In contrast to [61–63], we assume that routes cannot be controlled, as is usually the case in IP networks; in contrast to [59, 60] that focus on the offline cost for deploying monitors, we focus on the online cost for sending probes (i.e., the probing cost).

In the context of overlay networks, Chen et al. [64] show that monitoring a set of $O(nlog(n))$ paths is sufficient for monitoring an overlay network of n hosts, by selecting a set of paths that gives a basis of all the paths between the hosts. Li et al. propose a polynomial-time path selection algorithm that minimizes the total cost of selected paths to cover all the links [65]. These approaches differ from ours in that they focus on end-to-end performance, while we focus on identifying the performance of individual links.

Zheng et al. [66] introduced a problem similar to our third optimization (*Min-Cost-IL*). They study the problem of selecting the minimum number of probing paths that can uniquely identify all the identifiable links and cover all the unidentifiable links. Our formulation differs from theirs in that we allow general probing costs for the paths, and do not require coverage of all the links. These differences allow us to model paths with heterogeneous probing costs and further reduce the total cost

Table 4.1: Cost Reduction Using Selected Probing Paths

network name	#monitors	#paths	total cost	#selected paths	cost of selected paths
Abilene	11	55	244	12	46
BellCanada	20	190	4467	31	284
CAIDA	34	528	25553	56	1606

without losing identifiability. More importantly, the solution in [66] requires the calculation of all the irreducible path sets to identify each of the identifiable links, which has an exponential complexity. In contrast, we show that using rank as a proxy of identifiability gives an efficient solution that provides tight upper/lower bounds on the optimal solution (Theorem 14).

4.2.2 Motivation

We use an example to illustrate the cost saving that can be achieved by a careful selection of probing paths. Suppose that the cost of probing a path is equal to the total number of links on this path, which represents the amount of traffic that each probe on this path will generate. We consider three networks from the Internet Topology Zoo [4, 5], randomly select a subset of nodes in each network as monitors, and compute the shortest paths between each pair of monitors as the candidate probing paths. As shown in Table 4.1, probing all these paths generates a large number of transmissions and incurs a high cost (total cost). In contrast, using a selected subset of paths that preserve the rank (computed by Algorithm 7), we can obtain the same information at a much lower cost (cost of selected paths). As is shown, using the selected paths reduces the probing cost by a factor of 5.3–16 in this example. The large gap between the total probing cost and the probing cost of the given paths motivates the study of the path selection problem.

4.3 Problem Formulation

In this section, we describe our network model, performance measures and optimization problems.

4.3.1 Network Model

Given an undirected graph $G(V, L)$, where V represents the network nodes and L is the set of communicating links connecting the nodes, and a set of nodes $M \subseteq V$ employed as monitors, the set P of routing paths between all pairs of monitors specifies the set of candidate probing paths that we can select from. In our model, we assume IP packets from a source node s to a destination node t are being forwarded using a pre-determined routing algorithm. Our formulation and solutions support arbitrary routing algorithms, and the specific algorithm used for evaluation will be specified later (see Section 4.5). We denote a routing path r in G with a list of edges $r = \{e_1, ..., e_n\}$ and denote with k_r the cost of path r. Table 4.2 summarizes the notation used in our formulation.

4.3.2 Measures of Monitoring Performance

We use **identifiability** and **rank** functions to measure the monitoring performance of our path selection algorithms. Specifically, given a set P of all possible probing paths (e.g., routing paths between all the monitors), let A be the routing matrix of size $|P| \times |L|$, such that if path $r \in P$ contains link j, then $A[r, j] = 1$ and $A[r, j] = 0$ otherwise. Then the rank of P is calculated by the rank of A, denoted by $rank(A)$. Let $N = Null(A)$ denote the null space of A, i.e. for any vector $n \in Null(A)$, $A \cdot n = 0$. The next lemma specifies how to compute the set of identifiable links given A.

Lemma 8 *[64] For an arbitrary routing matrix A, let N represent the null space of A. Link $l_i \in L$ is identifiable, if and only if $\forall n \in N$ we have $n_i = 0$.*

Therefore, to find the set of identifiable links, $L_I \in L$, we first compute the null space of A and find all indices with zero values in the null space. The identifiability achieved by probing P is then the cardinality of L_I.

Table 4.2: Notation used in our formulations.

Notation	Explanation
$G(V, L)$	an undirected graph where V represents the set of nodes and L is the set of links
L_I	the set of identifiable links using all possible paths P
M	set of nodes where the monitors are located
$I(P_R)$	set of all identifiable links using paths in P_R
K	limit on the probing cost
$P_l := \{P_{l_i} : i = 1, ..., S_l\}$	set of all minimal solutions to link $l \in L$
X_l	decision to select an identifiable link l (if $X_l = 1$) or not (if $X_l = 0$)
Y_r	decision to select a path r (if $Y_r = 1$) or not (if $Y_r = 0$)
Z_s	decision to select a minimal solution s (if $Z_s = 1$) or not (if $Z_s = 0$)
P	the total set of uncontrollable paths using all of the monitoring nodes M
P_R	a subset of all possible probing paths with indices in R
$r_{u,v}$	given a source node u and a destination node v and a predefined routing algorithm, $r_{u,v}$ gives the routing path from u to v
A_R	a routing matrix of size $\|R\| \times \|L\|$, such that if path $r \in R$ contains link j, then $A_R[r, j] = 1$ and $A_R[r, j] = 0$ otherwise
$rank(A_R)$	the rank of routing matrix A_R
k_r	probing cost of path r
$c(P_R) = \sum_{r \in P_R} k_r$	total cost of a set of probing paths $P_R \subseteq P$
K	limit on probing cost

4.3.2.1 Relationship between Identifiability and Rank

While identifiability is a more accurate measure of the usefulness of the paths for network tomography, rank is easier to optimize as is shown later (see Section 4.4.2). Below, we establish the relationship between the two measures.

Let a set P of routing paths $\{r_1, ..., r_n\}$ be given. Corresponding to any subset $P_R \subseteq P$ of these elements, let $rank(A_R)$ be the rank of the routing matrix corresponding to the selected paths in P_R. We define L_1 to be any subset of identifiable links ($L_1 \subseteq L_I$) and provide the necessary and sufficient condition for a subset of paths to identify all identifiable links.

Theorem 9 *Let A_{*,L_1} be the sub-matrix of A by selecting all the columns corresponding to a subset of identifiable links $L_1 \subseteq L_I \subseteq L$, and $A_{*,L\setminus L_1}$ be the sub-matrix of A by selecting the columns corresponding to links in $L \setminus L_1$. A subset of paths $P_R \subseteq P$ with indices in R can identify all links in L_1 if and only if*

$$rank(A_{R,*}) = |L_1| + rank(A_{R,L\setminus L_1}) \ , \tag{4.1}$$

Proof In Section 1 (APPENDIX).

An illustrative example: Figure 4.1 shows an example of a network with 5 links and four candidate monitors $M = \{m_1, ..., m_4\}$. Using all possible paths between candidate monitors we have the following routing matrix.

$$A = \begin{array}{c} \begin{array}{ccc|cc} l1 & l2 & l3 & l4 & l5 \end{array} \\ \left(\begin{array}{ccc|cc} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{array}\right) \begin{array}{l} : r_{m_1,m_2} \\ : r_{m_1,m_3} \\ : r_{m_1,m_4} \\ : r_{m_2,m_3} \\ : r_{m_2,m_4} \\ : r_{m_3,m_4} \end{array} \\ \begin{array}{ccc|cc} & A_{*,L_1} & & A_{*,L\setminus L_1} & \end{array} \end{array}$$

The rank of this matrix is 4 while the null space shows only 3 identifiable links $l1, l2, l3$. If we only probe paths in $R = \{r_{m_1,m_2}, r_{m_1,m_3}, r_{m_2,m_3}\}$, the corresponding routing matrix A_R satisfies Theorem 9.

$$A_R = \begin{array}{c} \begin{array}{ccc|cc} l1 & l2 & l3 & l4 & l5 \end{array} \\ \left(\begin{array}{ccc|cc} 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \end{array}\right) \begin{array}{l} : r_{m_1,m_2} \\ : r_{m_1,m_3} \\ : r_{m_2,m_3} \end{array} \\ \begin{array}{ccc|cc} & A_{R,L_1} & & A_{R,L\setminus L_1} & \end{array} \end{array}$$

Meanwhile, it is also clear that probing these paths suffices to identify l_1, l_2 and l_3. We can solve the identifiable links using Gaussian elimination, where the reduced

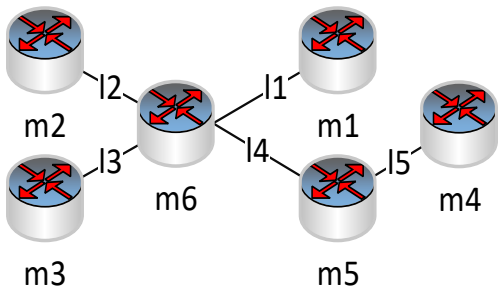


Figure 4.1: A simple network example with 5 links and 4 monitors $\{m_1, \ldots, m_4\}$. Candidate paths: r_{m_1,m_2}, r_{m_1,m_3}, r_{m_1,m_4}, r_{m_2,m_3}, r_{m_2,m_4}, r_{m_3,m_4}.

row echelon form ($rref(A)$) is

$$\left(\begin{array}{ccc|cc|l} 1 & 0 & 0 & 0 & 0 & l_1 : (r_{m_1,m_2} + r_{m_1,m_3} - r_{m_2,m_3})/2 \\ 0 & 1 & 0 & 0 & 0 & l_2 : (r_{m_1,m_2} + r_{m_2,m_3} - r_{m_1,m_3})/2 \\ 0 & 0 & 1 & 0 & 0 & l_3 : (r_{m_1,m_3} + r_{m_2,m_3} - r_{m_1,m_2}/2 \\ \hline 0 & 0 & 0 & 1 & 1 & l_4 + l_5 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right)$$

As shown, the reduced row echelon form contains an identity matrix for columns corresponding to identifiable links and by choosing $\{r_{m_1,m_2}, r_{m_1,m_3}, r_{m_2,m_3}\}$, the conditions of theorem 9 are satisfied. Further, the total probing cost reduces from 15 to 6.

4.3.3 Optimization Problems

4.3.3.1 Max-IL-Cost problem

Let $I(P_R)$ be the set of identifiable links using paths in P_R and $|I(P_R)|$ be the number of identifiable links using paths in P_R. The constrained path selection optimization problem aims at maximizing the number of identifiable links (*Max-IL-*

Cost) with a limited probing cost K, which can be formulated as follows:

$$\text{Maximize} \quad |I(P_R)| \tag{4.2a}$$

$$\text{subject to} \sum_{r \in P_R} k_r \leqslant K, \tag{4.2b}$$

$$P_R \subseteq P, \tag{4.2c}$$

where k_r is the probing cost of path r. As a concrete example, we can define the probing cost of each path to be its total number of hops. Then the total probing cost represents the total number of transmissions generated by probing the selected set of paths.

ILP formulation for Max-IL-Cost: To better understand properties of Max-IL-Cost, we re-write it as an integer linear programming (*ILP*). The basis of our formulation is the notion of *minimal solutions* (simply called *solutions* in [66]). Each minimal solution to link $l \in L$ is a subset of paths $P' \subseteq P$ such that (i) P' can identify l, but (ii) no proper subset of P' can identify l. As an example, consider the network in Figure 4.1. Consider the following two sets of paths $P_1 = \{r_{m_1,m_2}, r_{m_1,m3}, r_{m_2,m_3}\}$ and $P_2 = \{r_{m_2,m_3}, r_{m_2,m_4}, r_{m_3,m_4}\}$, which are both minimal solutions to link l_2.

