The Pennsylvania State University
The Graduate School
College of Engineering

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

A Dissertation in
Computer Science and Engineering
by
Diman Zad Tootaghaj

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2018

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

*Signatures are on file in the Graduate School.

Abstract

This dissertation 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. To this end, I focus on network recovery assuming partial and uncertain knowledge of the failure locations.

I first studied large-scale failures in a communication network. In particular, I proposed a new recovery approach under uncertain knowledge of failures. I proposed 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, I selected a node with highest centrality at each iteration step to repair and exploit as a monitor to increase the knowledge of network state, until all critical services are restored. The recovery problem can be addressed by giving different priority to three performance aspects including: 1) Demand loss, 2) computation time and 3) number of repairs (or repair cost). These aspects are in conflict with each other and I studied the trade-off among them.

Next, I focused on failure recovery of multiple interconnected networks. In particular, I focused on the interaction between a power grid and a communication network. I modeled the cascading failures in a power gird using a DC power flow model. I tackled the problem of mitigating an ongoing cascade by formulating the minimum cost flow assignment problem as a linear programming optimization. The optimization aimed 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 unsatisfied load due to the re-distribution of the power in the generators and loads without violating the overload constraint at each line.

Then, I focused on network monitoring techniques that can be used for diagnosing the performance of individual links for localizing *soft failures* (e.g. highly congested links) in a communication network. I studied the optimal selection of the monitoring paths to balance identifiability and probing cost. I 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. I showed 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). I proposed an optimal greedy-based approach for (4) and proposed a $(1 - 1/e)$-approximation algorithm for (2). My experimental analysis revealed that, compared to several greedy approaches that directly solve the identifiability-based optimization (i.e. (1) and (3)), the proposed rank-based optimization (i.e. (2) and (4)) achieved better trade-offs in terms of identifiability and probing cost.

Finally, I addressed, a minimum disruptive routing framework in software defined networks. I showed that flow disruption, congestion and violation of policies can occur during updates of flow tables in software defined networks. I aimed to minimize the update disruption and minimize the number of affected flows during the update, while taking into account link capacity constraints and the importance of various flows to upper-layer applications. I formulated the problem as an integer linear programming and showed that it is NP-Hard. I proposed two randomized rounding algorithms with bounded congestion and demand loss to solve this problem. In addition to a small SDN testbed, I performed a large-scale simulation study to evaluate my proposed approaches on real network topologies. Extensive experimental and simulation results show that the two random rounding approaches have a disruption cost close to the optimal while incurring a low congestion factor and a low demand loss.

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 dissertation 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.

I wish to express my deepest gratitude to my parents, Soraya and Ebrahim, for their endless love and support throughout my life. I would also like to thank my parent-in-laws, Ashraf and Mohammad, for their constant kindness and support. I am also grateful to my brothers, Saman and Siamand, and my sisters-in-law, Mahshid, Nasim and Sima who have always inspired me to pursue my goals and supported me during ups and downs of my life.

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 under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA) and I would like to express my gratitude to them.

Dedication

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

Chapter 1

Introduction

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 [2]. 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 [3,4], 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 dissertation, I 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. I then focus on network monitoring techniques that can be used for diagnosing individual links' performance or localizing the failures. In Chapter 4, I study the optimal selection of the monitoring paths to balance identifiability and cost. In Chapter 5 I study a minimum disruptive way of updating on flow rules for software defined networks.

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, I 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 [1, 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, I propose an itrative stochastic recovery approach (ISR) in Chapter 2, that runs a

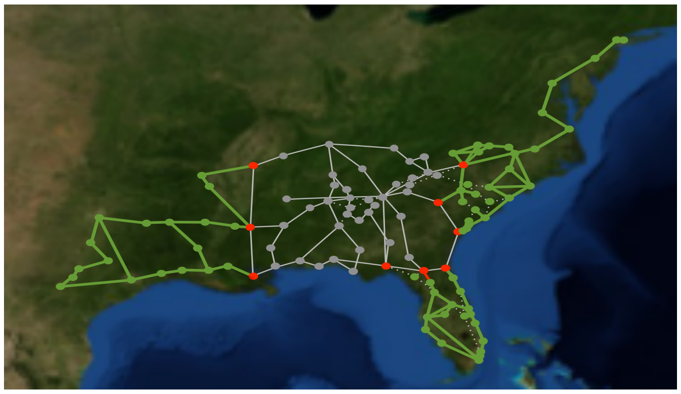


Figure 1.1: Large-scale failure in Deltacom topology.

multi-stage stochastic optimization algorithm. At each iteration step, ISR finds 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

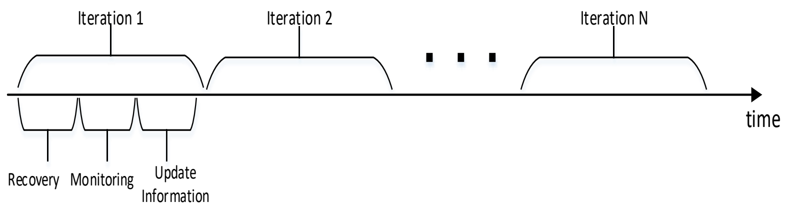


Figure 1.2: Different steps of the proposed progressive recovery approach.

systems, financial transactions and power grids, transportation systems and food 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 my knowledge, the proposed progressive recovery approach is the first work that studies progressive recovery of a disrupted network under uncertainty. Figure 2.1 shows different steps of the proposed recovery approach. At each iteration step, based on the available resources, the proposed algorithm repairs some of the damaged network elements, performs a monitoring step and gains more information and iterates this procedure until all critical services are restored.

1.2 My Contribution

This dissertation aims 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, I 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. I 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, I tackle for the first time, the problem of network recovery after massive disruption under uncertainty of the exact location of the disrupted nodes/links. I 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. I propose several algorithms to find a feasible solution set at each iteration of the algorithm. Experimental results show that ISR outperforms the state-of-the-art ISP algorithm while having a configurable 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, I show that the inter-connectivity and dependency between different elements makes complex networks more vulnerable to failure [10, 11]. I study the inter-dependency between a power grid and a communication network. I 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. I 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. The recovery phase aims at maximizing the restored accumulative flow. I show that the recovery problem (*Max-R*) is NP-Hard and propose heuristic recovery strategies that work under partial knowledge of damage locations. I 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, I study the optimal selection of monitoring paths to balance identifiablity and cost [12, 13]. To this end, I 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. I 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). I proposed an optimal greedy-based approach for (4) and proposed a $(1 - 1/e)$-approximation algorithm for (2). Experimental analysis reveals that, compared to several greedy approaches, my rank-based optimization performs better in terms of identifiability and probing cost.

1.2.4 A Minimally Disruptive Rule Update in Software Defined Networking

Rule forwarding updates occur very frequently in SDN networks due to arrival of new traffic, topology changes and network re-configurations; they are an inseparable part of network management and traffic engineering systems. Service disruption and inconsistencies can occur during the updates leading to degraded quality of service or disconnectivity of the exisiting services. In Chapter 5, I study network reconfiguration techniques in software defined networks. I introduce a minimally disruptive rule update problem (*Min-touch*) and show that it is NP-Hard. I propose two randomized rounding algorithms with bounded approximation factors on congestion and demand loss.

1.3 Organization

The remainder of the dissertation is organized as follows. Chapter 2 proposes a 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]. I 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. I then formulate a recovery plan to maximize the total amount of power delivered to the demand loads during the recovery intervention. In Chapter 4, I study the optimal selection of monitoring paths to balance identifiability and cost [12]. Chapter 5 studies network reconfiguration with minimum disruption to the existing flows. I propose two randomized rounding algorithms with bounded approximation factors on congestion and demand loss. Finally, I conclude the dissertation and discuss the future work in Chapter 6.

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 [14]. The storm led to the loss of service over 2.5 million lines on the BellSouth (now AT&T) network [2]. 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, I focus on network recovery assuming partial and uncertain knowledge of the failure pattern.

In this Chapter, I 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

Large-scale failures in communication networks due to natural disasters or malicious attacks can severely affect critical communications and threaten lives of people in that area. In 2005, Hurricane Katrina led to an outage of over 2.5 million lines in the BellSouth (now AT&T) network [2]. In 2017, more than 7 million subscribers to cable or wire-line telecommunication services lost service due to Hurricane Irma [15]. 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, I focus on network recovery assuming partial and uncertain knowledge of the failure pattern.

I propose a multi-stage stochastic recovery algorithm, that uses three optimization techniques to repair a part of the network at each iteration until critical services are restored. To clarify the discussion, I consider different states of network components. Depending on the available knowledge, I 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. I improve the knowledge of the network state by installing monitors on top of the nodes repaired at each iteration. A monitor is a piece of software, which can be installed on a working node to discover the reachable nodes. Monitor nodes provide additional information about the status of the network, which can be used to revise and improve the recovery plan. The **contributions** of this work are the following:

- I tackle for the first time the problem of network recovery after massive disruption under uncertainty of the exact location of the disrupted nodes/links.
- I 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.

- I propose a multi-stage iterative stochastic recovery (ISR) algorithm, that is presented in three different versions (depending on the optimization algorithm that is used), namely, *Iterative shortest path* (ISR-SRT), *Iterative Branch and Bound* (ISR-BB), and *iterative multi-commodity LP relaxation* (ISR-MULT) to find a feasible solution and solve the MINER problem.

- In order to provide a fair comparison with my approach, I 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. I 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 my algorithms.

- Further, I compare my algorithms with the state-of-the-art *Centrality based Damage Assessment and Recovery* (CeDAR) algorithm [16]. CeDAR works under incomplete knowledge of failure but aims at maximizing the total satisfied flow during the recovery process, while I aim at minimizing the recovery cost. Further, CeDAR does not use a probabilistic knowledge of failures. Experimental results show that ISR-BB and ISR-MULT outperform the state-of-the-art ISP and CeDAR algorithms in terms of recovery cost.

I observed that since the algorithms in [6] have been designed for known network failure patterns, poor decisions in the first iterations of the algorithm propagate through the entire execution, while my algorithm can correct previous decisions after each iteration as more information becomes available and therefore, reduces the recovery cost. Different configurations of my algorithm can significantly improve the total number of repairs over other heuristics while performing close to the optimal in terms of recovery cost.

The remainder of this chapter is organized as follows. Section 2.2 discusses the background and motivation behind this work. In Section 2.3, I explain the minimum expected recovery (*MINER*) problem and show it is NP-Hard. Section 2.4 describes my approach to minimize the expected recovery cost. Section 2.6 shows my evaluation methodology and experimental results and Section 2.7 concludes the chapter with a summary.

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, 17–24]. To the best of my knowledge, network recovery has not been extensively studied under uncertainty.

In the absence of complete knowledge of disrupted network components, prior works propose network tomogrpahy techniques to localize node failures from binary states of end-to-end paths, infer the performance degradation of links or identifying system performance from large noisy data sets [25–32]. However, binary tomography approaches are limited to localizing sparse failures and cannot be applied to large-scale disruptions.

In a different line of research the problem of network recovery has been studied in the case of interdependent networks [33–35]. Dependent failures in interdependent networks between a power grid and a communication network has been studied in [34]. Tootaghaj et al. propose a progressive recovery approach in an inter-dependent network consisting of a power grid and a communication network [33]. They assume a limited amount of resources available at each iteration step and propose a progressive recovery heuristic that restores the power lines while maximizing the total load served during the recovery intervention.

Wang et al. and Ciavarella et al. studied progressive network recovery for large-scale failures. They proposed a progressive recovery approach to maximize the weighted sum of total flow over the entire steps of recovery [8, 16]. While both Wang et al. and Ciavarella et al.'s work and my work aim to design a progressive recovery approach, the objective is different. In [8] and [16], the objective is maximizing the throughput over time, whereas I 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 my work and [16] do not make this assumption. I assume availability of a probabilistic estimate of the failure scenario, while [16] assumes lack of knowledge.

The problem of minimizing the recovery cost to satisfy multiple demand flows under network capacity or quality of service constraint has been proven to be NP-hard and several heuristics have been proposed in the literature to reduce the complexity. Bartolini et al. propose a polynomial-time heuristic, called ISP, to break the problem into smaller 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.

I propose a progressive version of ISP in Section 2.5.1 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, I show that by running my multi-stage recovery approach, the total number of repairs is reduced compared to ISP and unnecessary repairs are avoided. Furthermore, I 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.

I 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, my 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, I show the gap between optimal recovery and ISP [6] when I 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 (gray), meaning that their state is only known with some probability. If I use an algorithm like ISP to determine

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

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 that once the algorithm is run, all repairs are determined. Therefore, it does not have an opportunity to learn the state of the uncertain nodes. I performed a set of simulations on three different network topologies to illustrate the gap in performance between ISP run with full knowledge vs. uncertain knowledge. I 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, I assume the whole network is gray, 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.

I then modified ISP, as described in Section 2.5.1, to run in iterations, where in each iteration a repair is made. Information about network state is then gathered for any new portions of the network now visible due to the repair, and the algorithm is run again with the new information until all demands are met. I 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 me to develop my iterative stochastic recovery (ISR) algorithm that progressively repairs network elements and updates its knowledge of the state of the network.

Table 2.2: Summary of notations.

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

2.3 Problem Definition

I 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, I 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, I assume knowledge of the probability of failure of each node/link in the gray area, which can be found

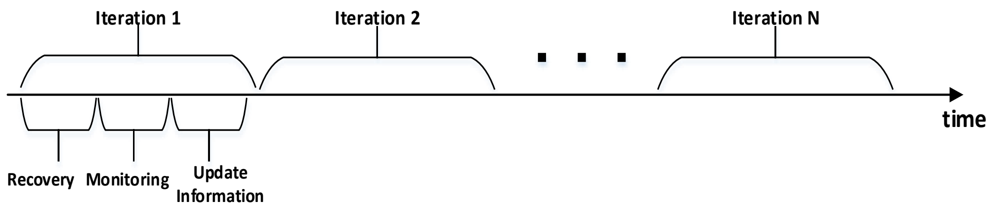


Figure 2.1: Different steps of my progressive recovery approach.

using machine learning algorithms [36, 37], seismography analysis in case of an earthquake [38], or understanding the robustness of different parts of the network. The network elements in the gray area might have geographical or independent correlation of failures [39]. For example, if I am 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. I model the intensity of the disruption according to a geographic failure distribution. This may be a bi-variate Gaussian function, whose variance determines the extent of the disruption around the epicenter of a disruptive event, or any other geographical distribution. In the absence of information on the probabilistic model of the disruption, I consider a uniform distribution of failures. I am interested in gradual recovery of the network such that the total cost of repaired nodes/edges during all steps of the recovery is minimized.

Figure 2.1 shows different steps of the proposed recovery approach. At each iteration step, based on the current knowledge of the network state, I 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.

2.3.1 Network Model

Given an undirected graph $G = (V, E)$ and a set of demand pairs $E_H = \{(s_1, t_1), ..., (s_k, t_k)\}$, where $E_H \subseteq V \times V$ and each demand pair $(s_h, t_h) \in E_H$ has a source s_h, a destination t_h and a positive demand flow d_h, the goal is to minimize the expected recovery cost (ERC) to satisfy the demands while having capacity constraint c_{ij} for every edge in the graph. Table 5.1 shows the notation used in this chapter.

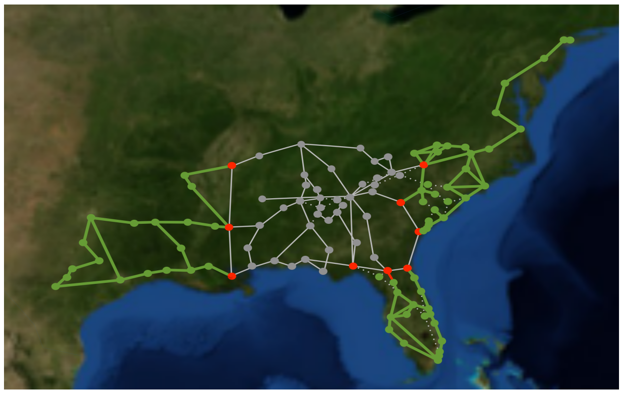


Figure 2.2: An example of a failure in a real network topology from the internet topology zoo [1].

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 I know for sure have failed,

2. the sets $E_U \subseteq E$ and $V_U \subseteq V$ are the sets of edges and nodes in the gray area whose failure patterns is **unknown**,

3. the sets $E_W \subseteq E$ and $V_W \subseteq V$ are the sets of nodes and edges in the green area which are known to be **working** correctly in the system.

To clarify the discussion, consider the network shown in Figure 2.2. Assume there exists a massive failure in the gray area in Figure 2.2. At the beginning of the recovery process, I do not have complete knowledge about the failures and the network is partitioned into a green, red and gray area. It is possible that the nodes in the middle of the gray area have not failed and therefore, recovery of a few nodes in the gray area may lead the whole graph of the network to be connected.

At the beginning of the iterative recovery process, I 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 I repair more nodes/edges in the network, I exploit the repaired nodes as monitors to discover the gray area and adjust the initial estimate $\zeta(0)$ about the failure probability distribution.

2.3.2 Recovery Problem

To model the optimization problem as a decision making process which includes uncertainty, I model the cost of repair as a function of the failure probability for each node/edge in the network. My estimate of the probability of failure in the location of the considered network elements at the n^{th} iteration is $\zeta(n) = \{\zeta^e_{ij}(n) \quad \forall e_{ij} \in E, \zeta^v_i(n) \quad \forall v_i \in V\}$, where $\zeta^v_i(n)$ and $\zeta^e_{ij}(n)$ are representing my estimate about the failure probability of node v_i and edge e_{ij} in the network at the n^{th} iteration. I use heterogeneous non-uniform cost function in my evaluation, where k^v_i and k^e_{ij} is the cost of repairing each vertex v_i and edge e_{ij} in the network. Therefore, the objective function is to minimize the expected recovery cost (ERC) given the information from the monitoring nodes to satisfy the given demand.

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

$$\text{minimize} \sum_{(i,j)\in E_U \cup E_B} k^e_{ij} \cdot \zeta^e_{ij}(n)\delta^e_{ij}(n) + \sum_{i \in V_U \cup V_B} k^v_i \cdot \zeta^v_i(n)\delta^v_i(n) \tag{2.1a}$$

$$\text{subject to } c_{ij}.\delta^e_{ij}(n) \geqslant \sum_{h=1}^{|E_H|} f^h_{ij}(n) + f^h_{ji}(n), \qquad \forall (i,j) \in E \tag{2.1b}$$

$$\delta^v_i(n) \cdot \eta_{max} \geqslant \sum\nolimits_{(i,j)\in E_B} \delta^e_{ij}(n), \qquad \forall i \in V \tag{2.1c}$$

$$\sum\nolimits_{j \in V} f^h_{ij}(n) = \sum\nolimits_{k \in V} f^h_{ki}(n) + b^h_i(n), \qquad \forall (i,h) \in V \times E_H \tag{2.1d}$$

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

$$\delta^v_i(n), \delta^e_{i,j}(n) \in \{0,1\}, \quad \forall (i,j) \in E, \quad \forall i \in V \tag{2.1f}$$

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

Constraint 6.1a specifies that the fraction of flow that will be routed through link (i, j) has to be smaller than or equal to the capacity of that edge; constraint 6.1b specifies that the degree of each node is smaller than or equal to the maximum degree of the network; 6.1c 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; 6.1d states that I consider non-negative assignment of flows and finally 6.1e shows the binary variables representing the decisions to use nodes and edges at iteration n. My goal here is to minimize the expected recovery cost. Since my initial estimate about the failure in the system is not always correct, it may happen that I 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, I distinguish between necessary and unnecessary recovery interventions. I 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, I 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 I decide to use, $\delta^e_{ij}, \delta^v_i = 1$, and are found to be properly functional after a local inspection. On the other hand, necessary repairs are the total number of nodes/edges in the gray or red area, $n_i \in \{V_U \cup V_B\}, e_{i,j} \in \{E_U \cup E_B\}$ which I decide to use, $\delta^e_{ij}, \delta^v_i = 1$ and are found to be broken.

Theorem 1 *The problem MINER is NP-Hard*

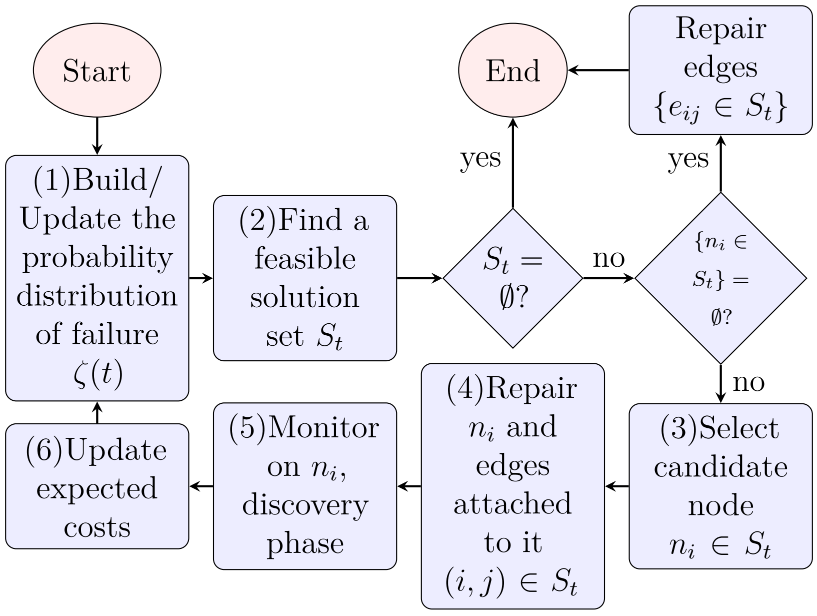
Proof of theorem 1 The Steiner Forest problem which is NP-Hard and APX-Hard in general graphs [133–135], is a special case of my optimization problem. To reduce the Steiner Forest problem to an instance of MINER problem, I create one unit of demand flow for each demand pair in the supply graph. I assume all source/destination pairs in the supply graph are the set of node pairs in the Steiner Forest problem for which I want to find a forest with minimum cost. Furthermore, I assume there exist no broken/gray nodes in the supply graph and all edges are broken with expected repair cost of $k^e_{ij}\zeta^e_{ij}$. I also assume that the capacity of edges is large enough to accommodate the sum of all demand flows. Since there exists no broken node, this instance of MINER returns a set of edges to recover, and since the capacity of links are large enough to accommodate the sum of all flows, a single path between any source/destination pair suffices to accommodate the demand. Therefore, the union of repaired edges generates a Steiner forest because any cycle implies unnecessary repairs. Also, since MINER minimized the repair cost, the forest is the one with minimum cost. Therefore, MINER problem is also NP-Hard.

2.4 Iterative Stochastic Recovery Algorithms

In this section, I propose the *Iterative Stochastic Recovery* (ISR) algorithm, in its three variants, namely, Iterative shortest path (ISR-SRT), Iterative branch and bound (ISR-BB), and iterative multi-commodity (ISR-MULT). The skeleton of these versions follow the same structure and only differ in terms of the approximate algorithm they use. I summarize ISR algorithm in six main steps shown in Figure 2.3 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, $S_t = \{(i \in V_U \cup V_B \quad | \delta_i = 1), ((i,j) \in E_U \cup E_B \quad | \delta_{ij} = 1)\}$ (Step 2). In my evaluation, I do not consider infeasible problems, i.e., there exists at least one feasible solution which can satisfy all critical services.

I 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 I 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). I repair the candidate node, and use it to monitor (Step 5) the surrounding network and obtain more information about the status of the network. In step 6, the algorithm updates the previous estimate of the costs after the discovery. The procedure is repeated until all the demands are satisfied or no more repairs are possible although there is a demand loss.



Start
End
Repair edges $\{e_{ij} \in S_t\}$
yes
yes
(1)Build/ Update the probability distribution of failure $\zeta(t)$
(2)Find a feasible solution set S_t
$S_t = \emptyset?$
no
$\{n_i \in S_t\} = \emptyset?$
no
(4)Repair n_i and edges attached to it $(i,j) \in S_t$
(5)Monitor on n_i, discovery phase
(6)Update expected costs
(3)Select candidate node $n_i \in S_t$

Figure 2.3: Different steps of the proposed iterative stochastic recovery (ISR).