We can compute all the minimal solutions for each link l by first finding a solution to l and then use a linear replacement method to generate other solutions, as described in [66]. Let $\mathcal{P}_l$ be the set of all the minimal solutions to link $l \in L$ ($\mathcal{P}_l = \emptyset$ if l is not identifiable). Then $\mathcal{P} = \bigcup_{l \in L} \mathcal{P}_l := \{P_s\}_{s \in S}$ is the collection of all the minimal solutions for the identifiable links. For ease of presentation, we index the solutions in $\mathcal{P}$ and denote by S the set of solution indices.

Based on the minimal solutions, we can write the Max-IL-Cost problem as the following ILP:

$$\underset{X_l, Y_r, Z_s}{\text{Maximize}} \quad \sum_{l \in L} X_l \tag{4.3a}$$

$$\text{subject to } X_l \leqslant \sum_{s: l \in I(P_s)} Z_s, \qquad \forall l \in L, \tag{4.3b}$$

$$\sum_{r \in P} Y_r \cdot k_r \leqslant K, \tag{4.3c}$$

$$Z_s \leqslant Y_r, \qquad \forall s \in S, r \in P_s, \tag{4.3d}$$

$$X_l, Y_r, Z_s \in \{0,1\}, \qquad \forall l \in L, r \in P, s \in S. \tag{4.3e}$$

Here the binary variables X_l, Y_r and Z_s respectively represent the decision to select an identifiable link (if $X_l = 1$), a probing path (if $Y_r = 1$), and a minimal solution (if $Z_s = 1$).

First, we show that given solutions to Y_r's, the ILP is easy to solve.

Lemma 10 *The ILP optimization problem can be relaxed over the integer variables X_l and Z_s and still gives an optimal integer solution.*

Proof In Section 1 (APPENDIX).

Remark: While the problem can be relaxed over X_l and Z_s, finding all minimal solutions has an exponential complexity. Furthermore, similar to [66], optimizing Y_r's is hard to solve. Therefore, we use the rank function as a proxy to identifiability in Section 4.3.3.2 and show the upper/lower bounds for identifiability.

4.3.3.2 Max-Rank-Cost Problem

Creating all the minimal solutions in Max-IL-Cost has an exponential complexity which limits the scale of applicability to small networks. Therefore, we replace the identifiability measure in this problem by rank. The resulting optimization derived from Max-IL-Cost, referred to as *Max-Rank-Cost*, is formulated as follows:

$$\text{Maximize} \quad rank(A_R) \tag{4.4a}$$

$$\text{subject to} \quad \sum_{r \in P_R} k_r \leq K, \tag{4.4b}$$

$$P_R \subseteq P. \tag{4.4c}$$

The rank function has an important property that makes its maximization easy to solve. To this end, we introduce the following definition.

Submodularity Let P be a finite ground set. A set function $f : 2^P \to \mathbb{R}$ is

submodular if for all sets $P_a, P_b \subseteq P$, we have

$$f(P_a \cup P_b) + f(P_a \cap P_b) \leqslant f(P_a) + f(P_b). \tag{4.5}$$

Intuitively, f is a *submodular* function if it has the property of *diminishing return*, i.e., the marginal gain of adding an element e to a set P_a is at least as high as the marginal gain of adding e to any superset of P_a.

The significance of this property is that if $f(P)$ is monotone (i.e., increasing as we add elements to P) and submodular, then there is a generic greedy algorithm in [57] for maximizing $f(P)$ subject to a budget on P, which is within a $(1 - 1/e)$-factor of the optimal. It is known that the rank function is submodular.

Lemma 11 *[67] The rank function is monotone and submodular.*

However, the number of identifiable links $|I(P)|$ is not submodular. To see this, consider the example in Figure 4.2, which shows a network with 4 monitoring nodes (m_1, m_2, m_3, m_4). Consider the following path sets: $P_a = \{l_2\}$, and $P_b = \{(l_1, l_2), (l_3, l_2)\}$, where (l_i, l_j) denotes a 2-hop path traversing links l_i and l_j. Then it is easy to see that $I(P_a) = \{l_2\}$, $I(P_b) = \emptyset$, $I(P_a \cup P_b) = \{l_1, l_2, l_3\}$, and $I(P_a \cap P_b) = \emptyset$. Thus,

$$|I(P_a \cup P_b)| + |I(P_a \cap P_b)| > |I(P_a)| + |I(P_b)|,$$

violating submodularity.

4.3.3.3 Min-Cost-IL Problem

The problem of preserving identifiability using minimum probing cost is the dual of Max-IL-Cost problem. As a special case, Zheng et al. [66] considered the same problem when $k_r = k$ (i.e. all the paths have an identical probing cost). They show that even the special case is NP-hard by giving a reduction from set cover problem. They proposed a heuristic-based approach to cover all links by enumerating all possible combination of equations/paths that can cover each identifiable link. The constructed bipartite graph is then used to select the minimum number of probing

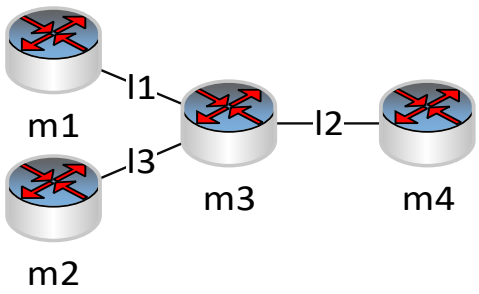


Figure 4.2: An example that shows identifiability is not a submodular or supermodular function.

paths that can cover all links where set cover is a special case of the problem. They assume each probing path has the same cost, while Min-Cost-IL allows non-uniform, heterogenous costs. Furthermore, while [66] also requires the coverage of non-identifiable links, our proposed algorithm only selects minimal sets that identify identifiable links. The constrained path selection optimization problem to minimize the probing cost to identify all identifiable links (*Min-Cost-IL*) is formulated as follows:

$$\text{Minimize} \quad \sum_{r \in P_R} k_r \tag{4.6a}$$

$$\text{subject to} \quad |I(P_R)| = |I(P)|, \tag{4.6b}$$

$$P_R \subseteq P. \tag{4.6c}$$

ILP formulation for Min-Cost-IL: Similar to (4.3), we re-write Min-Cost-IL as an integer linear programming (*ILP*) as follows:

$$\underset{Y_r, Z_s}{\text{Minimize}} \quad \sum_{r \in P} Y_r k_r \tag{4.7a}$$

$$\text{subject to} \quad 1 \leqslant \sum_{s: l \in I(P_s)} Z_s, \qquad l \in I(P_s), \tag{4.7b}$$

$$Z_s \leqslant Y_r, \qquad \forall s \in S, r \in P_s, \tag{4.7c}$$

$$Y_r, Z_s \in \{0, 1\}, \qquad \forall r \in P, s \in S. \tag{4.7d}$$

The optimization minimizes the total cost of selected paths. The first constraint indicates that at least one of the minimal solutions for each identifiable link should be selected. The second constraint indicates that if a minimal solution is selected, all paths in the minimal set should also be selected.

4.3.3.4 Min-Cost-Rank Problem

Similar to section 4.3.3.2, we define the *Min-Cost-Rank* problem as minimizing the probing cost (total hop-count) subject to preserving rank. Let $P = \{r_1, ..., r_{|M|(|M|-1)/2}\}$ be the total set of uncontrollable paths using all monitors M and let $P_R \subseteq P$ be a subset of selected paths. We define the routing matrix A_R of size $|R| \times |L|$ to be a matrix consisting of 0 and 1s, such that if $r \in P_R$ contains link j then $A_R[r, j] = 1$ and $A_R[r, j] = 0$ otherwise. We aim to select a subset of paths, $P_R \subseteq P$ such that the rank of both matrices be the same.

$$\text{Minimize} \quad \sum_{r=1}^{|M|(|M|-1)/2} k_r Y_r \tag{4.8a}$$

$$\text{subject to} \quad rank(A_R) = rank(A) \tag{4.8b}$$

$$Y_r \in \{0, 1\} \tag{4.8c}$$

Where the binary variable Y_r represent the decision to select a probing path r in A_R and k_r is the probing cost of path r.

4.4 Path Selection Algorithms

In this section, we give different algorithms for the four optimization problems. We propose a greedy heuristic and an iterative branch-and-bound algorithm for the *Max-Cost-IL* and the *Min-Cost-IL* problems. We also show a greedy algorithm that is optimal for *Min-Cost-Rank* and a modified greedy algorithm that achieves a $(1 - 1/e)$-approximation for *Max-Rank-Cost*.

4.4.1 Algorithms for Identifiability Optimization

4.4.1.1 Greedy-Max-IL-Cost and Greedy-Min-Cost-IL

We explain how to select a given set of paths using a set of feasible monitors and a pre-defined routing algorithm. To compare with the existing greedy-based heuristic which was proposed in [66], we construct a bipartite graph that reflects the coverage of probing path and the target links. Algorithm 6 shows a greedy-based approach for the mentioned bipartite graph model that iteratively chooses the set of paths that can identify more links with smallest cost. At each iteration step, the algorithm selects a minimal solution S_i that maximizes the value of the following function:

$$\frac{\text{New Identified Links in } S_i}{\text{Cost of New Paths in } S_i}, \tag{4.9}$$

where the numerator is the number of uncovered identifiable links that can be covered by selecting S_i and the denominator is the cost of unselected paths in the selected set S_i.

Remark: Greedy-Min-Cost-IL is similar to the greedy heuristic proposed in [66] but with two key differences. Unlike [66] that uses uniform cost for all selected paths, we allow an arbitrary cost for each path. Furthermore, unlike [66] that requires the selected paths to cover all links, we only require the paths to identify all the identifiable links.

We use a second greedy-based approach that we don't show (due to space limitation) for the dual problem (*Min-Cost-IL*) by changing the breaking condition. The breaking condition in line 3 of algorithm 6 is changed to $while(IL \neq I(P))$, meaning that we continue adding a new probing set S_i until all identifiable links are covered.

4.4.1.2 Iterative Branch-and-Bound

The ILP formulation (4.3, 4.7) allows us to apply general ILP solvers to *Max-IL-Cost* and *Min-Cost-IL*. Specifically, we use an iterative branch-and-bound algorithm [23] to achieve a configurable trade-off between complexity and optimality. For brevity, we explain the algorithm for maximization and minimization works

Algorithm 6: Greedy-Max-IL-Cost

Data: A set of feasible paths P, Limit on the number of paths K, Minimal combination of path sets that can identify an identifiable link l: $Z_s = \{S_l \quad l \in E\}$ where $S_l = \{r_i, ..., r_j\}$ is the set of paths that can identify link $l \in E$

Result: A set paths $P_R \subset P$ that maximizes the number of identifiable links in $G(V, E)$, A set of identified links $IL = \{l \in E\}$

$IL = \emptyset$;
$P_R = \emptyset$;
while $\exists S_l \in Z_s$ *that* $(K - \sum_{i=1}^{|P_R|} k_{r_i}) > (Cost\ of\ New\ Paths\ in\ S_l)$ **do**
    Select an un-selected set $S_i = argmax \frac{\text{New Identified Links in } S_i}{\text{Cost of New Paths in } S_i}$;
    for $i = 1$ to New Identified Links(S_i) $\quad IL = IL \cup l \quad l \in I(S_i)$;
    for $r_j \in S_i$
    $P_R = P_R \cup \{r_j\}$;
return IL and P_R

analogously.