2.4.1 An Approximate Feasible Solution

This section describes the three different proposed approaches to find an approximate feasible solution, step (2) in Figure 2.3, of the MINER problem. I use this approximate solution set (S_t) to select a candidate node to repair and gain information in the 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, I 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

```
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
1  DemandSatisfied= False;
2  t= 0;
3  $Solution = \emptyset$ ;
4  while DemandSatisfied !=True do
5      Find an approximate solution set of nodes/edges to repair from the
       $MINER$ problem that satisfy the demand:
       $S_t = \{V_{s(t)} \in (V_B \cup V_U), E_{s(t)} \in (E_B \cup E_U)\}$ using ISR-SRT, ISR-BB, or
       ISR-MULT.;
6      if $S_t == \emptyset$ then
7          DemandSatisfied = True;
8          break;
9      else
10         SelectedNode = Select a node with highest flow in the current solution
           $S_t$ ;
11         if $|SelectedNode| > 1$ then
12             SelectedNode = Select the node with maximum failure probability ;
13         Repair the SelectedNode, $n_i$ and edges attached to it, $e_{n_ij} \in S_t$ ;
14         $Solution = Solution \cup n_i \cup e_{n_ij \in S_t}$ ;
15         Put a monitor on the selected node and run $m$-hop discovery phase;
16         $t = t + 1$ ;
17         Update the estimated belief $\zeta(t)$ from failure probability distribution
           from the discovered nodes/edges ;
18 return Solution
```

each demand separately, without considering potential conflict among them. To account for the impact of uncertainty, I use a new notion of path length. For a path at the n^{th} iteration, the length of each link $e_{ij} \in E$ is defined as $l^{(n)}(e_{ij}) = k^e_{ij} \cdot \zeta^e_{ij}(n) + (k^v_i \cdot \zeta^v_i(n) + k^v_j \cdot \zeta^v_j(n))/2$, where $k^e_{ij} \cdot \zeta^e_{ij}(n)$, $k^v_i \cdot \zeta^v_i(n)$ and $k^v_j \cdot \zeta^v_j(n)$ are the expected cost of repair for edge e_{ij} and nodes i and j based on the estimated probability distribution at the n^{th} iteration. Therefore, the algorithm finds the shortest expected repair cost paths for each demand pair to repair independently. I 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, I use an iterative branch and bound optimization [40]. 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 [41]. 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 I have the optimal solution, otherwise, I pick a fractional variable, δ_x, and make two branches by creating two more constraints in the optimization ($\delta_x = 0$ and $\delta_x = 1$). I continue this procedure by making more branches to get closer to optimal. The smallest branch that satisfies all integrality constraints is called an *incumbent*. I 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 I 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 I have a guarantee on the execution time of the algorithm. In the worst-case scenario, I need to put all fractional variables from the LP-relaxation of MINER in the solution set. At each iteration, I 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, I 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, I propose a new iterative multicommodity solution. In this algorithm, I do not explore all possible branches, but only select the branch which is more likely to stay in the final solution (best candidate node selection). I first start by constructing a linear programming (LP) relaxation of the *MINER* problem which can be solved in polynomial time providing non-integer solution for $0 \leq \delta_i \leq 1$ and $0 \leq \delta_{i,j} \leq 1$. The LP relaxation gives a lower bound on the objective function of MINER, but it can result in many repairs if I repair all fractional variables. To reduce the number of repairs, I 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. I 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 I only select one of the branches and do not explore other possible branches. At each iteration of the algorithm, I repair the node or the edge which contributes the maximum flow. In case of ties, I choose the network element with maximum failure probability.

2.4.2 Best Candidate Node Selection

The choice of the best candidate node, step (3) in Figure 2.3, is performed based on a centrality ranking, where I 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. I also tried a different criterion for selecting the best candidate node. In particular I tried to select the node with maximum failure probability. Extensive simulation analysis shows that in most scenarios, selecting the node with maximum centrality gives better results and reduces the cost of repair. Therefore, I choose centrality as the main metric and whenever this metric is the same for several nodes, I 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

This section describes how monitor nodes probe the surrounding network to derive more information on the status of the reachable nodes and links. I 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 solution of the current iteration of the approximate does not change after discovery phase, I 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 the algorithm the next solution set $S(t + 1)$ remains the same and therefore I 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 the optimization needs to run only if $X - X' < Y - Y'$ because there exists a possibility that there is a better solution other than the current solution.

2.5 Comparison with Prior Works

In this section, I introduce two prior works, ISP [6] and CeDAR [16] that aim at recovering the network progressively after large-scale disruption. Since the state-of-the-art ISP algorithm assumes perfect knowledge of the failed components, I modify it to work under uncertainty and call it progressive ISP.

2.5.1 Progressive ISP

This section describes progressive ISP which is my 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 I 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, I 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. I 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 divided by c_{ij} to give higher cost to the paths which have smaller capacity. Further, we put monitoring software on the node which is chosen to split the demand at each iteration to discover the gray area. However, once the demand splits over a candidate node, a routing decision is made on the selected node. Therefore, as I will see in Section 2.6, 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 I use the terms “ progressive ISP ” and “ ISP ” interchangeably.

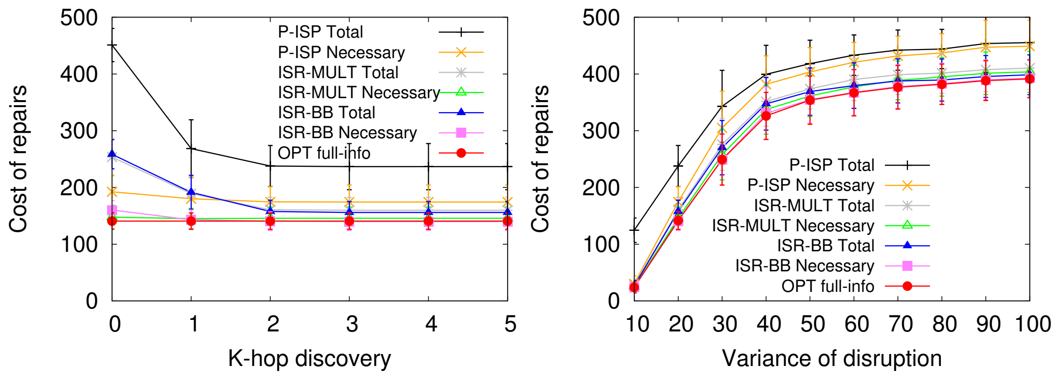
2.5.2 CEDAR

Ciavarella et al. [16] propose a polynomial-time heuristic called *Centrality based Damage Assessment and Recovery* (CeDAR) that progressively recovers the network under incomplete knowledge of failure. CeDAR aims at maximizing the total satisfied flow during the recovery process. At each iteration step of CeDAR, a limited amount of budget is available to repair nodes/edges and the repairs are scheduled according to the availability of the resources at each time step. While my progressive recovery approach and CeDAR differ in the objective function, they both propose a progressive recovery under incomplete knowledge of failures and therefore, I compare my result with CeDAR in Section 2.6 in terms of number of repaired nodes/edges and monitors placed.

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

2.6 Evaluation

In this section, I compare ISR algorithms, presented in 2.4, to the modified version of ISP introduced in Sections 2.5.1. I use different network topologies including



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

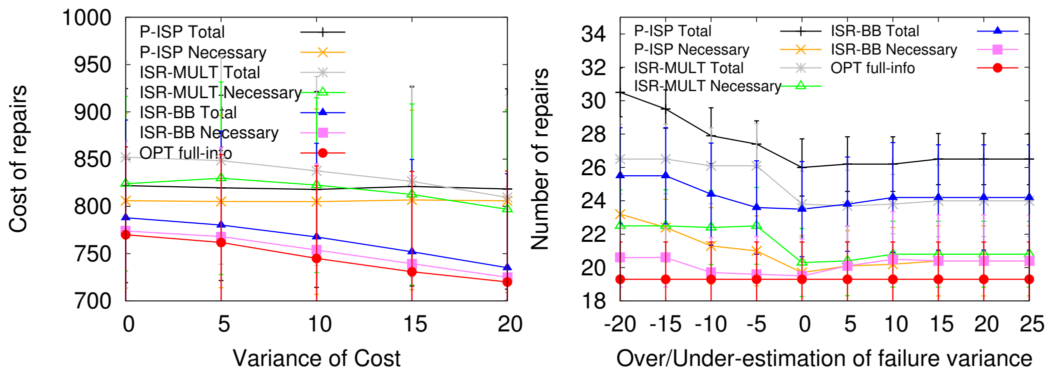
Figure 2.4: m-hop discovery and disruption variance, (BellCanada, m-hop=2).

Table 2.3: Network characteristics used in the 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

planar and non-planar real topologies taken from the Internet Topology Zoo [1,5]. Table 5.2 shows the characteristics of the topologies used for the evaluation. In addition to the real network topologies, I use synthetic Erdos-Renyi graphs with 100 nodes, where I 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, I consider several scenarios, in which I 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 I randomize the results running 20 different trials, in which, depending on the scenario, I vary the random selection of source/destination pairs and the random disruption of network elements. I implement my recovery algorithms in python and used the *Gurobi* optimization toolkit, on a 24-core, 2.6 GHz, 32G RAM cluster [42].



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

Figure 2.5: a) The impact of heterogeneous repair cost variation on total cost of repair, b) Over/under-estimation of the disruption by adding an error between -20 to 25 to the variance (BellCanada, m-hop=2).

2.6.1 m-hop Discovery Impact

In this section, I investigate the impact of the depth of discovery phase on the performance of the proposed algorithm. I changed the number of discovered hops for the monitoring nodes from 1 to 5. I used the BellCanada topology with 10 demand pairs and 4 units of flow per demand. The link capacity is set randomly in the interval $[20, 50]$. I used heterogeneous repair cost for each node randomly from a uniform distribution in $[0, 10]$ and for each edge from a uniform distribution in $[0, 20]$. I used higher repair cost for edges due to difficulty in locating the failure and accessibility. From Figure 2.4a, I can see that increasing the number of discovered hops improves the restoring performance of my algorithms in terms of total repair cost. I performed similar experiments with different topologies, which I 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 2.4b, I changed the amount of disruption in the network to evaluate the performance of the algorithms. I 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 $[20, 50]$. I 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 2.4b shows the simulation results for this scenario. I 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 I increase the disruption variation, the total number of repairs increases until it gets saturated and the whole network gets disrupted. Therefore, the uncertainty in the gray area has less impact on the restoring performance of the algorithms because the whole gray area is failed. Furthermore, the number of necessary and unnecessary repairs is the same for dense disruptions since most of the nodes in the network are failed in higher disruption variations and the discovery phase does not help to reduce the number of unnecessary repairs by a large amount.

2.6.3 Heterogeneous Repair Cost

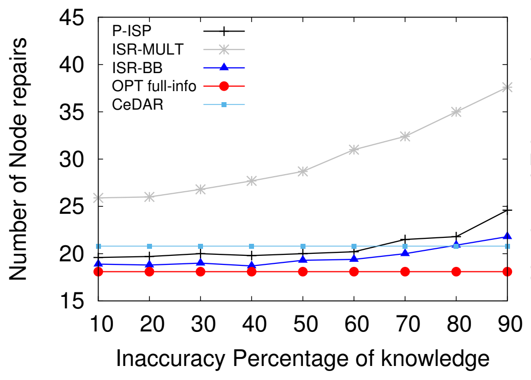
In this scenario, I analyze the impact of heterogeneous repair cost. I considered BellCanada topology with 5 demand pairs and 5 units of flow per demand pair. The link capacity is set randomly in the interval [20, 50]. I considered a scenario where the whole network is disrupted and used heterogeneous repair cost with the average of 20 derived from a uniform distribution, and changed the variance of cost from 0 to 20. Figure 2.5a shows the total and necessary repair cost for this scenario. As shown, my 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 I consider homogeneous repair cost.

2.6.4 Sensitivity Analysis

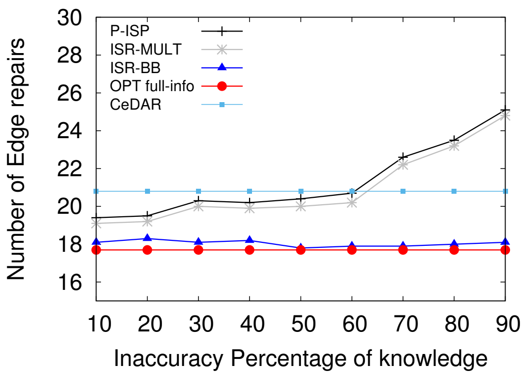
In the next set of experiments, I study the sensitivity of the proposed algorithm with respect to the correctness of the initial failure estimation. I use the BellCanada topology where the link capacity is set randomly in the interval $[20, 50]$. The network disruption is randomly generated according to a Gaussian geographic distribution with variance of 50 that destroys 50% of network components on average. I consider a varying error in the estimate of the disruption extent, and I overestimate/underestimate the disruption by adding an error between -20 to 25 to the variance of the disruption. Figure 2.5b 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. I observe that when I 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, I am 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 Impact of Accuracy of Estimate in the Initial Probability Distribution

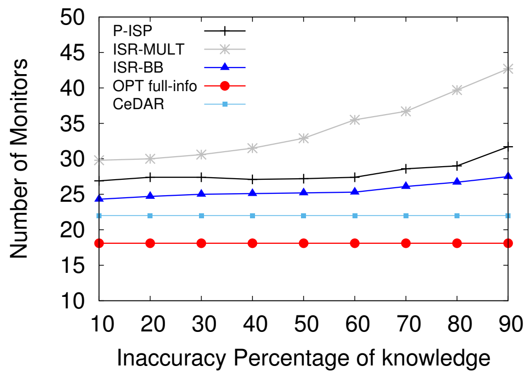
In this set of experiments, I use the DeltaCom topology with 6 demand pairs and 5 units of flow per demand pair, where the link capacity is set randomly in the interval $[20, 50]$. The network disruption is randomly generated according to a geographic failure, where 68% of network elements reside inside a circle within which the network elements fail with a probability of 0.9 and the rest of the network elements fail with a probability of 0.1. I vary the accuracy of the estimate of the failure probability from 10% to 90%. For example, if the accuracy measure on the x-axis is 10, then the assumption on the failure probability is off by 10%, i.e. I estimate a failure of 0.8 or 0.2 for the network elements within or outside the circle, respectively. Figure 2.6 shows the number of nodes repaired (Fig 2.6a); number of edges repaired (Fig 2.6b); and monitors placed (Fig 2.6c) as I decrease the accuracy



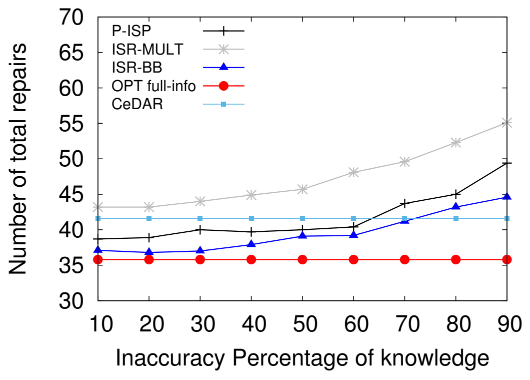
(a) Number of node repairs..



(b) Number of edge repairs.



(c) Number of monitors.



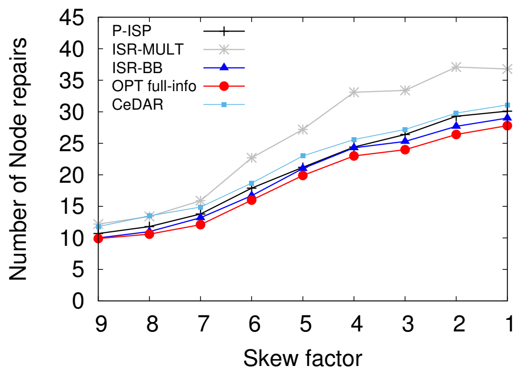
(d) Number of total repairs.

Figure 2.6: Impact of imprecision in the accuracy of the probability of failed nodes and edges, 50% disruption.

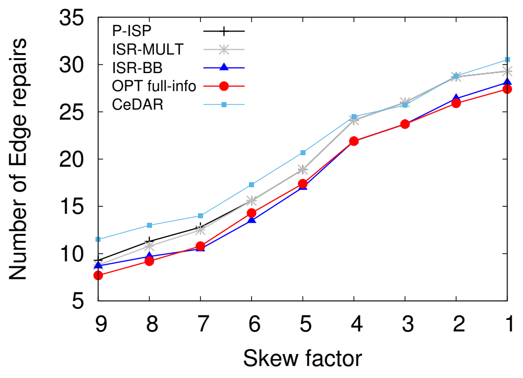
of my assumed probability distribution of failures.

2.6.6 Skew Factor

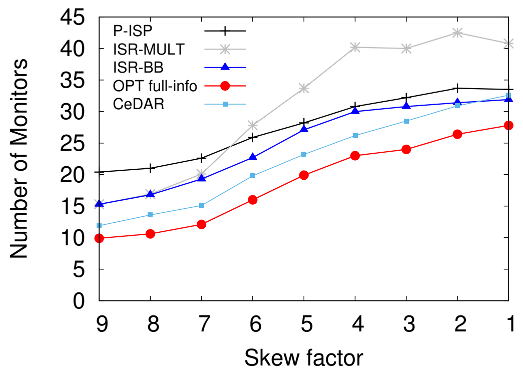
In this section, I use the DeltaCom topology with 6 demand pairs and 5 units of flow per demand pair, where the link capacity is set randomly in the interval $[20, 50]$. The network disruption is randomly generated according to a geographic failure, where 50% of network elements reside inside a circle within which the network elements fail with a probability of 0.1 and the rest of the network elements fail with a probability of 0.9. This allows me to evaluate the impact of the certainty of the failure on my algorithm. I expect that the higher the certainty I have of the status of a gray network element, the better the results.



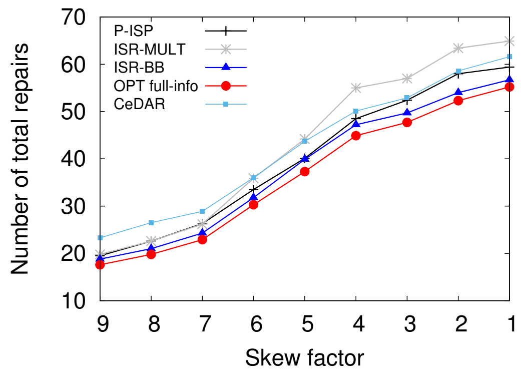
(a) Number of node repairs.



(b) Number of edge repairs.



(c) Number of monitors.



(d) Number of total repairs.

Figure 2.7: Impact of skew factor in the accuracy of the probability of failed nodes and edges, 50% disruption.

I then increase the probability of failure inside the circle from 0.1 to 0.9. The skew factor is 9 in the beginning and decreases to 1 as I increase the failure probability of the circle. Figure 2.7 shows the number of nodes repaired (Fig 2.7a); number of edges repaired (Fig 2.7b); and monitors placed (Fig 2.7c) as I increase the imprecision in my initial the probability distribution of failures.

2.6.7 Trade-off on Demand Loss, Time Complexity, and Number of Repairs

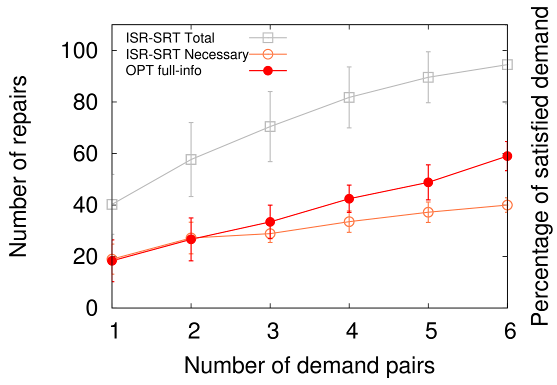
The recovery problem can be addressed by giving different priority to performance aspects such as: 1) demand loss, 2) execution time and 3) number of repairs (cost). These aspects are in conflict with each other; therefore, I study the trade-off among them.

In this scenario, addressed in Figure 2.8, I considered the Deltacom topology, where I set the link capacity randomly in the interval $[20, 30]$. I compare ISR-SRT to OPT to determine the amount of demand flow loss in ISR-SRT. I 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.

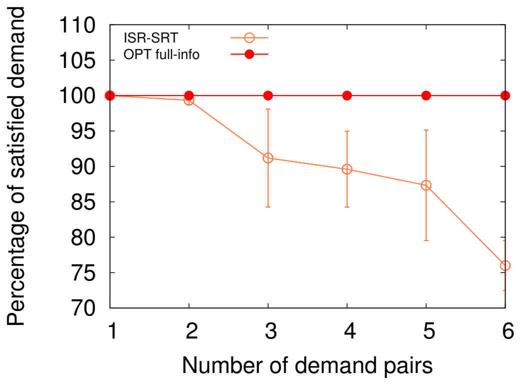
Figure 2.8a shows that ISR-SRT performs a smaller number of necessary repairs than OPT but a much higher number of total repairs, meaning that ISR-SRT schedules repairs for nodes that are found to be working. Figure 2.8b also shows that ISR-SRT does not meet the demand requirements. The percentage of 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. I 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 my goal is to restore all critical services, I do not further evaluate ISR-SRT. However, due to its low computational complexity, the algorithm can be used in scenarios where the demand load is low and a short computation time is required.

In the next experiment I used the same topology, under a larger disruption, corresponding to 75% of network elements on average. I consider 5 demand pairs, of 17 flow units each. In order to evidence the tradeoff between the number of repairs



(a) repairs.



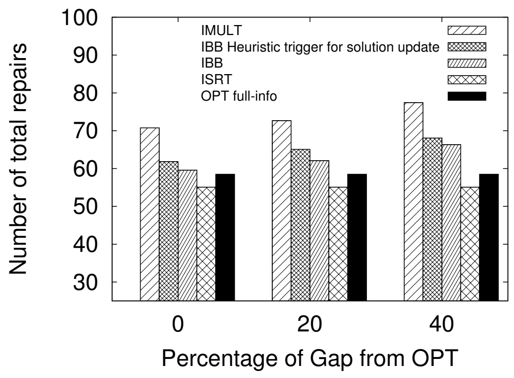
(b) demand loss.

Figure 2.8: Trade-off between number of repairs and demand loss.

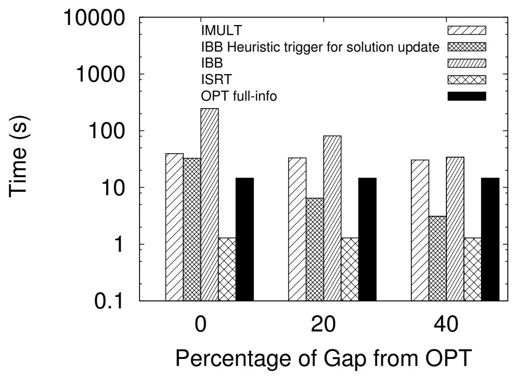
and computation time, in Figure 2.9 I vary the gap between the lower bound of the objective function and the solution of iterative ISR-BB and ISR-MULT algorithms from 0 to 40%. I recall that by increasing this gap, I decrease the number of iterations of the optimization algorithms, and therefore I obtain 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 2.9a shows that, when I 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, I 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. I 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, I used synthetic Erdos-Renyi non-planar graphs. In an Erdos-Renyi graph, any two nodes are connected through an edge with probability p. I considered an Erdos-Renyi topology with 100 nodes where each link has a capacity of 1,000 units of flow. I 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 I increase the parameter p and

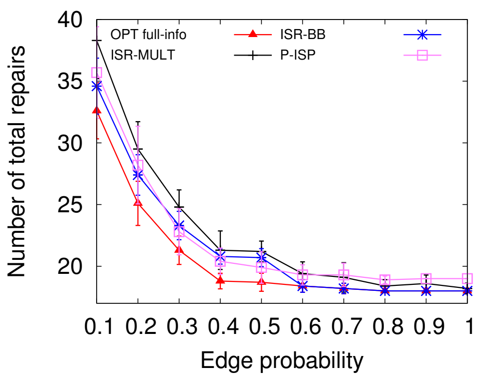


(a) repairs.

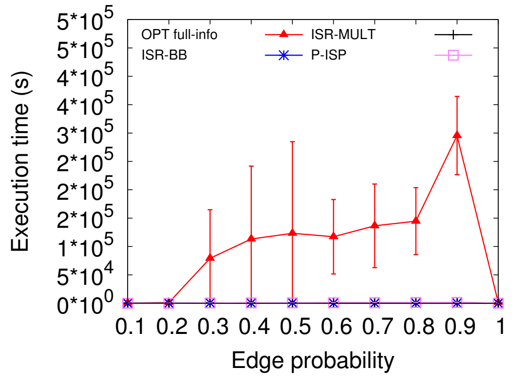


(b) time.

Figure 2.9: Trade-off between execution time and repairs.

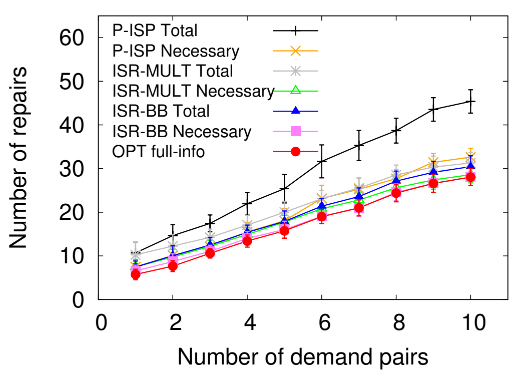


(a) repairs.

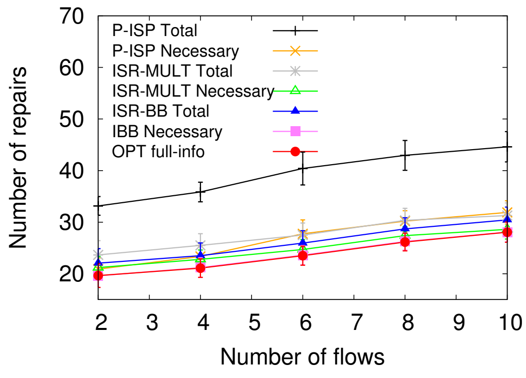


(b) time.

Figure 2.10: Synthetic Erdos-Renyi topology with 100 nodes.



(a) Increasing demand.



(b) Increasing flows.

Figure 2.11: Increasing demand pairs and flows (BellCanada).

the graph becomes gradually non planar. I 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, I 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, I put all fractional variables in the solution and select a node to repair and continue this procedure till all critical services are restored. Figure 2.10b shows the execution time of the approximate solutions with respect to optimal as I increase the value of p. I observe that, since MINER is NP-Hard, the optimal recovery with full information has a very high execution time, while if I 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.