The algorithm first removes the integrality restrictions. The resulting linear programming (LP) relaxation of Max-IL-Cost has a polynomial time complexity and gives an upper bound (UB) for the maximization. If the solution satisfies all the integral constraints, then we have the optimal solution. Otherwise, we pick a fractional variable, Y_r, and make two branches by creating one more constraint in the optimization: $Y_r = 0$ or $Y_r = 1$. We continue this procedure by making more branches to get closer to the optimal. The branch with the largest objective value that satisfies all the integrality constraints is called an *incumbent*. Also, at any iteration during the branch-and-bound algorithm we have a valid current upper bound, which is obtained by taking the maximum of the optimal objective values of all of the current leaf nodes. We stop branching once the gap between the incumbent's objective function (LB) and the current upper bound is smaller than a threshold (Gap), or we can stop branching after passing a given time limit. Optimality is achieved when the gap is zero. In the first case the algorithm gives a solution with an approximation ratio of $LB/(LB + Gap)$ since we have

$$\frac{LB}{OPT} \geq \frac{LB}{LB + Gap}. \tag{4.10}$$

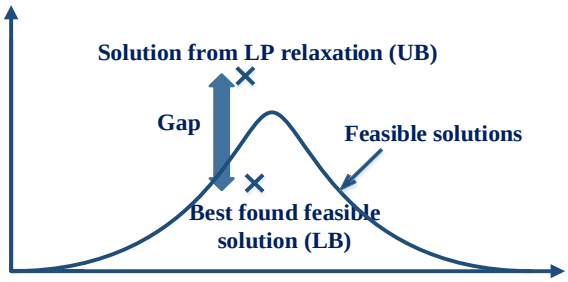


Figure 4.3: The iterative branch and bound algorithm that shows the gap between the incumbent and the upper bound for Max-IL-Cost.

In the second case, there is no guarantee on the approximation ratio but we have a guarantee on the execution time of the algorithm. Similarly, for a minimization problem (e.g., Min-Cost-IL), the incumbent (the branch with the smallest objective value and an integral solution) gives a upper bound (UB) on the optimal solution, and the LP relaxation gives a lower bound (LB). If the algorithm stops when $UB - LB \leq Gap$, then the incumbent gives a $UB/(UB - Gap)$-approximation since we have

$$\frac{UB}{OPT} \leq \frac{UB}{UB - Gap}. \tag{4.11}$$

The advantage of this algorithm is its flexibility. We can control the stopping rule of the branch-and-bound procedure to achieve trade-off between optimiality and complexity.

4.4.2 Algorithms for Rank Optimization

In this section, we propose two greedy-based approaches, called *Greedy-Min-Cost-Rank* for *Min-Cost-Rank* problem and *Greedy-Max-Rank-Cost* for *Max-Rank-Cost* optimization problem. We show that in terms of the rank objective, Greedy-Min-Cost-Rank provides an **optimal** solution for Min-Cost-Rank problem. In addition, Greedy-Max-Rank-Cost gives $1 - 1/e$ approximation for *Max-Rank-Cost* problem.

We first review the definition and properties of matroids [68] as they will prove

to be useful in the remainder of the chapter. Matroids play an essential role in combinatorial optimization and provide efficient and strong tool for solving computationally intractable problems.

Definition A *Matroid* is a pair $\mathcal{M} = \{L, \mathcal{I}\}$ of a finite ground set L and a collection $\mathcal{I} \subseteq 2^L$ of subsets of L such that [69, 70]:

- $\emptyset \in \mathcal{I}$
- $\forall I_x \subset I_y \subseteq L,\ if\ I_y \in \mathcal{I}\ then\ I_x \in \mathcal{I}$
- $\forall I_x,\ I_y \in \mathcal{I}\ ,\ |I_x| < |I_y|\ \rightarrow\ \exists r \in I_y \setminus I_x\ where\quad I_x \cup \{r\} \in \mathcal{I}$

We define $\mathcal{M} = \{P, \mathcal{I}\}$, where P is the set of all paths, $\mathcal{I}$ contains the sets $P_R \subseteq P$ such that paths in P_R are linearly independent.
We are able to achieve **optimal** solution for *Min-Cost-Rank* and $1 - 1/e$ **near-optimal** approximation solution for *Max-Rank-Cost*. The first is due to the fact that the sets of linearly independent paths form a matroid, and we are selecting a basis of this matroid with minimum cost. The approximation solution for *Max-Rank-Cost* is due to the submodularity of the rank function introduced in 4.3.3.2.

4.4.2.1 Greedy-Min-Cost-Rank

We now consider one of the interesting properties of matroids. We show that finding a maximal basis B of matroid, $\mathcal{I}$, of minimum weight can be solved optimally using a greedy-based heuristic. The greedy-based algorithm is similar to Kruskal's algorithm [71] that finds a minimum spanning tree in the graph. The algorithm iteratively adds a path with minimum cost to the set of selected paths until the rank of the selected paths is equal to the rank of the original routing matrix.

Theorem 12 *[68] For any routing path elements P and any probing cost function k_i, Greedy-Min-Cost-Rank (Algorithm 7) is optimal for Min-Cost-Rank, i.e., it returns a basis of P with the minimum probing cost.*

Complexity Analysis: Let $F(|P|)$ be the time complexity of testing whether a ground set is independent or not (line 5-6) which is the time complexity of checking whether the rank function is increasing or not. The Greedy-Min-Cost-

Rank algorithm runs in $O(|P|log(|P|) + |P|.F(|P|))$. Using Guassian Elimination algorithm to compute the rank function [72], that has a time complexity of $min(|L|, |P|) \times (|P| \times |L|)$ the complexity of the algorithm is $O(|L|^2 \times |P|^2)$, where $|P| = \frac{|M| \times (|M|-1)}{2}$.

Lemma 13 *If Greedy-Min-Cost-Rank returns a basis B for A_{*,L_I} where $rank(A_{B,L\setminus L_I}) = 0$, then B is the minimum cost set of paths that identifies all identifiable links, i.e. optimal solution to Min-Cost-IL.*

Proof In Section 1 (APPENDIX).

However, if *Greedy-Min-Cost-Rank* returns a minimum cost basis X for A_{*,L_I} where $rank(A_{X,L\setminus L_I}) = j \neq 0$ and the selected paths' cost is K_1, then K_1 is the lower bound for Min-Cost-Rank.

Theorem 14 *For any routing matrix A, and a set of identifiable links L_I, let Greedy-Min-Cost-Rank returns a basis $B_{A_{*,L_I}}$ for A_{*,L_I} with the minimum cost $K_{LB} = k_1 + k_2 + ... + k_{|L_I|}$, and let Greedy-Min-Cost-Rank returns a basis B_A for the routing matrix A with the minimum cost $K_{UB} = k'_1 + k'_2 + ... + k'_{rank(A)}$. Also let K^{opt} be the optimal cost solution of Min-Cost-IL, we have:*

$$K_{LB} \leq K^{opt} \leq K_{UB} \tag{4.12}$$

Proof In Section 1 (APPENDIX).

Remark: Note that the difference between the lower bound K_{LB} and the upper bound K_{UB} is no larger than $k'_{|L_I|} + ... + k'_{rank(A)}$. Since we have more constraint for selecting the first $|L_I|$ paths for A_{*,L_I} than A, we always have

$$k'_1 + k'_2 + ... + k'_{|L_I|} \leq k_1 + k_2 + ... + k_{|L_I|} \tag{4.13}$$

Therefore,

$$\begin{aligned} K_{UB} - K_{LB} &\leq k'_{|L_I|} + ... + k'_{rank(A)} \\ &\leq (rank(A) - |L_I|) * k'_{rank(A)} \end{aligned} \tag{4.14}$$

Algorithm 7: Greedy-Min-Cost-Rank approach for *Min-Cost-Rank* problem

Data: A set of uncontrollable paths $P = \{r_1, ..., r_{|M|(|M|-1)/2}\}$, a set of cost functions for each path $Cost = \{k_1, ..., k_{|M|(|M|-1)/2}\}$.

Result: A subset of paths $R^* \subseteq P$ that preserves the rank, i.e. $rank(R^*) = rank(P)$ with minimum probing cost.

```
P*_R = ∅;
TotalCost = 0;
sort P in increasing order of cost ;
forall r_i ∈ P do
    IncreaseRank_{r_i} = rank(P*_R ∪ {r_i}) − rank(P*_R) ;
    if IncreaseRank_{r_i} > 0 then
        P*_R = P*_R ∪ {r_i} ;
        TotalCost = TotalCost + k_i ;
    if rank(P*_R) ≥ rank(P) or |I(P*_R)| = |L_I| then
        break ;
return P*_R, TotalCost
```

4.4.2.1.1 Tightness of the bound For special routing matrices, the lower or upper bound is tight and coincides with the optimal for identifiability. To prove that, we first construct a routing matrix where the lower bound is tight. For this scenario, consider a routing matrix A, where the minimum cost basis $B_{A_{*,L_I}}$ for A_{*,L_I} does not pass any of the non-identifiable links (i.e. $rank(A_{B,L\setminus L_I}) = 0$). In this scenario, the lower bound is tight and coincides with the optimal. The minimum cost basis $B_{A_{*,L_I}}$ for A_{*,L_I} returned by *Greedy-Min-Cost-Rank* is always optimal for *Min-Cost-IL* (i.e., it identifies all links in L_I with minimum cost), if $rank(A_{B_{A_{*,L_I}},L\setminus L_I}) = 0$.

For the second scenario, we consider a network topology, where every monitor is connected to another monitor through one hop. Therefore, routing matrix is full rank and all links are identifiable. In this scenario, we need to select all paths to identify all links and thus the upper bound is tight and coincides with the optimal. The minimum cost basis B_A for A returned by *Greedy-Min-Cost-Rank* is always optimal for *Min-Cost-IL* if $rank(A) = rank(B_A) = |L_I|$.

4.4.2.2 Greedy-Max-Rank-Cost

Since the rank function is submodular, we can apply a modified greedy algorithm called *Greedy-Max-Rank-Cost* that gives $(1-1/e)$-approximation of the *Max-Rank-Cost* problem.

Algorithm 8 shows a Greedy-Max-Rank-Cost approach that enumerates all subsets of up to 3 paths, and iteratively augments each of these subsets by adding one path at a time to maximize the increment in rank per unit cost within the probing budget. The path set with the maximum rank is then selected as the overall solution. Since the rank function is monotone and submodular (Lemma 11), we can leverage an existing result for budgeted submodular maximization.

Theorem 15 *[57] Greedy-Max-Rank-Cost (Algorithm 8) achieves* $(1-1/e)$*-approximation for the Max-Rank-Cost problem, i.e., the rank of its solution* P *is no smaller than* $(1-1/e)$ *times the maximum rank.*

Complexity Analysis: In the worst case scenario, the algorithm has to find the maximum increase of the rank function $|P|^5$ times. Therefore, the complexity of the algorithm is $O(|P|^5 F(P))$. Where $F(P)$ is the complexity of calculating rank of P. Using Guassian Elimination algorithm to compute the rank function [72], the complexity of the algorithm is $O(|L|^2 \times |P|^6)$.

Algorithm 8 provides upper/lower bounds on the maximum identifiability that can be achieved under the given probing budget.

Theorem 16 *Let* P_R *be the set of paths returned by Greedy-Max-Rank-Cost for a probing budget* K*, which induces a routing matrix* $A_{R,*}$ *and identifies* I_R *links. Then the maximum number of links* I^{opt} *that can be identified under budget* K*, given by the optimal solution of Max-IL-Cost, is bounded by:*

$$I_R \leq I^{opt} \leq min\{rank(A_{R,*}) \cdot \frac{e}{e-1}, |L_I|\}, \tag{4.15}$$

where L_I *is the set of identifiable links using all possible paths* P*.*

Proof In Section 1 (APPENDIX).

Algorithm 8: Greedy-Max-Rank-Cost approach for *Max-Rank-Cost* problem

Data: A set of uncontrollable paths $P = \{r_1, ..., r_{|M|(|M|-1)/2}\}$, a set of cost functions for each path $Cost = \{k_1, ..., k_{|M|(|M|-1)/2}\}$, Limit on the probing cost K.