2.6.8 Increasing Number of Demand Pairs and Amount of Flow

In this section, I 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. I consider the BellCanada topology, where I set random link capacity with values in the interval $[20, 50]$. I increased the number of demand pairs from 1 to 10, where each demand has a requirement of 10 units of flow. Figure 2.11a shows the simulation results for this scenario. I used a Gaussian disruption with disruption variance of 20, which destroys around 40% of the network. As I 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, I consider the same network topology, and same disruption parameters. I set the number of critical demand pairs to 5 and increased the units of flow per demand pair from 2 to 10. Figure 2.11b shows the simulation results in this scenario for my iterative algorithms and optimal recovery with full knowledge. I observe that for less than 4 units of flow, ISP performs slightly better

Table 2.4: 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.

than the ISR-MULT solution in terms of number of necessary repairs. However, as I 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. I observe that in larger topologies, ISP performs better than ISR-MULT when the total demand load (sum of all the demand flow requirements for all the demand pairs) is lower than 40% of the network capacity. This opens up the opportunity to have a hybrid scenario for low flow/pair and high flow/pair scenarios where one can get advantage of progressive ISP under low demand load and the ISR-MULT for higher demand load.

Table 2.4 shows the comparison between progressive ISP, ISR-MULT, ISR-BB and ISR-SRT. I observed that each of the proposed algorithms has pros and cons, which makes them suitable for scenarios where I 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, for the first time, a progressive network recovery algorithm under uncertainty. I 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 I 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. I iterate the process, alternating monitoring and repair activities, until all critical services are restored. Simulation results show that ISR reduces the total cost of repair significantly with respect to the state-of-the-art ISP algorithm. I also observed a configurable choice of trade-off between the demand loss, total number of repairs and execution time.

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. I focus on cascading failures involving the power grid and its communication network with consequent imprecision in damage assessment. I tackle the problem of mitigating the ongoing cascading failure and providing a recovery strategy. I propose a failure mitigation strategy in two steps: 1) Once a cascading failure is detected, I limit further propagation by re-distributing the generator and load's power. 2) I formulate a recovery plan to maximize the total amount of power delivered to the demand loads during the recovery intervention. My approach to cope with insufficient knowledge of damage locations is based on the use of a new algorithm to determine consistent failure sets (*CFS*). I 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

Today's critical infrastructures are highly interdependent. Because of the interdependency between different components, perturbations caused by failures, physical attacks or natural disasters may propagate across different networks. Examples of such coupled critical infrastructures include the food supply and water systems, financial transactions and power grids, transportation systems and food supply, etc. [43, 44]. These 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.

One of the most critical infrastructures in our everyday lives is the power grid. Large-scale blackouts in the power grid due to propagating failures, natural disasters or malicious attacks can severely affect the operation of other interdependent 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 [3]. A similar event occurred the same year in the Northeast of the United States, leading to over 50 million people losing power for several days. The cascade lasted approximately for four hours, a time sufficient for enabling countermeasures which could have mitigated and limited the blackout propagation. This highlights the necessity of a coherent power control strategy that allows prompt intervention to mitigate or stop the failure propagation. Furthermore, it is crucial to have a strategic recovery plan that ensures effective use of the available resources during the recovery process.

Despite considerable amount of research in the past few decades leading to major improvements in the reliability of communication and power networks, most of the research focused on the recovery of a single network [6–9, 45]. Unlike previous work, I jointly address the following three challenges:

- Providing a realistic model of cascading failures in interdependent networks which takes into account the peculiarity of both the communication network and the power grid, and overcomes the limitations of the simplified epidemic models which cannot represent real infrastructures [46, 47].

- Modeling lack of knowledge in failure localization, by considering the potential uncertainty due to failures in the monitoring systems, which may hamper the use of appropriate countermeasures to prevent a cascading failure or recover network services.

- Progressively restoring the network after a large-scale disruption or a cascading failure in multiple stages, within the limits of recovery resources (time, cost, human personnel).

I make the following contributions to address the above challenges:

- To cope with insufficient knowledge of failure locations, I propose a new Consistent Failure Set (*CFS*) algorithm to determine all failure scenarios, namely the sets of components whose failure is consistent with the observations. *CFS* is used to determine local monitoring interventions that allow the identification of the state of all network components.

- I tackle the problem of mitigating an ongoing cascade (first phase) by formulating a minimum cost flow assignment (*Min-CFA*) as a linear programming optimization problem. *Min-CFA* aims at finding a DC power flow setting that stops the cascading failure with minimum change in power generation and satisfied demand.

- I formulate a progressive recovery problem (second phase) to maximize the satisfied demands (*Max-R*) over multiple consecutive recovery interventions based on the available recovery resources at each stage. I show that *Max-R* is NP-hard, hence I propose two heuristic recovery strategies: 1) *Max-R-Greedy*, as a baseline algorithm, and 2) *Max-R-Backward*, which consider different time spans in scheduling the recovery interventions.

- I perform an extensive experimental evaluation, considering a real network scenario of interdependent power grid and communication network, under different interdependency models. The experiments highlight the efficiency of the proposed cascade prevention algorithm. For example, in a scenario where 60% of the lines of the power grid has failed, the adoption of *Min-CFA* stops the propagation with 54% of served load, while without intervention, the entire system would fail. In the same scenario, the proposed backward recovery approach *Max-R-Backward* performs full recovery, serving on average 20% more accumulative load than the baseline algorithm *Max-R-Greedy*.

3.2 Related Work

The existing works on cascading failures in interdependent networks, of interest to my approach, can be broadly classified into two categories: 1) those, that study the interaction through percolation theory and stochastic analysis [46–49], 2) studies that try to identify the most vulnerable nodes [34, 50], and aim at finding the root cause of failures and identify performance degradation [12, 51, 52].

The approaches of the first category rely on prior knowledge of 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 certainly stop working.

Concerning the works of the second category, I underline that finding the root cause of the propagating failure is key to the design of failure-resilient systems, but does not provide a mitigation solution.

Another line of research addresses the problem of cascading failure in the power grid and studies the peculiarity of the propagation across power lines. Cascading failures in power grids can be due to a permanent failure, e.g. a tree falls on a transmission line etc., or to a temporary failure, e.g. a temporary short circuit in a transmission line. When a short circuit happens in a transmission line, a protective overcurrent relay sends a “trip” signal to the breakers and the breakers set open. Then the relay attempts to re-close the breaker a few times. In case of a permanent failure, auto-reclosing fails and 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 the overload of other lines, trigger new failures, spreading over the entire network.

Unlike the approach proposed in [53], that re-distributes the power flow evenly over all transmission lines, I use the DC power flow model [54, 55]. Notice that this model is widely used in studies of cascading failures in power grids and is acknowledged to be a good approximation of the AC power flow model.

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\Energy Management System (SCADA\EMS) and more recently, the Wide-Area Monitoring, Protection, and Control (WAMPAC) system to respond to emergency situations. The required data is measured and gathered at the substations from the Intelligent Electronic Devices (IEDs), fault recorders, breaker status monitors, and Phasor Measurement Units (PMUs) [56, 57]. While substation automation is increasing the intelligence and autonomy of local protection units, it is facilitating the trend of centralized wide-area protection popularly called the Remedial Action Scheme (RAS) or the Special Protection Scheme (SPS). In the latter, data is communicated to a centralized location, which can initiate corrective actions such as generation re-dispatch or load shedding to remote locations. A nice description can be found in [58] and [59] and references therein. The security of such centralized control systems is itself an important challenge on the reliability of the power grids (e.g. a compromised RAS can send an anomalous load shedding signal to disrupt the power grid) [60–63].

Concerning the problem of restoring network functionalities, previous work only considers one homogeneous network [6–9, 45]. The proposed recovery approach in this chapter, takes inspiration from the works in [7, 8, 45] and aims at restoring the functionality of multiple interdependent networks in a progressive manner, depending on resource and incremental knowledge availability. The proposed recovery approach represents a step ahead with respect to these previous contributions in that it deals with heterogeneous interdependent networks, and includes a runtime feedback control of the flow during the recovery process. To the best of my knowledge, the problem of jointly mitigating and recovering from cascading failures, during the transient regime of the propagation process, was never studied extensively before as I do in the present work. I address such a problem with particular focus on failures in the power grid and the interdependent communication network.

3.3 Network Model and Background

I model the interdependency between the power grid and the communication system for which some failures are detected while the propagation is still in progress. I propose a mitigation strategy to minimize further cascades and a recovery plan

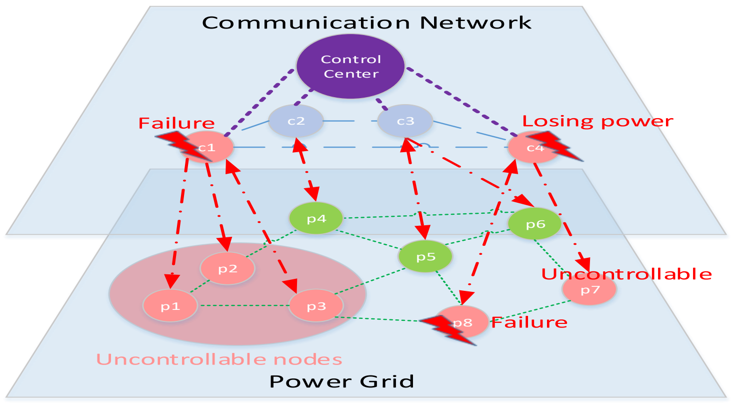


Figure 3.1: Interdependency model between a power grid and a communication network.

to entirely restore the power grid functionality, while maximizing the accumulative amount of delivered power during multiple stages of progressive recovery. My approach can be extended to the use of other measures related to the operation of the grid such as the total number of working power lines, etc. Table 5.1 shows the notation used in this chapter.

3.3.1 Notation

The power and communication networks are modeled as undirected graphs $\mathcal{G}_p = (V_p, E_p)$ and $\mathcal{G}_c = (V_c, E_c)$, respectively. Each node i in the power grid is monitored by several sensors deployed nearby. The monitoring data is then sent to the node of the communication network which hosts the control functionalities related to node i of the power grid. In addition, control commands are sent to the dependent communication node for generator re-dispatch or load shedding. I acknowledge that several other emergency actions could be taken (e.g. system separation, dynamic braking, fast valving, etc.), which are not considered here.
The set of nodes V_p of the power grid is composed by three disjoint sets of generators G where the power is generated, loads L where the power is consumed, and junctions J where power flows by, with $V_p = G \cup J \cup L$.

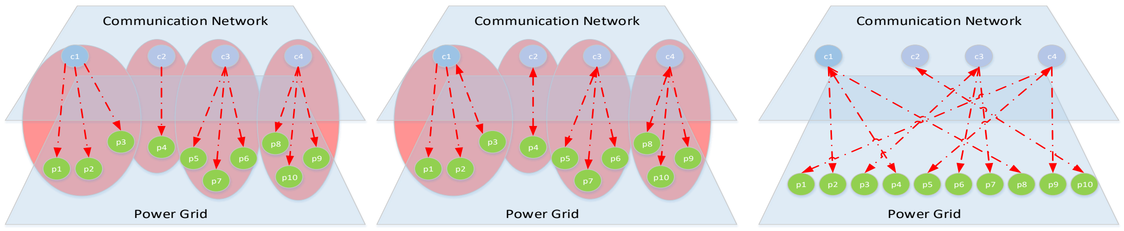
As node failures are less likely [64], I hereby assume that initial failures only occur in power lines (E_p). A node in the power grid is considered failed if it is not able

Table 3.1: Summary of notations.

Notation	Explanation
$\mathcal{G}_p = (V_p, E_p)$	undirected graph modeling the power grid. V_p is the set of nodes and E_p is the set of links
$\mathcal{G}_c = (V_c, E_c)$	undirected graph modeling the communication network. V_c is the set of nodes and E_c is the set of links
$G \subset V_p$	set of generator nodes G_i where power is generated
$L \subset V_p$	set of load nodes L_i where the power is consumed
$J \subset V_p$	set of junction nodes J_i where the power just flows
$E_p^{B,t} \subseteq E_p$	set of broken (red) edges
$E_p^{U,t} \subseteq E_p$	set of unknown status (grey) edges
$E_p^{W,t} \subseteq E_p$	set of working (green) edges
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
Y^t	nodal admittance matrix at time t
w_{G_i}	cost of increasing the power in generator G_i
w_{L_i}	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 stage k
$\delta_{(ij),k}$	decision to repair $(ij) \in E_p^{B,t}$ at the kth stage
r_{ij}	resources needed for repairing (ij)
R_k	available resource at stage k of the recovery
α_i	a constant factor showing the reduction of the new power distribution in node i, where $0 \leq \alpha_i \leq 1$.

to deliver the required power to the loads. Further, I consider the interdependency between the power grid and the communication network such that (i) failures in the communication network lead to lack of information and controllability of power grid in the control center, and (ii) failures in the power grid can cause further failures in the communication system due to lack of power. The edges in the power grid graph $\mathcal{G}_p$ can be in three different states: 1) the set $E_p^{B,t} \subseteq E_p$ is the set of **broken** edges (hereby denoted as red edges) at time t [1]; 2) the set $E_p^{U,t} \subseteq E_p$ is

[1]Notice that in order to be able to assess an edge failure, the edge must be connected to a working communication node in $\mathcal{G}_c$. The working communication node provides the failure status



(a) One-way (b) Location-based (c) Random

Figure 3.2: Three interdependency model between a power grid and a communication network: a) One-way, 2) Location-based and 3) Random.

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

3.3.2 Interdependency Model

To clarify the interdependency model between the communication network and the power grid, consider the example shown in Figure 3.1. The figure shows the interdependency model between a communication network with 4 nodes $\{c_1, .., c_4\}$, and a power grid with 8 nodes $\{p_1, ..., p_8\}$. The red arrows show the interdependency between the two networks. For example, c_1 controls three power nodes $\{p_1, p_2, p_3\}$ and gets power from p_3. Now consider a failure in one of the communication nodes c_1. In this case three power grid nodes $\{p_1, p_2, p_3\}$ become uncontrollable as the controller cannot send the power adjustment control commands to them. Next, consider a failure in a node in the power grid p_8. In this case, the communication node c_4 that gets power from p_8 loses power and consequently the dependent power grid node p_7 becomes uncontrollable. In this work, I consider three types of interdependency models between the communication network and the power grid: the one-way interdependency model (Fig. 3.2a); the location-based interdependency model (Fig. 3.2b); the random interdependency model (Fig. 3.2c).

In the one-way interdependency model I assume the power lines are monitored and controlled by the closest communication node, while if a grid node fails, the

of the edge to the central controller and can send power adjustment commands to the connected loads or generators.

communication nodes get backup power from an external source (e.g., battery). This is the case for many telecommunication deployments with battery backup. In the one-way interdependency model, failures in the power grid will not cascade in the communication network. However, the disruption in the communication network is observed as lack of knowledge and uncontrollability in the power grid.

As in the one-way model, in the location-based interdependency model, each power line is monitored and controlled by the closest communication node. Nevertheless, in the location-based model I assume that each node in the communication network gets power from the closest node in the power grid.

Both in the one-way and location-based interdependency models, the communication network and power grid are divided into dependent regions and the failure in one region does not cascade or affect other nodes in a different region.

In the random interdependency model, I assume each power line is monitored and controlled by a random communication node and each communication node gets power from a random substation in the power grid. In the random interdependency model, the failure in one node may cascade and spread over the entire network.

Notice that most prior works on the interdependency between the power grid and the communication network use a random interdependency model (see examples in [47, 65–67]) and focus on the ping-pong failures from the power grid to the communication network and vice versa.

While a random interdependency model is easy to analyze and simulate on synthetic graphs, it fails to capture most real network topologies. Nevertheless, I include this model in my analysis for the sake of completeness in providing comparisons with previous work and because I consider this as a stress-test due to its potentiality to produce larger cascades.

By comparing the experiments conducted on the random interdependency model with the ones using location-based and one-way interdependency models, in Section 3.6.1 I will show that the extent of the cascading failure phenomenon is mostly related to the lack of controllability consequent to failures in the communication network.

3.3.3 DC Power Flow Model

I model the cascading failure in a power system using a DC load flow model [54]. Let F_{ij}^t be the power flow in line (ij) at time t, x_{ij}^t be the series reactance of line (ij) and θ_i^t and θ_j^t be the voltage angles of node i and j, at time t[2]. 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.

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

The power flow of node i can be found by summing up the power flows of all its adjacent power lines:

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

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

$$P^t = Y^t \theta^t, \tag{3.3}$$

where Y^t is nodal admittance matrix at time t, $y_{ij}^t = -\frac{1}{x_{ij}^t}$ for $i \neq j$, and $y_{ii}^t = \sum\limits_k \frac{1}{x_{ik}^t}$ [68]. Once events like over-current are detected in a transmission line (ij), a protective relay trips a circuit breaker ($x_{ij} = \infty$) and the power is redistributed according to the DC model. In particular, if the current flow exceeds the maximum threshold on another line $(i'j')$, in a cascading manner, the transmission line $(i'j')$ may also trip.

Notice that in order to determine the power flowing through each line after one or more failures, I need to solve the system of equations 3.1, which requires the solution of Equation 3.3 to obtain the values of the vector θ^t, for each connected component of the power grid graph.

Remark 1 *[69] The system of equations (3.3) has a feasible solution for each connected component of the power graph.*

Discussion on remark 1 Let us consider a unique connected component. The nodal admittance matrix Y of a connected graph with n nodes has always rank

[2]Notice that the reactance x_{ij}^t of line (i,j) varies with t as a consequence of failures or recovery events.

$(n-1)$ because one can construct a graphic matroid 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 $(n-1)$ [69, 70]. To make this equation solvable, one of the equations is removed and the corresponding node can be chosen as a reference node. Without loss of generality, the removal of the first equation implies $\theta_0 = 0$ and the other $n-1$ values of the vector θ can be calculated inverting the system of equations 3.3 reduced after the removal of the first equation, according to the technique in [71, 72]. The reduced admittance matrix has full rank and thus invertible. If instead the power grid graph has c connected components due to the disruption of several lines, then the admittance matrix will have rank $n-c$ and each component must be addressed by means of the same technique. Let Y'^t be the admittance matrix of a connected component, and θ'^t its phasor vector, let also P'^t be its power vector, at time t. Then the power flow system of equations of the considered connected component is $P'^t = Y'^t\theta'^t$, which can be solved independently of the other connected components, in the way described for the case of a unique connected component, with the removal of one equation and the introduction of a reference phasor vector, as described in [71, 72]. Therefore, the DC power flow model for each connected component of the graph has a feasible solution.

3.4 Cascade Mitigation and Network Recovery

In this section, I address the problem of cascading failure mitigation and related recovery process in an interdependent network formed by a power grid and a communication network. Figure 3.3 illustrates the proposed two-phase approach. As described in the figure, whenever a new failure event shows up, a preliminary monitoring activity is performed to localize the failure sites. After the failure assessment it follows a first phase in which further cascades are mitigated or prevented by means of a combination of load shedding and adjustment of the generated power. Once the cascade is stopped, a progressive recovery activity follows, using either a greedy or backward approach. Recovery is performed in multiple stages according to resource availability. After the system is recovered, the monitoring activity restarts, until new failures occur.

3.4.0.1 Cascade Mitigation

Once I detect an outage of a transmission line, I readjust power and load according to the optimization problem described in the following.

For clarity of presentation, I hereby assume that the power grid is formed by a unique connected component. Notice that in the presence of multiple connected component, all the following techniques are still valid, when applied to each connected component, independently. 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 prevent the cascading failures can be formulated as follows:

$$\begin{aligned}
\text{minimize} \quad & \sum_{G_i \in G, L_j \in L} w_{G_i}(P^0_{G_i} - P^t_{G_i}) - w_{L_j}(P^t_{L_j} - P^0_{L_j}) \\
\text{subject to} \quad & 0 \leqslant P^t_{G_i} \leqslant P^0_{G_i}, \ \ \forall G_i \in G && (3.4a) \\
& 0 \leqslant P^t_{L_j} \leqslant P^0_{L_j}, \ \ \forall L_j \in L && (3.4b) \\
& -F^{max}_{ij} \leqslant F^t_{ij} \leqslant F^{max}_{ij}, \ \ \forall (ij) \in E^t_p && (3.4c) \\
& \sum_{G_i \in G} P^t_{G_i} + \sum_{L_j \in L} P^t_{L_j} = 0 && (3.4d) \\
& P^t_{G_i} = \sum_{j:(G_i,j) \in E^t_P} F^t_{ij}, \ \ \forall G_i \in G && (3.4e) \\
& -P^t_{L_i} = \sum_{j:(L_i,j) \in E^t_P} F^t_{ij}, \ \ \forall L_i \in L && (3.4f) \\
& P^t = Y^t \theta^t && (3.4g) \\
& F^t_{ij} = \frac{(\theta^t_i - \theta^t_j)}{x_{ij}}, \ \ \forall (ij) \in E^t_p && (3.4h)
\end{aligned}$$

The decision variables in *Min-CFA* are the continuous values of power in the generators ($P^t_{G_i}$) and loads ($P^t_{L_j}$).

Constraint 3.4a indicates that the power generated at each generator at time t cannot exceed the initial power observed at time 0. In case of full knowledge of the location of failures, I could have a more relaxed constraint to increase the power at some of the generators without violating a maximum threshold. However,

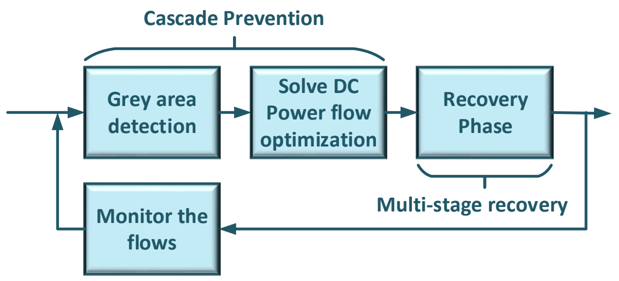


Figure 3.3: Recovery Process: 1) Cascade mitigation phase, and 2) Recovery phase.

under uncertain failure I reduce my solution space to decrease the possibility of consequent cascades due to imperfect knowledge. Constraint 3.4b shows that the reduced load cannot exceed the demand. Constraint 3.4c shows that the power flowing through each line cannot exceed the maximum capacity of the line (*thermal constraint*). Constraint 3.4d 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. Constraints 3.4e and 3.4f show that the total power generated/consumed at each node should be equal to the total power flowing through its edges. Constraints 3.4g and 3.4h reflect the DC power flow model, to be solved according to Remark 1 [72].

I next provide the sufficient condition for each connected component of the power grid, to ensure that the solution of the power re-distribution model under uncertain knowledge of the failure does not increase the power flowing through any line in the system, making it exceed the thermal threshold given by constraint 3.4c. Suppose that in the new power assignment for each node i, P_i is reduced by a factor α_i, i.e. $P_i^t = \alpha_i P_i^0$, for $i = 0, \ldots, n-1$, where $0 \leq \alpha_i \leq 1$. Without loss of generality, let the reference voltage angle be at node 0 ($\theta_0 = 0$). Also, using the DC power flow model, the voltage angles at each node is calculated as follows:

$$\theta_i = \sum_{m=1}^{n-1} k_{im} P_m, \qquad \text{with} \qquad k_{im} \geq 0, \tag{3.5}$$

where n is the number of nodes in the power grid, and k_{im} reflects impedance values, that are always non negative [71, 72].

Let $\alpha^- \triangleq min\{\alpha_i : i = 1, ..., n-1\}$ and $\alpha^+ \triangleq max\{\alpha_i : i = 1, ..., n-1\}$. A sufficient condition for not having cascade propagation is the following.

Theorem 2 *Sufficient condition for no additional cascading failures in Min-CFA problem is as follows:*

$$\frac{1-\alpha^-}{1-\alpha^+} \leq \frac{\sum_{m:G_m\in G}(k_{im}-k_{jm})P_m}{\sum_{m:L_m\in L}(k_{im}-k_{jm})(-P_m)} \quad \forall (ij)\in E_p^t \tag{3.6}$$

Proof of Theorem 2 Without loss of generality, I assume $F_{ij} \geq 0$, otherwise I can simply interchange i and j index. In order to check whether the new power flow $F_{ij,new}$ provided by my control approach is not larger than the previous flow F_{ij}, I need to check if $F_{ij} \geq F_{ij,new}$, which is equivalent to check if $(\theta_i - \theta_j) \geq (\theta_{i,new} - \theta_{j,new})$. Note that the new and previous voltage angles at each node is computed as follows:

$$\theta_i = \sum_{m=1}^{n-1} k_{im}P_m, \text{ where } k_{im} \geq 0$$

$$\theta_{i,new} = \sum_{m=1}^{n-1} k_{im}\alpha_m P_m, \text{ where } 0 \leq \alpha_m \leq 1, \quad k_{im} \geq 0$$

Therefore, I have to verify the following inequality:

$$(\sum_{m=1}^{n-1} k_{im}P_m - \sum_{m=1}^{n-1} k_{jm}P_m) \geq (\sum_{m=1}^{n-1} k_{im}\alpha_m P_m - \sum_{m=1}^{n-1} k_{jm}\alpha_m P_m)$$

To check if the flow in F_{ij} does not increase after the power flow adjustment, I need to verify the following inequality, for each edge $(i,j) \in E_p^t$:

$$\sum_{m:G_m\in G}(k_{im}-k_{jm})P_m(1-\alpha_m) \geq \sum_{m:L_m\in L}(k_{im}-k_{jm})P_m(\alpha_m - 1) \tag{3.7}$$

Now I show that if condition 3.6 is satisfied for every flow, I 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. We have:

$$\begin{aligned}
&\sum_{m:G_m\in G}(k_{im}-k_{jm})P_m(1-\alpha_m) \geq \\
&\sum_{m:G_m\in G}(k_{im}-k_{jm})P_m(1-\alpha^+) = \\
&(1-\alpha^+)\sum_{m:G_m\in G}(k_{im}-k_{jm})P_m \geq \\
&(1-\alpha^-)\sum_{m:L_m\in L}(k_{im}-k_{jm})(-P_m) = \\
&\sum_{m:L_m\in L}(k_{im}-k_{jm})(-P_m)(1-\alpha^-) \geq \\
&\sum_{m:L_m\in L}(k_{im}-k_{jm})(-P_m)(1-\alpha_m)
\end{aligned}$$

Therefore, if equation 3.6 holds, the inequality 3.7 is also verified, which implies that the new power set at each line does not exceed the previous value.