Result: A subset of paths $P_{Max} \subseteq P$ that maximizes $rank(P_{Max})$ subject to a limited monitoring cost K

$P_{Max2} = \emptyset$;
$TotalCost = 0$;
$P_{Max1} = argmax\{rank(P_x) : P_x \subseteq P, |P_x| \leq 3, c(P_x) \leq K\}$;
forall $P_g \subseteq P, |P_g| = 3, c(P_g) \leq K$ **do**
    $P' = P \setminus P_g$;
    $P = P_g$;
    while $P' \neq \emptyset$ **do**
        forall $r_i \in P'$ **do**
            $IncreaseBonus_{r_i} = (rank(P_{Max2} \cup \{r_i\}) - rank(P_{Max2}))/k_i$
        $r_{MaxIncrease} = argmax_{r_i \in P'} IncreaseBonus_{r_i}$
        if $k_{MaxIncrease} + TotalCost \leqslant K$ **then**
            $P = P \cup \{r_{MaxIncrease}\}$;
            $TotalCost = k_{MaxIncrease} + TotalCost$;
            $P' = P' \setminus (\{r_{MaxIncrease}\} \cup \{r \in P' : k_r > K - TotalCost\})$
    if $rank(P) > rank(P_{max2})$ **then**
        $P_{Max2} = P$
return $argmax_{P \in \{P_{Max1}, P_{Max2}\}} rank(P)$

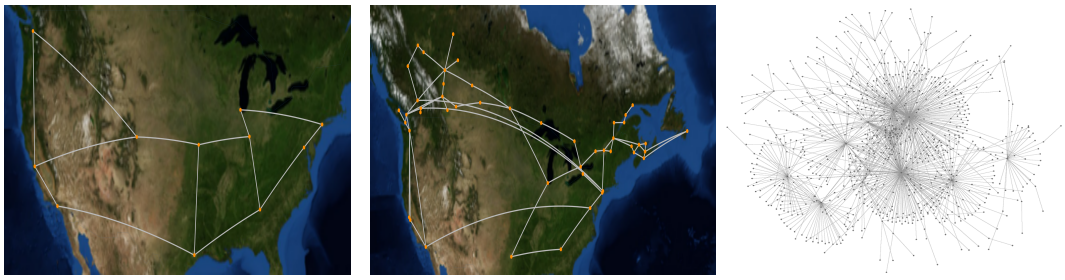
4.5 Evaluation

In this section, we consider several scenarios to compare the probing cost of our proposed algorithms compared to the case where we use all feasible probes or the optimal (OPT) brute-force solution. For each scenario, we randomize the results by running 10 different trials, where we vary the random selection of monitors from the entire set of nodes. We implement our low cost monitoring algorithms in python and used the *Gurobi* optimization toolkit, on a 120-core, 2.5 GHz, 4TB RAM cluster [25]. We assume shortest path routing (based on hop count), with ties broken arbitrarily.

We use different network topologies including a small, medium and a large real

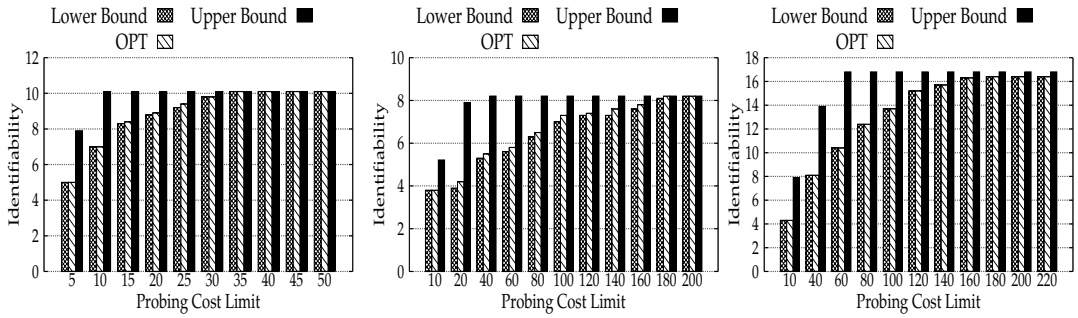
Table 4.3: Network characteristics used in our evaluation.

Network Name	# of nodes	# of edges	Average Node degree
Abilene	11	14	2.5
BellCanada	48	64	2.62
CAIDA	825	1018	2.46



(a) Abilene. (b) BellCanada. (c) CAIDA.

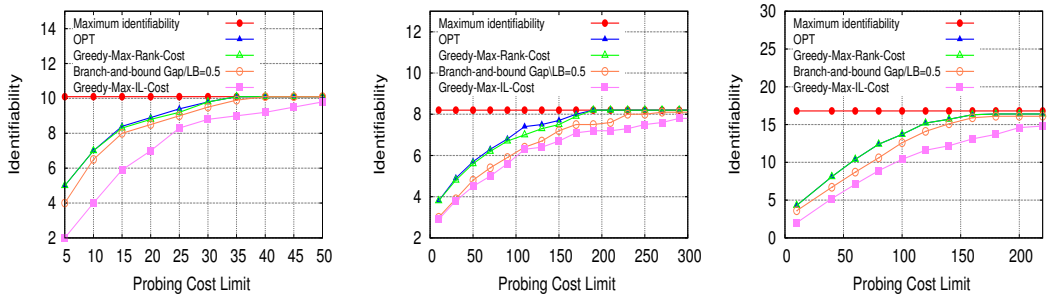
Figure 4.4: Network topology of graphs used in the evaluation a) Abilene, b) BellCanada and c) CAIDA topology.



(a) Abilene. (b) BellCanada. (c) CAIDA.

Figure 4.5: Upper and lower bound on the number of identifiable links as a function of limit on the probing cost in a) Abilene (9 monitors), b) BellCanada (10 monitors) and c) CAIDA topology (9 monitors).

topology taken from the Internet Topology Zoo [4, 5]. We also consider AS28717 (CAIDA) topology taken from the CAIDA (Center for Applied Internet Data Analysis) resource collection [73]. The network topologies used in our evaluation is shown in Figure 4.4. Table 4.3 shows the characteristics of the topologies used for the evaluation.



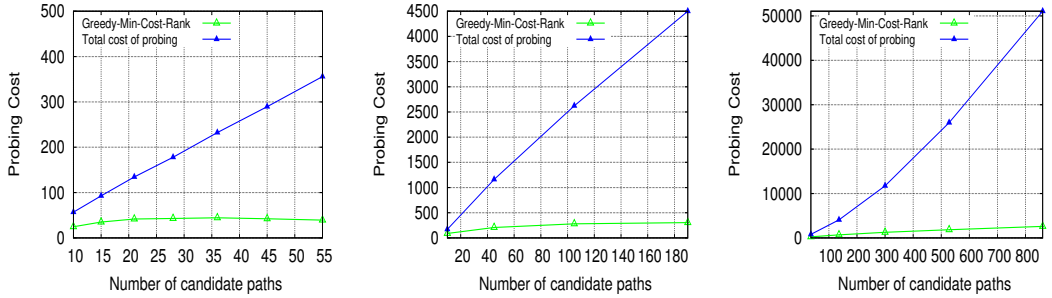
(a) Abilene. (b) BellCanada. (c) CAIDA.

Figure 4.6: Number of identifiable links as a function of limit on the probing cost in a) Abilene (9 monitors), b) BellCanada (10 monitors) and c) CAIDA topology (9 monitors).

4.5.1 Identifiability Maximization

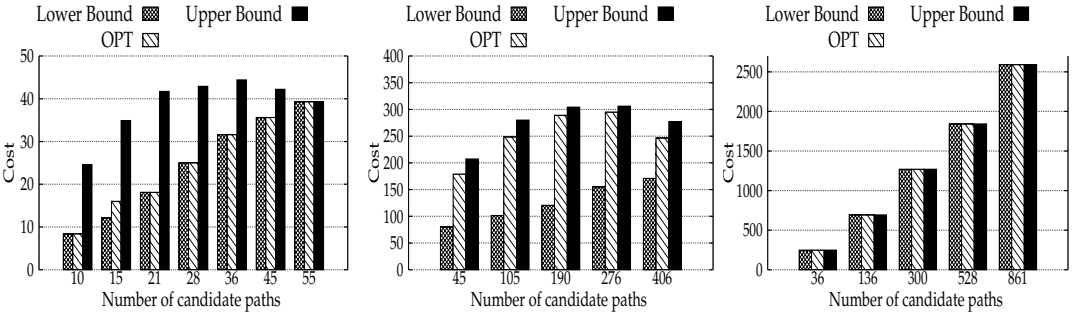
In the first set of simulations, we consider the impact of probing cost limit on the number of identifiable links. We first compare the upper and lower bound of Theorem 16, introduced in Section 4.4.2.2, with maximum number of identifiable links using all candidate paths. Figures 4.5a, 4.5b and 4.5c show the lower and upper bound on the number of identifiable links and the optimal number of identifiable links for each topology. We note that the optimal number of identifiable links is always upper bounded by the minimum of (i) maximum identifiability and (ii) the upper bound in Theorem 16. As shown, the difference between the optimal number of identifiable links in the upper and lower bound is very small which shows that Greedy-Max-Rank-Cost gives a solution close to optimal. Further, we note that the lower bound is closer to the optimal and gives a tighter bound in terms of number of identifiable links.

We next compare the number of identifiable links in Greedy-Max-IL-Cost heuristic (Algorithm 6), Greedy-Max-Rank, and the optimal case (OPT). We use gurobi optimization toolkit to solve the ILP problem formulation (equation 4.2). We also use our iterative branch-and-bound algorithm and stop the search when $Gap \leq 0.5 \cdot LB$. We recall that, the larger the gap is, the lower is the number of iterations of the optimization algorithm is and therefore we have an approximation of the solution which is farther from optimal. Figures 4.6a, 4.6b and 4.6c show scenarios where we increase the limit on the probing cost of monitors for the Abilene,



(a) Abilene. (b) BellCanada. (c) CAIDA.

Figure 4.7: Probing cost vs number of feasible probing paths in a) Abilene (5-11 monitors), b) BellCanada (10-29 monitors) and c) CAIDA (9-42 monitors) topology.



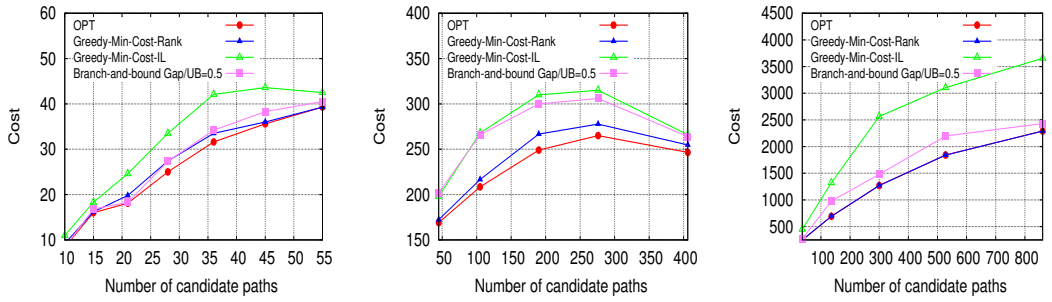
(a) Abilene. (b) BellCanada. (c) CAIDA.

Figure 4.8: Upper and lower bound on the probing cost as a function of number of candidate paths in a) Abilene (5-11 monitors), b) BellCanada (10-29 monitors) and c) CAIDA (9-42 monitors) topology.

BellCanada and CAIDA topology. As we increase the probing cost limit, more links are uniquely identified and all algorithm eventually converge to maximum identifiability, while Greedy-Max-Rank is closest to the optimal.

4.5.2 Cost Minimization

In the next set of simulations, we evaluate the performance of *Greedy-Min-Cost-Rank* algorithm that preserves rank and the greedy-based heuristics that preserve identifiability. We consider Abilene, BellCanada and CAIDA topology and run Greedy-Min-Cost-Rank and Greedy-Min-Cost-IL algorithms that preserve rank and identifiability respectively. We also run our branch-and-bound formulation and



(a) Abilene. (b) BellCanada. (c) CAIDA.

Figure 4.9: Number of identifiable links as a function of limit on the probing cost in a) Abilene (5-11 monitors), b) BellCanada (10-29 monitors) and c) CAIDA (9-42 monitors) topology.

stop branching once $Gap \leq 0.5 \cdot UB$. In the first set of experiments, we increase the number of candidate paths and evaluate the cost saving of our Greedy-Min-Cost-Rank algorithm with respect to the case where we use all candidate paths.

Figures 4.7a, 4.7b and 4.7c show the simulation results for this scenario. As shown, probing all candidate paths generates a large amount of traffic and incurs a high cost, while our Greedy-Min-Cost-Rank algorithm significantly reduces the cost. We also compare the accuracy of Greedy-Min-Cost-Rank by running the algorithm on (i) the entire routing matrix A, and (ii)a subset of the routing matrix with columns corresponding to the set of identifiable links A_{*,L_I}; the former gives the upper bound and the latter gives a lower bound on the probing cost according to Theorem 14. Figure 4.8 shows the upper and lower bound on the probing cost as we increase the number of candidate paths in each network topology. The simulation results for CAIDA topology, shows that the number of identifiable links is equal to the rank of the routing matrix A and thus the upper bound and lower bound are equal. Therefore, the solution to Greedy-Min-Cost-Rank for this topology is optimal. In Abilene and BellCanada topology, the lower bound and upper bound are closer to the optimal respectively.

We next evaluate the probing cost of each algorithm compared to optimal. Figures 4.9a, 4.9b and 4.9c show the probing cost of each network topology as we increase the number of candidate paths. As shown, our Greedy-Min-Cost-Rank algorithm is closer to the optimal in all topologies and coincides with the optimal in CAIDA.

4.6 Conclusion

This chapter studies the optimal selection of monitoring paths to balance identifiability and cost. We consider the constrained optimization problem of 1) maximizing identifiability under limited probing budget, 2) maximizing the rank function under a limited probing budget, 3) minimizing the probing cost subject to preserving identifiability, and 4) minimizing the probing cost subject to preserving the rank. While (1) and (3) are hard to solve, (2) and (4) posses desirable properties that allow efficient computation while providing good approximation to (1) and (3). We proposed an optimal greedy-based approach for (4) and proposed a $(1 - 1/e)$-approximation algorithm for (2). Our experimental analysis reveals that, compared to several greedy approaches, our rank-based optimization performs better in terms of identifiability and probing cost. Furthermore, our solution can reduce the total probing cost by an order of magnitude while achieving the same monitoring performance.

Chapter 5

Conclusion and Future Works

In this chapter, we first summarize our research contributions toward network recovery after large-scale failures in a communication or power grid network. We then outline several interesting future directions and some of the challenges and open problems that we aim to further explore in this dissertation. Finally, we briefly propose some of our ongoing research problems and primarily results.

5.1 Summary of Contributions

In this proposal, we provide comprehensive solutions to recover a network after massive disruption. We proposed novel schemes to monitor and recover a network under uncertain knowledge of failure while targeting four main goals: (1) Minimizing the number of necessary repaired elements, (2) Minimizing the amount of demand loss, (3) Minimizing the execution time and (4) Minimizing the cost of monitoring probes. These critical goals are in conflict with each other and we study the trade-off among them.

In the following we summarize the main contributions of each chapter.

- **Chapter 2:** In this chapter, we propose a progressive network recovery under uncertain knowledge of damages. The problem is formulated as a mixed integer linear programming (MILP) and is shown to be NP-Hard. The

proposed iterative stochastic recovery (ISR) approach recovers the network in a progressive manner while satisfying the critical service demands [9]. At each iteration step, ISR makes a decision to repair a part of the network and gathers more information by putting a monitor on the selected node. We propose several algorithms to find a feasible solution set at each iteration of the algorithm. Our results show that ISR outperforms the state-of-the-art ISP algorithm while we can configure our choice of trade-off between the execution time, number of repairs and demand loss.

- **Chapter 3:** We show that the inter-connectivity and dependency between different elements makes complex networks more vulnerable to failure [10]. We study the inter-dependency between a power grid and a communication network. We propose a failure mitigation and recovery strategy that first detects the failure and limits further propagation of the disruption by re-distributing the generator and load's power. We then formulate a recovery plan to maximize the total amount of power delivered to the demand loads during the recovery intervention. The cascade mitigation problem is formulated as a linear programming optimization that minimizes the cost of new flow assignment (*Min-CFA*) and aims at finding a DC power flow setting that stops the cascading failure at minimum cost. At the recovery phase, we aim at maximizing the restored accumulative flow. We show that the recovery problem (*Max-R*) is NP-Hard and propose heuristic recovery strategies that work under partial knowledge of damage locations. We propose a consistent failure set algorithm (*CFS*) to locate the failures.

- **Chapter 4:** We study the optimal selection of monitoring paths to balance identifiablity and cost [11]. To this end, we considered four closely related optimization problems: (1) *Max-IL-Cost* that maximizes the number of identifiable links under a probing budget, (2) *Max-Rank-Cost* that maximizes the rank of selected paths under a probing budget, (3) *Min-Cost-IL* that minimizes the probing cost while preserving identifiability, and (4) *Min-Cost-Rank* that minimizes the probing cost while preserving rank. We show that while (1) and (3) are hard to solve, (2) and (4) posses desirable properties that allow efficient computation while providing good approximation to (1) and (3). We proposed an optimal greedy-based approach for (4) and proposed a

$(1 - 1/e)$-approximation algorithm for (2). Our experimental analysis reveals that, compared to several greedy approaches, our rank-based optimization performs better in terms of identifiability and probing cost.

5.2 Future Directions

We provide several solutions to network recovery after large-scale failures in a communication network under uncertainty, and interdependent networks including a communication and a power grid. Our recovery approach and failure detection mechanism with incomplete information is one of the first steps towards understanding disruption management techniques under uncertainty and opens up the area of designing reliable systems under incomplete on noisy information. In our most recent ongoing research we address a method of updating flow rules for software defined networks with minimum disruption to the existing flows. In the following subsections, we outline several future works that we aim to explore in the future works and the primarily results of our ongoing research.

5.2.1 Minimum disruptive network updates

The emergence of Software Defined Networks (SDN) that decouples control plane and data plane, provides a powerful tool for network management, traffic engineering, and network policy enforcement. Decoupling the control functions and data plane elements brings significant advantages, including flexibility, being vendor agnostic, centralized control, programmability, and eliminating middleboxes [74, 75].

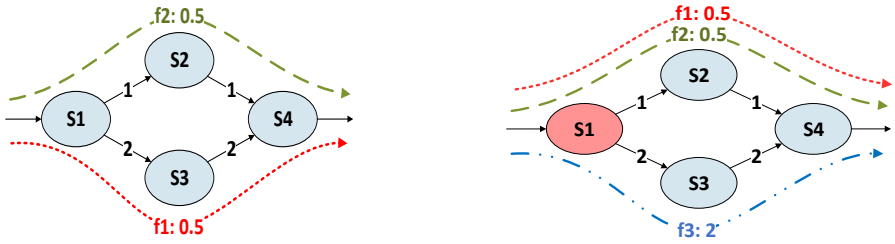
In an SDN paradigm, the controller monitors and controls network elements and defines the forwarding rules via an interface like the OpenFlow protocol [76, 77]. Routing decisions in OpenFlow switches are based on the flow tables implemented in ternary content addressable memory (TCAM). Each entry in the flow table consists of a set of matching rules associated with an action. Openflow switches are required to support *output*, *drop* and *group* actions [77]. Packets are matched based on IP source/destination pairs and forwarded to a single port according to the *output* action in the corresponding entry of the flow table. Packets whose output

action are not specified should be dropped, and the *Group* action processes the packets based on the specified group [77, 78].

While SDN provides a rich framework for packet forwarding, controlling and enforcing rule policies in data plane, network forwarding policies change very frequently due to new traffic demands, topology changes, network congestion or failures. One of the important challenges in software defined networking is the ability to react quickly to the network conditions (changing topologies, new traffic demands and other network re-configurations), which requires fast and safe update of the entries in the flow tables of switches. Updates of flow forwarding rules in SDN are the main source of service disruption (security vulnerabilities and instabilities in the network). Rule updates can disrupt the existing traffic by causing delays and opening security holes in the system [78–81]. Rule updates may result in two types of disruptions to an existing flow: 1) the disruption caused by re-routing the existing flow to accommodate new flows, and 2) the disruption caused by halting the existing flow due to rule updates on a switch traversed by the existing flow.

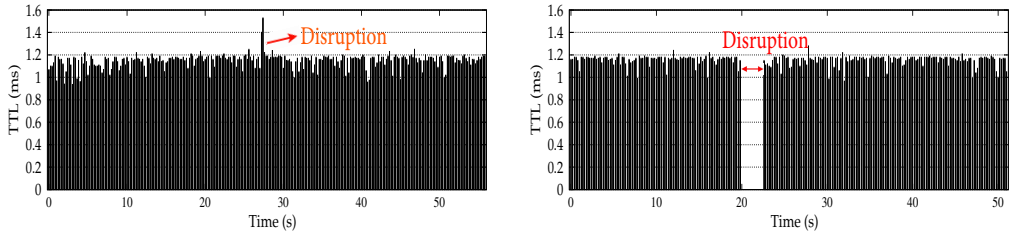
To clarify the discussion, consider a simple example shown in Figure 5.1. It shows a network with four switches and two flows f_1 and f_2. The upper links have a capacity of 1 unit of flow and the lower links have a capacity of two units of flow. Initially two flows exist in the network with 0.5 units of demand. Let us consider an update where a new flow f_3 which requires 2 units of demand flow arrives at the system. Due to the arrival of the new flow, f_1 needs to be re-routed from $S_1 - S_3 - S_4$ to $S_1 - S_2 - S_4$. In order to update the network from the original state to the target state, the controller updates the flow table in S_1 (marked with red in Figure 5.1b). Depending on the size of the flow tables, the update on this switch can take several hundreds of milliseconds to complete, causing disruption of flow f_1 for a long duration. Furthermore, updating S_1 can also cause a delay to the existing flow f_2 which is not being re-routed, but traverses the updated switch.

Figures 5.2a and 5.2b show experimental results for the two types of disruption. We first install rules on a 24-port Brocade (ICX 6610) SDN commercial switch and connect two computers. We then install 250 new rules on the SDN swicth and measure the round trip time (RTT) delays on the existing flow while installing the new rules. Figure 5.2a shows a spike in the RTT the existing flow while installing the new rules. The delay is about 31.5% higher than the average RTT delay before installing the new rules. In the next set of simulations we re-route the existing flow



(a) original state. (b) target state.

Figure 5.1: Rule update that causes the disruption of re-routed flows.



(a) Delayed flow. (b) Re-routed flow.

Figure 5.2: Disruption caused by a) delaying, b) re-routing an exisitng flow.

and measure the RTT. Figure 5.2b shows the disruption caused by rerouting the existing flow which disrupts the existing flow for about 2.6 seconds.

Motivated by these observations, we propose a minimum disruptive rule update that minimizes the two types of disruptions and satisfies the critical services in the network.