3.4.0.2 Recovery Phase

In this paragraph I address the problem of scheduling recovery interventions in order to maximize the total accumulative flow absorbed by the loads during K stages of recovery. The number of stages can be set according to the assumption that at least one edge can be repaired at each stage. Therefore K can be set equal to the number of broken edges. I hereafter refer to the multiple stages of progressive recovery shortly with the word *stages*.

Let $\delta_{(ij),k}$ be a binary variable representing the decision to repair edge (i,j) at time $k = 1,\ldots,K$. Namely, if edge (i,j) is being repaired at time k, $\delta_{(ij),k} = 1$ and $\delta_{(ij),k'} = 0$, $\forall k' \neq k$. Similarly, if an edge (i,j) had never failed, I set $\delta_{(ij),0} = 1$ to keep into account that it must not be scheduled for repair. For shortness of notation I also define the decision matrix Δ_k, whose ij-th element corresponds to the decision $\delta_{(ij),k}$.

The recovery of a broken line (ij) requires r_{ij} recovery resources. At each recovery stage, R_k resources are available for recovery interventions. I assume to have a budget rollover, so that resources that have not been consumed until the end of stage $k-1$ are available at the k-th recovery stage, and summed up to the R_k

newly available.

Notice that in my model, the power grid graph at each stage k includes all the edges repaired according to the repair schedule performed until time k. Hence, I denote the power absorbed by load L_j at time k by $P^k_{L_j}(\Delta_1, \ldots, \Delta_k)$, calculated by means of the iterative solution of problem *Min-CFA* at stage k.

The maximum recovery (*Max-R*) optimization problem aims at maximizing the accumulative delivered power over K recovery stages. The *Max-R* recovery problem is formulated as follows:

$$\text{maximize} \quad \sum_{k=1}^{K} \sum_{L_j \in L} P^k_{L_j}(\Delta_1, \ldots, \Delta_k)$$

$$\text{s.t.} \quad \sum_{m=1}^{k} \sum_{(ij) \in E_p} \delta_{(ij),m} \cdot r_{ij} \leq \sum_{m=1}^{k} R_m, \;\; k = 1, ..., K \tag{3.8a}$$

$$\sum_{m=0}^{k} \delta_{(ij),m} \leq 1, \; \forall (ij) \in E_p, \;\; k - 1, \ldots, K \tag{3.8b}$$

$$\delta_{(ij),k} \in \{0, 1\}, \; \forall (ij) \in E_p, \;\; k = 1, ..., K \tag{3.8c}$$

where $\delta_{(ij),k}$ is the decision variable to repair $(ij) \in E^B_p$ at the k-th stage of the algorithm. Constraint 3.8a indicates that at stage k of the recovery schedule, R_k new recovery resources become available and resources rollover from the previous stages so that if some available resources are not used until the k-th stage of the recovery, they are still available in the following stages. Constraint 3.8b shows that each broken line can be repaired only in one stage of the recovery schedule.

Notice that *Max-R* is a combinatorial and nonlinear optimization problem. In fact, the objective function is the accumulative power flow measured at the loads in the K steps of execution of the algorithm. I underline that the total delivered power appearing in the objective function depends on the recovery decisions adopted at each stage of the recovery schedule. With $P^k_{L_j}(\Delta_1, \ldots, \Delta_k)$ I denote the power absorbed by load L_j when the recovery decisions given by the decision matrix Δ_m are made according to the schedule up to step $m = 1, \ldots, k$. Such an optimization problem is combinatorial and non-linear, in that it requires the solution of the *Min-CFA* optimization problem to find $\sum P^k_{L_j}(\Delta_1, \ldots, \Delta_k)$, since the set of working lines at stage k changes based on the current and previous decisions of the recovery schedule Δ_m, for $m = 1, \ldots, k$.

Note that in the recovery phase, I remove the generator's power reduction constraint and the generator and load's power increases gradually until all demand loads are all satisfied.

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

Proof of Theorem 3 I prove the NP-hardness of the *Max-R* problem showing that it generalizes the Knapsack problem. I 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 I show how I 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 I consider a single stage of the *Max-R* problem, I assume R resources are available to repair all disrupted lines $(ij) \in E_p^{B,t}$. I also assume that I have complete information about the disrupted lines. Let me 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. I also assume that the load L has a demand of at least the summation of all flows ($\sum_{i \in I} V_i$). I 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*, showing that *Max-R* is at least as difficult as the Knapsack problem.

There exists no polynomial-time solution for general instances of knapsack problem. Therefore, each stage of Max-R is NP-hard. I also 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 in the previous recovery stages. 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 3.4.0.1 which itself is a linear programming optimization.

As *Max-R* is NP-hard, and due to the broadness of the feasible region of the problem, which includes all possible permutations of repair interventions, in the following section I propose polynomial heuristic approaches to stop failure propagation and recover the network under uncertain failure localization.

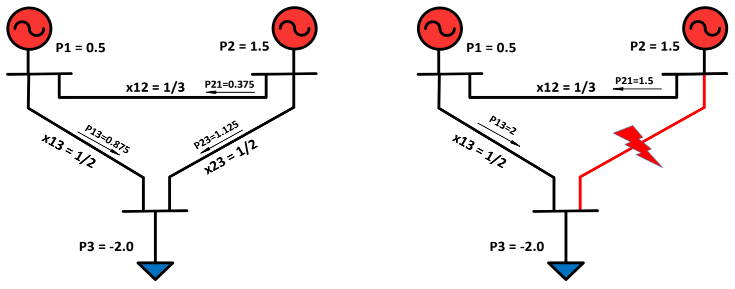
3.5 Heuristics for Cascade Mitigation and Recovery under Uncertain Knowledge

In this section, I first apply a linear algebraic approach to increase the incomplete knowledge of the phasor voltages as much as possible, and then I describe the consistent failure set approach to detect the status of grey lines. The grey lines, as described in Section 3.3 are the set of edges whose working status is unknown to the controller. Then, I describe two heuristic algorithms to solve *Max-R*. Inspired by the approach proposed in [45] that finds a progressive recovery schedule in a data communication network, I first propose a greedy approach with polynomial time complexity and then propose a polynomial time backward approach that solves a single stage of the problem and finds the solution for all stages in reverse recovery scheduling order.

3.5.1 Identifiability of Voltage Phasors

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

$$\begin{pmatrix} Y_{known} \\ Y_{unknown} \end{pmatrix} \times \begin{pmatrix} \theta_{known} \\ \theta_{unknown} \end{pmatrix} = \begin{pmatrix} P_{known} \\ P_{unknown} \end{pmatrix} \tag{3.9}$$



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

Figure 3.4: An example of a 3-bus network where active power and reactances are in pu.

Therefore, some of the unknown voltage phasors can be found by solving the following linear set of equations:

$$Y_{known} \times \begin{pmatrix} \theta_{known} \\ \theta_{unknown} \end{pmatrix} = P_{known} \tag{3.10}$$

Let $Null(Y_{known})$ denote the null space of Y_{known}, i.e., for any vector $v \in Null(Y_{known})$, I have $Y_{known} \cdot v = 0$ [12].

Theorem 4 *[73] For an arbitrary matrix Y_{known}, let $Null(Y_{known})$ represent the null space of Y_{known}. Voltage phasor θ_i is identifiable if and only if $\forall v \in Null(Y_{known})$ I have $v_i = 0$.*

Therefore, in order to find a set of identifiable voltage phasors, I can first compute the null space of Y_{known} and find all indexes with zero values in the null space. The value of the identifiable voltage phasors can be found by solving the linear system of equations (3.10).

3.5.2 Finding a Consistent Failure Set (*CFS*)

In this section, I first consider an illustrative example showing the impact of incomplete knowledge on the extent of the failure propagation and uncontrollability. I then propose a Consistent Failure Set (*CFS*) algorithm to cope with lack of knowledge.

Example 1 Consider the network given in Figure 3.4, using the DC power flow model to calculate the power flows in the lines, where the reference angle is $\theta_1 = 0$, I have:

$$\begin{pmatrix} \theta_2^0 \\ \theta_3^0 \end{pmatrix} = \begin{pmatrix} 5 & -2 \\ -2 & 4 \end{pmatrix}^{-1} \begin{pmatrix} 1.5 \\ -2 \end{pmatrix} = \begin{pmatrix} 0.125 \\ -0.4375 \end{pmatrix} \tag{3.11}$$

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

$$F_{12}^0 = \frac{\theta_{12}^0}{x_{12}} = 3 \times (0 - 0.125) = -0.3750, \tag{3.12}$$

$$F_{13}^0 = \frac{\theta_{13}^0}{x_{13}} = 2 \times (0 - (-0.4375)) = 0.875, \tag{3.13}$$

$$F_{23}^0 = \frac{\theta_{23}^0}{x_{23}} = 2 \times (0.125 - (-0.4375)) = 1.125. \tag{3.14}$$

If the power line 23 gets disrupted as in Figure 3.4b, the power redistributes 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 I know the exact location of the failure, the RAS/SPS 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.

In order to have a correct damage assessment, and solve the uncertainty in the grey area I propose the following *CFS* algorithm. I assume that there is no local load shedding. Hence, the powers at the generators and loads are only controlled through the central controller unit, so the power generated/consumed in each generator/load or junction is known.

To explain my algorithm, I first consider the nodes that have the smallest number of grey links.

Lemma 1 *In the power grid graph G_p, if there exists a node $v \in V_p$ which has only one grey incident link $(v, w) \in E_p^t$, the exact status of the grey link (v, w) is* *identifiable.*

Proof of Lemma 1 The exact status of a single grey link incident to a node v can be determined using the power flow Equation 3.2. In fact, under the assumption of only centralized load shedding, the power generated/consumed at node v is known from the previous stage. Hence, the power flowing through the grey line can be found by solving the power flow Equation 3.2 where the only unknown variable is F^t_{vw}.

Lemma 2 *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 $O(|E^{U,t}_p|)$.*

Proof of Lemma 2 If the grey area does not contain any cycles, it forms a tree. Hence there exists at least one node that has only one grey incident link whose statuse can be identified according to lemma 1. This procedure can be repeated to find the status of all grey links in $O(|E^{U,t}_p|)$.

Therefore, I can find the state of the network for all grey links, which are not inside a cycle. In the presence of a grey cycle, the *CFS* algorithm breaks the cycle by selecting one arbitrary link within the cycle, and considering the two potential statuses, broken or working. *CFS* then finds one or multiple consistent failure sets, namely sets of status assignments to each grey link of the cycle, which are consistent with the available observations.

If the consistent failure set algorithm finds multiple solutions, *CFS* provides local inspection to determine the actual failure set.

The *CFS* algorithm is described in Algorithm 2. In more details, for the case study of a graph G_p which has one or multiple cycles in its grey area, *CFS* starts by considering a case in which there are no grey link cycles (**line 1**), finding the status of grey links by considering them iteratively, starting from the nodes with only one incident grey link, according to the function `CFS-Cycle-Free`, described in Algorithm 3.

If all nodes have at least two grey links in the graph, i.e. there exists a cycle in the grey area (**line 2**), *CFS* picks an arbitrary edge within a cycle (**line 3**).

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 according to a decision tree, henceforth generating 2^C possible link status combinations. *CFS* then uses Algorithm 3 to determine which combination is consistent with the observation (**line 4**) and provides the corresponding status of the remaining links of the cycles.

In cases where there exists multiple consistent failure sets, *CFS* requires a local inspection of a link which appears with a different status in any two solutions to determine which solution is consistent with the result of the local inspection.

Observation 1 *Assuming the grey area becomes a tree by removing C edges, CFS algorithm runs in $O(2^C|E_p^{U,t}|)$.*

Proof of Observation 1 The status of all grey links which are not within a cycle in $O(|E_p^{U,t}|)$ is found according to Lemma 2.

If there are cycles of grey links, I break the cycles by selecting C links. By making an assumption on the status of each of these C links *CFS* builds a decision tree, where at each node it assumes whether each link is either working or broken. Such a decision tree will have 2^C leaves corresponding to different failure sets whose consistency is validated separately by means of Algorithm 2, which will also provide the state of the remaining grey links in the broken cycles, in the case of consistency.