Minimum disruptive updates (Min-touch): We formulate the ***Min-touch*** optimization problem as an integer linear programming (ILP) as follows:

$$\text{minimize} \quad w_{high} \sum_{h \in H} \delta_h + w_{low} \sum_{h \in H} \theta_h$$

$$\text{subject to} \quad \delta_h + \theta_h < 2, \quad \forall h \in H \tag{5.1a}$$

$$\sum_{h \in H} (x_{ij}^h + x_{ji}^h) \cdot d_h \leq c_{ij}, \quad \forall (i,j) \in E \tag{5.1b}$$

$$\sum_{j \in V} x_{ij}^h = \sum_{k \in V} x_{ki}^h + sign(b_i^h), \quad \forall i \in V, \quad \forall h \in H \tag{5.1c}$$

$$\delta_h \geq \frac{\sum_{(i,j) \in E} [x_{ij}^h + k_{ij}^h - 2k_{ij}^h x_{ij}^h]}{|E|} \quad \forall h \in H \tag{5.1d}$$

$$\delta_h + \theta_h \geq \frac{\sum\limits_{j \in V} x_{ij}^h}{|V|} + t_i - 1, \quad \forall i \in V, \forall h \in H \tag{5.1e}$$

$$t_i \geq \frac{\sum\limits_{j \in V} \sum\limits_{h \in H} [x_{ij}^h + k_{ij}^h - 2k_{ij}^h x_{ij}^h]}{|V||H|} \quad \forall i \in V \tag{5.1f}$$

$$t_i \leq \sum_{j \in V} \sum_{h \in H} (x_{ij}^h + k_{ij}^h - 2k_{ij}^h x_{ij}^h), \quad \forall i \in V \tag{5.1g}$$

$$t_i, x_{ij}^h, \delta_h, \theta_h \in \{0, 1\}, \quad \forall (ij) \in E, \quad \forall h \in H \quad \forall i \in V \tag{5.1h}$$

where δ_h shows if an existing flow h gets re-routed due to the new update ($\delta_h = 1$) or not ($\delta_h = 0$), and θ_h shows that if an existing flow h perceives a delay due to the update in one of the switches which are used on the routing path in h. Since re-routing has more impact on the existing flows, as shown in Figure 5.2, we give higher weight to the re-routed flows than the flows which use the same route but get impacted due to a switch update $w_{high} \geq w_{low}$. The first constraint ensures mutual exclusion in counting the two types of disrupted flows δ_h and θ_h. Constraint 5.1b 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. Constraint 5.1c 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. Constraint 5.1d ensures that δ_h is set to 1 when flow h is re-routed. Constraint 5.1e together with the objective ensures that θ_h will be 1 if flow h is not re-routed but crosses and updated switch. The objective function will set θ_h to 1 when the value of δ_h is not forced to 1 by the previous constraint on δ_h, due to the lower weight in the linear combination for the minimization of the number of affected flows, according to the two different categories of "affected", the rerouted flows (captured by δ_h) and the non-rerouted, crossing altered switches flows (captured by θ_h). The last two constraints ensures that t_i will be 1 if there is a rule update on switch i.

5.2.2 Network Recovery under uncertainty

The progressive stochastic recovery approach in Chapter 2 minimizes the number of repairs. In a recent work, Ciavarella et al. propose a progressive recovery approach, called CeDAR, that aims at maximizing the accumulative flow over time

under recovery resource constraints [16]. We aim to extend our recovery approach to compare with CeDAR.

5.2.3 Power grid and communication network interdependency

We aim to extend our work in Chapter 3 to model the two way interdependency between the power grid and the communication network. Right now, our model assumes that the communication network gets power from an emergency source in case of failures in the power grid and ignores the dependency of the communication network to the power grid. We aim to modify this assumption and work on a more general dependency model.

5.3 Conclusion

This chapter first summarizes the main contributions of this dissertation and then outlines several interesting open problems and ongoing research that we aim to explore throughout this dissertation. We then study the minimum disruptive updates in software defined networking. We show that flow disruption and violation of policies can occur during an update in software defined networks. During an update the existing flows might get disrupted due to 1) being re-routed to admit new flows, or 2) being delayed due to an update in a switch on the path of an existing flow. We aim to minimize the two types of disruption and minimize the affected flows during the update, while taking into account the importance of various flows to upper-layer applications. We next proposed some of the future directions to extend our work in Chapters 2 and 3.

1 Appendix

Proof of Theorem 1 The nodal admittance matrix, B, of a connected graph with n nodes is always $rank(B) = n - 1$ because one can construct a graphic matroid from a given graph where the nodal admittance matrix is a weighted incident matrix. It is known that the rank of a weighted incident matrix is equal to the rank of any basis (tree) in the graph which is $n - 1$ [68, 82]. To make this equation solvable, one of the equations is removed and the node associated with that equation is chosen as a reference angle $\theta_1 = 0$. If the graph has c connected components, the rank of its admittance matrix is $n - c$. Therefore, the DC power flow model for each connected component of the graph has a unique solution.

Proof of Theorem 2 Suppose the power at each line is P_i. Using the power conservation constraint we have:

$$P_1 + P_2 + ... + P_n = 0 \tag{2}$$

$$P_n = -\sum_{i=1}^{n-1} P_i \tag{3}$$

Suppose that the in the new power schedule the power of each node is reduced by a factor of α_i where $0 \leq \alpha_i \leq 1$, we have:

$$\sum_{i=1}^{n} \alpha_i P_i = 0 \tag{4}$$

and,

$$\alpha_n = -\frac{\sum_{i=1}^{n-1} \alpha_i P_i}{P_n} = -\frac{\sum_{i=1}^{n-1} \alpha_i P_i}{-\sum_{i=1}^{n-1} P_i} \tag{5}$$

Next, we would like to know when the new power flow is larger than the previous power flow? From the DC power flow rule and assuming the reference voltage phasor is at the first node $\theta_0 = 0$, we have:

$$F_{i0} = \frac{\theta_i - 0}{x_{i0}} \tag{6}$$

and

$$\theta = (B)^{-1} P \tag{7}$$

Note that the inverse of the admittance matrix B is the impedance matrix with all positive values. Therefore, the new and previous reference angles at each edge is computed as follows:

$$\theta_i = \sum_{i=1}^{n-1} k_i P_i \qquad where \qquad k_i \geq 0 \tag{8}$$

$$\theta_{i,new} = \sum_{i=1}^{n-1} k_i \alpha_i P_i \qquad where \quad 0 \leq \alpha_i \leq 1, \quad k_i \geq 0 \tag{9}$$

Therefore, to check whether the new power flow after applying our control approach is larger than the previous flow we need to check if the following inequality holds or not:

$$\sum_{i=1}^{n-1} k_i P_i \geq \sum_{i=1}^{n-1} k_i \alpha_i P_i \tag{10}$$

Next, suppose that k_i^+, and α_i^+ are associated with the positive power nodes P_i^+ (generators) and k_i^- and α_i^- are associated with the negative power nodes (loads). For example, to check if the flow in F_{10}, F_{20} or F_{x0} increases after our power flow adjustment we need to check the following inequalities:

$$F_{10}: \quad \sum_{i=1}^{n-1} k_{1,i}^+ P_i^+ (1 - \alpha_i^+) \geq \sum_{i=1}^{n-1} k_{1,i}^- P_i^- (\alpha_i^- - 1) \tag{11}$$

$$F_{20}: \quad \sum_{i=1}^{n-1} k_{2,i}^+ P_i^+ (1 - \alpha_i^+) \geq \sum_{i=1}^{n-1} k_{2,i}^- P_i^- (\alpha_i^- - 1) \tag{12}$$

and,

$$F_{X0}: \quad \sum_{i=1}^{n-1} k_{X,i}^+ P_i^+ (1 - \alpha_i^+) \geq \sum_{i=1}^{n-1} k_{X,i}^- P_i^- (\alpha_i^- - 1) \tag{13}$$

Now we show that if the following condition is satisfied for every flow, we can make sure that the new power flow at each line is no more than the power flow without any adjustment of generator or load's power.

$$\frac{max(\alpha_i^+)}{min(\alpha_i^-)} \geq \frac{\sum k_i^+ P_i^+}{\sum k_i^- (-P_i^-)} \tag{14}$$

Or equivalently:

$$\frac{1 - min(\alpha_i^-)}{1 - max(\alpha_i^+)} \geq \frac{\sum k_i^+ P_i^+}{\sum k_i^- (-P_i^-)} \tag{15}$$

We have:

$$\sum_{i=1}^{n-1} k_i^+ P_i^+ (1-\alpha_i^+) \geq \sum_{i=1}^{n-1} k_i^+ P_i^+ (1-max(\alpha_i^+)) =$$
$$(1-max(\alpha_i^+)) \sum_{i=1}^{n-1} k_i^+ P_i^+ \geq (1-min(\alpha_i^-)) \sum_{i=1}^{n-1} k_i^- (-P_i^-) = \quad (16)$$
$$\sum_{i=1}^{n-1} k_i^- (-P_i^-)(1-min(\alpha_i^-)) \geq \sum k_i^- (-P_i^-)(1-\alpha_i^-)$$

This completes the proof!

Proof of Theorem 3 We prove the NP-hardness of the Max-R problem showing that it generalizes the Knapsack problem. We recall that the Knapsack problem considers a set of items I, each item $i \in I$ has a size S_i and a value $V_i > 0$. The problem is to find a subset $I' \subseteq I$ such that $S(I') \leq S$ and $V(I')$ is *maximized*, where $S(I') = \sum_{i \in I'} S_i$ and $V(I') = \sum_{i \in I'} V_i$.

In the following we show how we can build, in polynomial time, an instance of a single stage ($K = 1$) of Max-R problem whose solution corresponds to the solution of the generic formulation of the Knapsack problem given above.

Since we consider a single stage of the Max-R problem, we assume R resources are available to repair all disrupted lines $(ij) \in E_p^{B,t}$. We also assume that we have complete information about the disrupted lines. Let us consider a set of generators I, each generator corresponding to an element $i \in I$ of the Knapsack problem, producing a flow equivalent to the value V_i of the element. Each generator $i \in I$ is connected to a unique common load L with a broken line, whose repair cost is equivalent to the size S_i of the corresponding Knapsack element. We also assume that the load L has a demand of at least the summation of all flows ($\sum_{i \in I} V_i$). We set the recovery budget of Max-R equal to S, the size of the Knapsack. This instance of Max-R can be defined in polynomial time starting from any instance of Knapsack. Solving this instance of Max-R, corresponds to finding a list of links to be recovered with cost limited by S, such that the flow reaching the common load L is maximized, which is equivalent to selecting the Knapsack subset $I' \subseteq I$ with maximum value, and bounded size S, which completes the proof that any instance of the Knapsack problem can be polynomially reduced to the solution of an instance of Max-R, which implies the NP-hardness of Max-R.

Proof of Theorem 9 We first show the necessary condition, i.e. if a set of routing paths $P_R \subseteq P$ identifies all identifiable links in L_1, then $rank(A_{R,*}) = rank(A_{R,L_1}) + rank(A_{R,L\setminus L_1})$ and $rank(A_{R,L_1}) = rank(A_{*,L_1}) = |L_1|$. Suppose that the routing matrix $A_{R,*}$ is of size $n \times |L|$.

Without loss of generality (WLOG), suppose that the number of identifiable links in L_1 is $|L_1| = k$ and the first k columns of A_R correspond to these k identifiable links (one can exchange the columns in A_R to have this property). This means that the reduced row echelon form of $A_{R,*}$ should be as follows:

$$rref(A_{R,*}) = \left[\begin{array}{c|c} I_{k\times k} & 0_{k\times(|L|-k)} \\ \hline 0_{(n-k)\times k} & M_{(n-k)\times(|L|-k)} \end{array}\right] \tag{17}$$

Where, $I_{k\times k}$ is the identity matrix and $0_{k\times(|L|-k)}$ and $0_{(n-k)\times|L|}$ are matrices containing all zero entries and $M_{(n-k)\times(|L|-k)}$ is a matrix of general values. Therefore, the rank of $A_{R,*}$ is as follows:

$$rank(A_{R,*}) = k + rank(M) \tag{18}$$

It is clear that:

$$\begin{aligned} rank(A_{R,L_1}) &= k, \\ rank(A_{R,L\setminus L_1}) &= rank(M) \end{aligned} \tag{19}$$

Therefore,

$$\begin{aligned} rank(A_{R,*}) = k + rank(M) = \\ rank(A_{R,L_1}) + rank(A_{R,L\setminus L_1}) \end{aligned} \tag{20}$$

Next, we prove the sufficient condition, i.e. if for a selected subset of paths $P_R \subseteq P$ (4.1) is satisfied, then, P_R can solve all identifiable links.

Since $rank(A_{R,*}) \leq rank(A_{R,L1}) + rank(A_{R,L\setminus L1})$, and $rank(A_{R,L1}) \leq |L1|$, (4.1) implies that $rank(A_{R,L1}) = |L1|$, i.e., rows of A_{R,L_1} contain a basis of the row space of A_{*,L_1}. Therefore the reduced row echolen form of $A_{R,*}$ should contain the

identity matrix $I_{k\times k}$ as follows:

$$rref(A_{R,*}) = \left[\begin{array}{c|c} I_{k\times k} & B_{k\times(|L|-k)} \\ \hline C_{(n-k)\times k} & M_{(n-k)\times(|L|-k)} \end{array}\right] \tag{21}$$

We show that the submatrices $B_{k\times(|L|-k)}$ and $C_{(n-k)\times k}$ must be zero matrices. If $C_{(n-k)\times k}$ contains a non-zero entry, we can make them zero by using a sequence of elementary row operations. Note that (4.1) implies that

$$\begin{aligned} rank(rref(A_{R,*})) = rank(&\left[\begin{array}{c} I_{k\times k} \\ 0_{(n-k)\times k} \end{array}\right])+ \\ &rank(\left[\begin{array}{c} B_{k\times(|L|-k)} \\ M_{(n-k)\times(|L|-k)} \end{array}\right]) \end{aligned} \tag{22}$$

To prove that $B_{k\times(|L|-k)} = 0$, we re-write the reduced row echelon form of $A_{R,*}$ as follows:

$$\begin{aligned} rref(A_{R,*}) &= \left[\begin{array}{c|c} I_{k\times k} & B_{k\times(|L|-k)} \\ \hline 0_{(n-k)\times k} & M_{(n-k)\times(|L|-k)} \end{array}\right] \\ &= \left[\begin{array}{c|ccc} I_{k\times k} & b_1 & ... & b_{|L|-k} \\ \hline 0_{(n-k)\times k} & m_1 & ... & m_{|L|-k} \end{array}\right], \end{aligned} \tag{23}$$

where, b_i and m_i are the i-th column of $B_{k\times(|L|-k)}$ and $M_{(n-k)\times(|L|-k)}$ respectively. Let $[e_1, e_2, ..., e_k]$ be the columns of $\left[\begin{array}{c} I_{k\times k} \\ 0_{(n-k)\times(k)} \end{array}\right]$. Also let $[q_1, q_2, ..., q_{|L|-k}]$ be the columns of $\left[\begin{array}{c} B \\ M \end{array}\right]$, where $q_i = \left[\begin{array}{c} b_i \\ m_i \end{array}\right]$.
We define the indicator functions δ_i and δ'_i as follows:

$$\delta_i = \begin{cases} 1, & \text{if } \quad q_i \quad \text{is independent of} \\ & \{e_1, ..., e_k\} \cup \{q_1, ..., q_{i-1}\} \\ 0, & \text{Otherwise.} \end{cases}$$

$$\delta_i' = \begin{cases} 1, & \text{if} \quad q_i \quad \text{is independent of} \\ & \{q_1, ..., q_{i-1}\} \\ 0, & \text{Otherwise.} \end{cases}$$

Lemma 17 *We claim that*

$$\delta_i = \delta_i' \quad for i = 1, ..., |L| - k, \tag{24}$$

i.e., q_i is linearly independent of $\{e_1, ..., e_k\} \cup \{q_1, ..., q_{i-1}\}$, if q_i is linearly independent of $\{q_1, ..., q_{i-1}\}$.

To see this, we note that the left hand side (LHS) and right hand side of Equation (22) are as follows:

$$\text{LHS of (22):} \quad rank(rref(A_{R,*})) = k + \sum_{i=1}^{|L|-k} \delta_i \tag{25}$$

$$\text{RHS of (22):} \quad rank(rref(A_{R,*})) = k + \sum_{i=1}^{|L|-k} \delta_i' \tag{26}$$

It is clear that $\delta_i \leq \delta_i', \quad \forall i = 1, ..., |L| - k$, because if q_i is linearly independent of $\{q_1, ..., q_{i-1}\} \cup \{e_1, ..., e_k\}$ it has to be independent of $\{q_1, ..., q_{i-1}\}$ which is a subset of the former. Thus, if $\exists i \in \{1, ..., |L| - k\}$ such that $\delta_i < \delta_i'$, (25) will be smaller than (26), violating Equation (22). Thus, $\delta_i = \delta_i', \forall i = 1, ..., |L| - k$. Using the above claim, we prove $b_i = 0, \quad i = 1, ..., |L| - k$ by induction. For $i = 1$, if row $k+1$ in $rref(A_{R,*})$ contains a pivot in column $k+1$ (i.e. q_1 contains a pivot), then by definition of the reduced row echelon form, other entries in column $k+1$ should be zero and thus $b_1 = 0$. If row $k+1$ in $rref(A_{R,*})$ does not contain a pivot in column $k+1$ (i.e., not contain a pivot or contain a pivot in column $j > k+1$), then the non-zero entries (if any) in row $k+1$ and every row below row $k+1$ must be to the right of column $k+1$, i.e. $m_1 = 0$. Therefore, $q_1 = \begin{bmatrix} b_1 \\ m_1 \end{bmatrix}$ is linearly dependent with $\{e_1, ..., e_k\}$. By (24), $q_1 = 0$ and thus $b_1 = 0$.
For $i > 1$, assume $b_j = 0 \; for \; j = 1, ..., i-1$. If $q_i = \begin{bmatrix} b_i \\ m_i \end{bmatrix}$ contains a pivot, then the pivot must be in a row below row k (as (23) already indicates that the pivots in rows $1, .., k$ appear before column q_i). Thus by definition of reduced

row echelon form $b_i = 0$. If q_i does not contain a pivot, then q_i can be written as linear combination of $\{e_1, ..., e_k\}$ and $\{q_{i_l}\}_{l=1}^{i}$ where i_l is the index for those columns in $\{q_1, ..., q_i\}$ which contain a pivot. Thus, q_i is linearly dependent of $\{e_1, ..., e_k\} \cup \{q_1, ..., q_{i-1}\}$. By Lemma 17, q_i is linearly dependent of $\{q_1, ..., q_{i-1}\}$. Since $b_j = 0 \; for \; j = 1, ..., i-1$, b_i must be zero.

Therefore, using the reduced row echolen form of $A_{R,*}$ in (21), each link in L_1, corresponding to one of the first k columns in $rref(A_{R,*})$, can be uniquely determined from the set of selected paths P_R. We therefore, conclude that the necessary and sufficient condition for $P_R \subseteq P$ to identify a set of links L_1 is $rank(A_{R,*}) = |L_1| + rank(A_{R,L\setminus L_1})$.

Proof of Lemma 10 Suppose there exists an optimal solution of the LP-relaxation of Max-IL-Cost over Z_s and X_l, where $\exists l \in L$ with $0 < X_l < 1$. Therefore, $\exists s : l \in I(P_s)$ such that $Z_s > 0$. From (4.3-d), it implies that if $Z_s > 0$, we must have $Y_r = 1 \quad \forall r \in P_s$. Therefore, we can make $Z_s = 1$, and $X_l = 1$ to increase the value of the objective function without violating any constraint. This contradicts with the assumption that this solution is optimal. Similar argument shows a contradiction if $\exists s \in S \quad s.t. \quad 0 < Z_s < 1$. Therefore, the optimal solution of the LP relaxation over Z_s and X_l always gives an integer solution.

Proof of Lemma 13 A path set identifies all the identifiable links if and only if it satisfies the conditions of Theorem 9, i.e. $rank(A_{R,*}) = |L_I| + rank(A_{R,L\setminus L_I})$. Note that R is a solution to Min-Cost-Rank since $rank(A_{R,*}) = |L_I|$ Thus, the optimal solution to Min-Cost-IL is identical to the optimal solution to Min-Cost-Rank, given by Greedy-Min-Cost-Rank by theorem 12.

Proof of Theorem 14 The lower bound is obvious, since we showed that *Greedy-Min-Cost-Rank* returns the **optimal** minimum basis for A_{*,L_I}, there is no lower cost set of paths that is both a basis for A_{*,L_I} and satisfies the conditions of theorem 9. For the upper bound, note that any basis B_A for the routing matrix A identifies all links in L_I and thus has lower cost than K^{opt}.

Proof of Theorem 16 The lower bound $I_R \leq I^{opt}$ trivially holds due to the optimality of I^{opt}. For the upper bound, we denote by R^* the set of path indices in the optimal solution of Max-IL-Cost. Then by Theorem 9, $I^{opt} \leq rank(A_{R^*,*})$.

Meanwhile, by Theorem 15, we have that

$$rank(A_{R^*,*}) \leq rank^{opt} \leq rank(A_{R,*}) \cdot \frac{e}{e-1}, \tag{27}$$

where $rank^{opt}$ is the rank of the optimal solution of Max-Rank-Cost. This gives the upper bound on I^{opt}. Also, note that I^{opt} is always smaller than the maximum identifiability ($|L_I|$) using all possible paths in P.

Bibliography

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

[2] Bienstock, D. (2015) *Electrical Transmission System Cascades and Vulnerability: An Operations Research Viewpoint*, SIAM.

[3] Abraham et al., S. (2004) *Final report on the august 14, 2003 blackout in the united states and canada: Causes and recommendations*, US-Canada Power System Outage Task Force.

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

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

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

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

[8] Wang, J., C. Qiao, and H. Yu (2011) "On progressive network recovery after a major disruption," in *Proceedings IEEE INFOCOM*.

[9] Tootaghaj, D. Z., H. Khamfroush, N. Bartolini, S. Ciavarella, S. Hayes, and T. La Porta (2017) "Network Recovery from Massive Failures under Uncertain Knowledge of Damages," in *IFIP Proceedings of Networking (IFIP NETWORKING 2017)*.

[10] Tootaghaj, D. Z., N. Bartolini, H. Khamfroush, and T. La Porta (2017) "Network Recovery from Massive Failures under Uncertain Knowledge of Damages," in *IEEE Proceedings of the International Symposium on Reliable Distributed Systems (SRDS)*.

[11] Tootaghaj, D. Z., T. He, and T. La Porta (2017) "Parsimonious Tomography: Optimizing Cost-Identifiability Trade-off for Probing-based Network Monitoring," in *IFIP Proceedings of Performance (IFIP Performance 2017)*.

[12] Knabb, R. D., J. R. Rhome, and D. P. Brown (2006) "Tropical Cyclone Report: Hurricane Katrina, August 23-30, 2005," *Fire Engineering*, pp. 32–40.

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

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

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

[16] Ciavarella, S., N. Bartolini, H. Khamfroush, and T. La Porta (2017) "Progressive Damage Assessment and Network Recovery after Massive Failures," in *INFOCOM*, IEEE.

[17] Bent, R. and P. Van Hentenryck (2005) "Online Stochastic Optimization Without Distributions." in *ICAPS*.

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

[19] Neumayer, S. and E. Modiano (2010) "Network reliability with geographically correlated failures," in *Proceedings IEEE INFOCOM*.

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

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

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

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

[24] Hu, T. C. (1963) "Multi-commodity network flows," *Operations research*.

[25] "Gurobi Optimization, Inc. Gurobi optimizer reference manual," `http://www.gurobi.com/`, accessed in, 2012.

[26] Ciavarella, S., N. Bartolini, H. Khamfroush, and T. La Porta (2017) "Progressive damage assessment and network recovery after massive failures," in *INFOCOM*, IEEE.

[27] Parandehgheibi, M., E. Modiano, and D. Hay (2014) "Mitigating cascading failures in interdependent power grids and communication networks," in *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, IEEE, pp. 242–247.

[28] Rosato, V., L. Issacharoff, F. Tiriticco, S. Meloni, S. Porcellinis, and R. Setola (2008) "Modelling interdependent infrastructures using interacting dynamical models," *International Journal of Critical Infrastructures*.

[29] Rosato, V., L. Issacharoff, G. Gianese, and S. Bologna (2009) "Influence of the topology on the power flux of the Italian high-voltage electrical network," *arXiv preprint arXiv:0909.1664*.

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

[31] Brummitt, C. D., R. M. D'Souza, and E. Leicht (2012) "Suppressing cascades of load in interdependent networks," *Proceedings of the National Academy of Sciences*, **109**(12), pp. E680–E689.

[32] Sturaro, A., S. Silvestri, M. Conti, and S. K. Das (2016) "Towards a realistic model for failure propagation in interdependent networks," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, pp. 1–7.

[33] Parshani, R., S. V. Buldyrev, and S. Havlin (2010) "Interdependent networks: reducing the coupling strength leads to a change from a first to second order percolation transition," *Physical review letters*, **105**(4), p. 048701.

[34] Buldyrev, S. V., R. Parshani, G. Paul, H. E. Stanley, and S. Havlin (2010) "Catastrophic cascade of failures in interdependent networks," *Nature*, **464**(7291), pp. 1025–1028.

[35] Khamfroush, H., N. Bartolini, T. La Porta, A. Swami, and J. Dillman (2016) "On Propagation of Phenomena in Interdependent Networks," *IEEE Transactions on network science and engineering*.

[36] Shekhtman, L. M., M. M. Danziger, and S. Havlin (2016) "Recent advances on failure and recovery in networks of networks," *Chaos, Solitons & Fractals*.

[37] Chatziafratis, E., Y. Zhang, and O. Yagan (2016) "On the robustness of power systems: optimal load-capacity distributions and hardness of attacking," in *Information Theory and Applications Workshop (ITA)*.

[38] Bernstein, A., D. Bienstock, D. Hay, M. Uzunoglu, and G. Zussman (2014) "Power grid vulnerability to geographically correlated failuresâĂŤanalysis and control implications," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, IEEE, pp. 2634–2642.

[39] Tootaghaj, D. Z., F. Farhat, M. R. Pakravan, and M. R. Aref (2011) "Game-theoretic approach to mitigate packet dropping in wireless Ad-hoc networks," in *Consumer Communications and Networking Conference (CCNC)*, IEEE.

[40] ——— (2011) "Risk of attack coefficient effect on availability of Ad-hoc networks," in *Consumer Communications and Networking Conference (CCNC)*, IEEE.

[41] Das, A., J. Banerjee, and A. Sen (2014) "Root cause analysis of failures in interdependent power-communication networks," in *2014 IEEE Military Communications Conference*, IEEE, pp. 910–915.

[42] Das, A., C. Zhou, J. Banerjee, A. Sen, and L. Greenwald (2015) "On the smallest pseudo target set identification problem for targeted attack on interdependent power-communication networks," in *Military Communications Conference, MILCOM 2015-2015 IEEE*, IEEE, pp. 1015–1020.

[43] Carreras, B. A., V. E. Lynch, I. Dobson, and D. E. Newman (2002) "Critical points and transitions in an electric power transmission model for cascading failure blackouts," *Chaos: An interdisciplinary journal of nonlinear science*.

[44] Eppstein, M. J. and P. D. Hines (2012) "A random chemistry algorithm for identifying collections of multiple contingencies that initiate cascading failure," *IEEE Transactions on Power Systems*.

[45] Kayastha, N., D. Niyato, E. Hossain, and Z. Han (2014) "Smart grid sensor data collection, communication, and networking: a tutorial," *Wireless communications and mobile computing*.

[46] Pendarakis, D., N. Shrivastava, Z. Liu, and R. Ambrosio (2007) "Information aggregation and optimized actuation in sensor networks: enabling smart electrical grids," in *INFOCOM*, IEEE.

[47] Sridhar, S., A. Hahn, and M. Govindarasu (2012) "Cyber–physical system security for the electric power grid," *Proceedings of the IEEE*.

[48] Soltan, S., M. Yannakakis, and G. Zussman (2015) "Joint cyber and physical attacks on power grids: Graph theoretical approaches for information recovery," in *SIGMETRICS*, ACM.

[49] Dicorato, M., A. Minoia, R. Sbrizzai, and M. Trovato (2002) "A simulation tool for studying the day-ahead energy market: the case of Italy," in *Power Engineering Society Winter Meeting, 2002. IEEE*, IEEE.

[50] Adams, A., T. Bu, T. Friedman, J. Horowitz, D. Towsley, R. Caceres, N. Duffield, F. L. Presti, S. B. Moon, and V. Paxson (2000) "The use of end-to-end multicast measurements for characterizing internal network behavior," *IEEE Communications magazine*.

[51] Vardi, Y. (1996) "Network tomography: Estimating source-destination traffic intensities from link data," *Journal of the American statistical association*.

[52] Stallings, W. (1998) "SNMP and SNMPv2: the infrastructure for network management," *IEEE Communications Magazine*.

[53] Andersen, D., H. Balakrishnan, F. Kaashoek, and R. Morris (2001) *Resilient overlay networks*, ACM.

[54] Feamster, N., D. G. Andersen, H. Balakrishnan, and M. F. Kaashoek (2003) "Measuring the effects of Internet path faults on reactive routing," in *ACM SIGMETRICS Performance Evaluation Review*, ACM.

[55] Paxson, V. (1997) "End-to-end routing behavior in the Internet," *IEEE/ACM transactions on Networking*.

[56] Luckie, M., A. Dhamdhere, D. Clark, B. Huffaker, et al. (2014) "Challenges in inferring internet interdomain congestion," in *IMC*, ACM.

[57] Khuller, S., A. Moss, and J. S. Naor (1999) "The budgeted maximum coverage problem," *Information Processing Letters*.

[58] Hochbaum, D. S. (1982) "Approximation algorithms for the set covering and vertex cover problems," *SIAM Journal on computing*.

[59] Bejerano, Y. and R. Rastogi (2006) "Robust monitoring of link delays and faults in IP networks," *IEEE/ACM Transactions on Networking (TON)*.

[60] Kumar, R. and J. Kaur (2006) "Practical beacon placement for link monitoring using network tomography," *IEEE Journal on Selected Areas in Communications*.

[61] Gopalan, A. and S. Ramasubramanian (2012) "On identifying additive link metrics using linearly independent cycles and paths," *IEEE/ACM Transactions on Networking (TON)*.

[62] Ma, L., T. He, K. K. Leung, A. Swami, and D. Towsley (2014) "Inferring link metrics from end-to-end path measurements: Identifiability and monitor placement," *IEEE/ACM Transactions on Networking (TON)*.

[63] Ma, L., T. He, K. K. Leung, D. Towsley, and A. Swami (2013) "Efficient identification of additive link metrics via network tomography," in *IEEE ICDCS*.

[64] Chen, Y., D. Bindel, H. Song, and R. H. Katz (2004) "An algebraic approach to practical and scalable overlay network monitoring," in *ACM SIGCOMM Computer Communication Review*, vol. 34, ACM, pp. 55–66.

[65] Li, F. and M. Thottan (2006) "End-to-End Service Quality Measurement Using Source-Routed Probes." in *INFOCOM*.

[66] Zheng, Q. and G. Cao (2013) "Minimizing probing cost and achieving identifiability in probe-based network link monitoring," *IEEE Transactions on Computers*.

[67] Fujishige, S. (2005) *Submodular functions and optimization*, Elsevier.

[68] Oxley, J. G. (2006) *Matroid theory*, vol. 3, Oxford University Press, USA.

[69] Murota, K. (2009) *Matrices and matroids for systems analysis*, vol. 20, Springer Science & Business Media.

[70] He, T., N. Bartolini, H. Khamfroush, I. Kim, L. Ma, and T. La Porta (2016) "Service placement for detecting and localizing failures using end-to-end observations," in *IEEE ICDCS*.

[71] Kruskal, J. B. (1956) "On the shortest spanning subtree of a graph and the traveling salesman problem," *Proceedings of the American Mathematical society*.

[72] Strang, G. (2011) "Introduction to linear algebra," .

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

[74] Nunes, B. A. A., M. Mendonca, X. N. Nguyen, K. Obraczka, and T. Turletti (2014) "A survey of software-defined networking: Past, present, and future of programmable networks," *IEEE Communications Surveys & Tutorials*.

[75] Yeganeh, S. H., A. Tootoonchian, and Y. Ganjali (2013) "On scalability of software-defined networking," *IEEE Communications Magazine*.

[76] McKeown, N., T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner (2008) "OpenFlow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*.

[77] "OpenFlow Switch Specification." `https://www.opennetworking.org/images/stories/downloads/specification/openflow-spec-v1.3.1.pdf`.

[78] Kuźniar, M., P. Perešíni, and D. Kostić (2015) "What you need to know about sdn flow tables," in *International Conference on Passive and Active Network Measurement*, Springer.

[79] Reitblatt, M., N. Foster, J. Rexford, C. Schlesinger, and D. Walker (2012) "Abstractions for network update," in *ACM SIGCOMM*, ACM.

[80] Vissicchio, S., L. Vanbever, L. Cittadini, G. Xie, O. Bonaventure, et al. (2013) "Safe updates of hybrid SDN networks," *Université catholique de Louvain, Tech. Rep.*

[81] Wen, X., B. Yang, Y. Chen, L. E. Li, K. Bu, P. Zheng, Y. Yang, and C. Hu (2016) "Ruletris: Minimizing rule update latency for tcam-based sdn switches," in *ICDCS*, IEEE.

[82] Kettner, A. M. and M. Paolone (2017) "On the Properties of the Power Systems Nodal Admittance Matrix," *arXiv preprint arXiv:1702.07235*.

Vita

Diman Zad Tootaghaj

Website: http://www.cse.psu.edu/ dxz149/home.html
Email: diman.zt@gmail.com

Education:

- PhD Candidate, School of Electrical Engineering and Computer Science,
  The Pennsylvania State University,
  The Institute for Networking and Security Research (INSR),
  Adviser: Professor Thomas La Porta,
  Co-Adviser: Professor Ting He,
  Thesis Title: Modeling, Monitoring and Scheduling techniques for Network Recovery from Massive Failures.

- MS, Computer Science and Engineering,
  The Pennsylvania State University,
  Microsystems Design Lab (MDL),
  Thesis Title: Evaluating Cloud Workload Characteristics.

- MS, Electrical Engineering,
  Sharif University of Technology, Iran,
  Intelligent Systems Research Lab,
  Advisor: Professor Mohammad Reza Aref,
  Co-Adviser: Professor Mohammad Reza Pakravan,
  Thesis Title: Analysis of Routing Misbehavior in Ad-Hoc Networks.

- BS, Electrical Engineering,
  Sharif University of Technology, Iran,
  Advisor: Professor Javad Salehi,
  Thesis Title: Analysis of Dynamic Bandwidth Allocation Algorithms in Fiber to the Home.

Experience:

- Microsoft Research Internship: Working on Uncertain<T> probabilistic programming language for gesture recognition in Windows Mobile applications, Under Supervision of Kathryn S. McKinley and Todd Mytkowicz, 2015.

- Instructor: CMPSC 473, Operating Systems, The Pennsylvania State University, Fall 2015.

- Research and Teaching Assistant, The Pennsylvania State University.