Note that the consistent failure set is not unique and sometimes I may end up having multiple failure sets, which are all consistent with my partial knowledge.

Example 2 Figure 3.5 shows an example of a network with 6 grey links and shows different steps of the *CFS* algorithm. In the first step, I evaluate the status of all grey links, which are the unique grey incident links of a node and therefore are not part of a cycle.

In the second step, I evaluate the status of grey links forming a cycle. For link (23) I build a decision tree to remove the existing cycle. The decision tree rooted in the link (23) will have two branches, corresponding to the two potential different state of link (23), working or not working. Assuming edge (23) $\in E_p^{B,t}$ was broken, I do not find a consistent feasible solution, which lead me to evaluate the other branch, assuming that (23) is working, namely (23) $\in E_p^{W,t}$. The last graph shows a consistent failure set of broken and working edges. In this example, I derived only one consistent failure set. If I had multiple consistent failure sets, I would have performed a local inspection of the edges whose failure status is different in the possible consistent solutions, and picked the solution consistent with the local

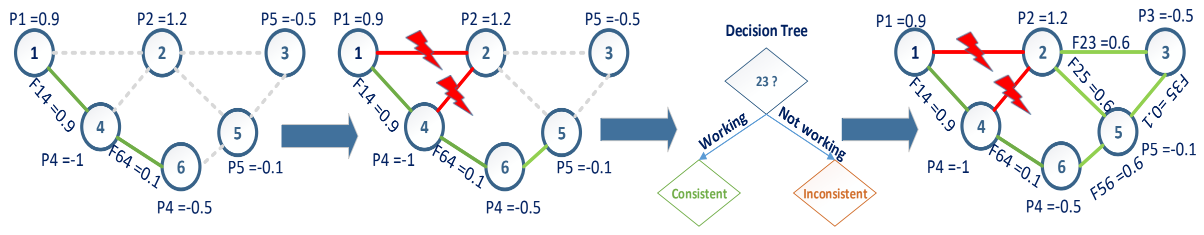


Figure 3.5: An example of a 6-bus network with 6 grey edges and different steps of *CFS* algorithm.

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 failed or working.

```
1: $C =$ Number of edges in $E_p^{U,t}$ that need to be removed to make the grey area cycle-free
       if $C = 0$ then
2:
       run CFS-Cycle-Free($E_p^{U,t}, G_p, P_{G_i}, P_{L_i}$). if $C > 0$ then
3:
       else
          pick an edge at each cycle to generate a cycle-free grey area
4: 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
       return $E_p^{B,t}, E_p^{W,t}$
```

inspection.

Remark 2 (Discussion on the number of grey edges.) *Table 3.2 shows the average number of grey edges which are part of a cycle in the graph of the Italian power grid network [44, 68, 74] 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, I only need a maximum of 10% local inspection of the grey edges.*

Algorithm 3: CFS-Cycle-Free

```
1  Function CFS-Cycle-Free ($E_p^{U,t}, G_p, P_{G_i}, P_{L_i}$)
2      $greys = argmin|(n_i j) \in E_p^{U,t}|$;
3      while $greys = 1$ do
4          Select a node $i \in V_p^t$ with one grey neighbor
            $greys = argmin|(ij) \in E_p^{U,t}|$ ;
5          detect whether $(ij) \in E_p^{U,t}$ is working or not using equation 3.2.;
6          if there exists no solution from equation 3.2 then
7              return INCONSISTENT;
8              break ;
9          if $(ij) \in E_p^{U,t}$ is working then
10             $E_p^{W,t} = E_p^{W,t} \cup (ij)$ and $E_p^{U,t} = E_p^{U,t} \setminus (ij)$ ;
11         else
12             $E_p^{B,t} = E_p^{B,t} \cup (ij)$ and $E_p^{U,t} = E_p^{U,t} \setminus (ij)$;
13     return CONSISTENT, $E_p^{B,t}, E_p^{W,t}$ ;
```

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

3.5.3 *Max-R-Greedy*

In this paragraph I introduce the baseline heuristic *Max-R-Greedy* to solve the *Max-R* problem. I recall that the objective function of the problem *Max-R* is the total accumulative flow in K stages of recovery. *Max-R-Greedy* greedily selects the links to repair based on the marginal value of their contribution to the accumulative

Algorithm 4: *Max-R-Greedy* 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 $\delta_{(ij)^*,k}$
```

flow.

At each stage k, *Max-R-Greedy* repairs the transmission lines that add the maximum to the total delivered power over the required resource, i.e. it repairs the line $\arg\max_{(ij)\in E_p^{B,k}}(F_{ij}/r_{ij})$ among the broken ones, until the total available resources for stage k are used. Algorithm 4 shows different stages of the *Max-R-Greedy* algorithm.

More in details, the algorithm works iterating repair interventions through stages. At each stage k, it first updates the current value of the rollover budget (**line 2**). Then, until the available budget allows, it selects a new broken link $(i, j)^*$ for recovery, based on the marginal contribution to the accumulative flow, with respect to the repair cost (**line 3**). Finally it schedules $(i, j)^*$ for recovery at stage k (**line 4**) and updates the available recovery resources accordingly (**line 5**).

3.5.4 *Max-R-Backward*

As an alternative heuristic to compute a more accurate solution of the *Max-R* problem, I use *Max-R-Backward*. The algorithm finds the recovery schedule in K stages as a list of K sets $\mathcal{S}_k,\ k = 1, \ldots, K$, of links to be repaired up to stage k. The algorithms works backward from stage K to stage 1, by considering a decreasing budget at each stage. It starts by solving a version of the problem which assumes $R = R_1 + ... + R_K$ resources are available (**line 3**). The solution of this

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 $Y^t$, 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: $\mathcal{S}_{K+1} = E_p^B$ for $k = K$ downto $k = 1$ do
3:
       $R = \sum_{m=1}^{k} R_m$
4: $\mathcal{S}_k = \mathcal{S}_{k+1}$ while $\sum_{(ij) \in \mathcal{S}_k} r_{ij} > R$ 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: $\mathcal{S}_k = \mathcal{S}_k \setminus (ij)^*$
8: solve DC power flow model to find $F_{ij}$, assuming $(ij) \in \mathcal{S}_k$ are working.
   return $\delta_{(ij)^*,k}$
```

execution of the algorithm produces a list of edges to be recovered from the start until stage K, named $\mathcal{S}_K$. Initially, $\mathcal{S}_K$ is the entire set of broken links E_p^B. In order to calculate the recovery schedule for each stage, the algorithm proceeds by selecting from $\mathcal{S}_K$ all the lines that exceed the budget available for the first $K-1$ stages, and which contribute the minimum marginal value of flow over cost (**line 5**) . The selected lines will be scheduled for recovery at stage K (**line 6**), and will be excluded from the recovery schedule of any previous stage (**line 7**). Before the end of each stage, the algorithm must solve the DC power flow problem taking account of the scheduled repairs, to update the values of the flows F_{ij}, for all the links of E_p (**line 8**).

This procedure repeats until the repair schedule of all stages is found.

3.6 Evaluation

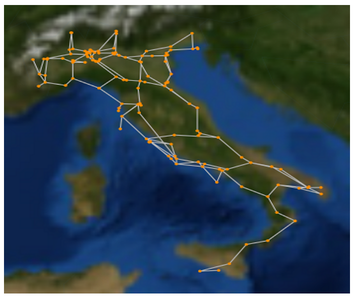
In this section, I perform an experimental evaluation of my algorithms in a real network setting. I consider the Italian power grid network, called HVIET, shown in Figure 3.6a consisting of 310 nodes, 113 generators and 97 demand loads. The network has 361 power lines. For the communication network I use the GARR network, shown in Figure 3.6b consisting of 39 nodes and 50 edges [68, 74]. Transmission lines in the power grid are monitored by several sensors deployed nearby. Based on the interdependency model, the aggregated data are then sent to the closest/random communication node and to the control center. In addition, the control commands for power adjustment are sent to the closest/random communication node. Therefore, each node in the communication network might monitor and control several lines and nodes in the power grid. In my evaluation, I assume the Italian power grid network to be purely inductive (lossless) with zero reactive injection, so that the DC power flow is actually accurate. I also note that, since the DC power flow model is an approximation to the AC power flow, applying my model to a real coupled system, can result in a lower performance. I implement my cascade prevention and recovery algorithms in Python and used the Gurobi optimization toolkit [42], on a 120-core, 2.5 GHz, 4TB RAM cluster.

In my experiments I vary the interdependency model and I randomize the results running 10 different trials with randomized selection of failed transmission lines.

Summary of observations. The key observations are as follows. First, I observed that the random interdependency model has a larger impact on the number of disrupted/uncontrollable nodes in the two networks compared to a one-way and a location-based interdependency model. This is due to the fact that in one-way and location-based interdependency model, the cascade is limited to the geographical interdependent regions in the two network while in a random interdependency model the cascade spreads from any part of the network to the other parts.

Second, I observed that my cascade prevention approach could still provide service, though potentially degraded, differently from the case without countermeasures.

Finally, I observed that the backward recovery approach performs better than the greedy approach. This is due to the fact that the greedy approach does not consider the correlation between different steps of the recovery approach and makes a repair decision at each stage independently.


(a) HVIET.

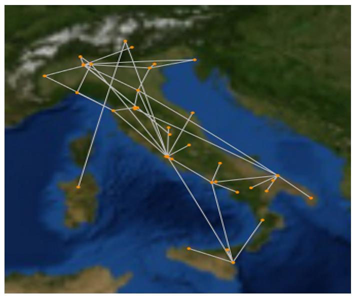

(b) GARR.

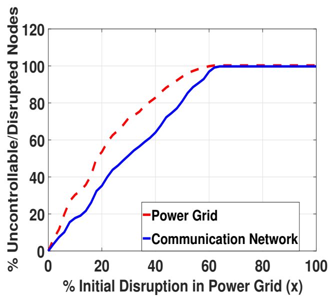
Figure 3.6: a) The Italian high-voltage (380 kV) transmission grid (HVIET), and b) its communication network (GARR).

3.6.1 Impact of Interdependency

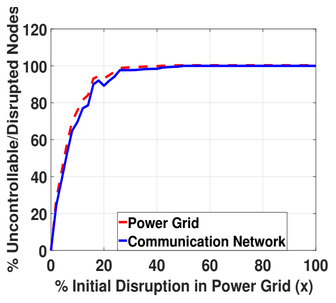
I first evaluate the impact of different interdependency models on the extent of failure propagation within the power grid and the communication network.

I assume the initial failures are in the power grid. In this experiment I gradually increase the percentage x of randomly failed lines in the power grid and I observe how the number of uncontrollable power grid nodes and disrupted communication node varies accordingly. In the first scenario, I connect the HVIET and GARR networks following the location-based interdependency model described in Section 3.3.2. Figure 3.7a shows the percentage of uncontrollable nodes in the power grid and disrupted nodes in the communication network which lose power due to the failures in the power grid, when the propagation stops naturally. I observe that in a location-based interdependency model the failure does not spread much and the percentage of failed power grid nodes and communication nodes is linear with respect to the initial failures in the power grid.

In the second scenario, I consider the random interdependency model. Figure 3.7b shows the simulation results for this scenario. As shown, compared to location-based interdependency model, in a random interdependency model, the failures spread more in both the power grid and the communication network. This is due to the fact that in a random interdependency model, the failures are not limited to geographical interdependent regions and can spread from any part of the network to the other. Hence, I consider this model as a stress-test for my algorithms.



(a) Location-based.



(b) Random.

Figure 3.7: a) Location-based and b) Random interdependency model in the Italian power grid (HVIET) and its communication network (GARR).

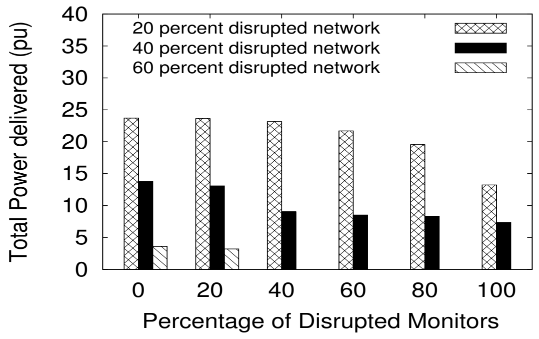


Figure 3.8: Total delivered power (pu) versus the percentage of failures on the monitoring nodes in the Italian power grid network.

3.6.2 Impact of Incomplete Knowledge

In this section, I investigate the impact of incomplete knowledge of the exact location of failures. Initially, x% of the communication nodes get disrupted. I consider the one-way interdependency model where the consequences of failures in the communication network are lack of information and loss of controllability in the power grid. Figure 3.8 shows the simulation results of this experiments. When 100% of the communication network and 20% of the power grid is disrupted, the total delivered power can drop by 10.48 power units (pu). Assuming the maximum unitary profit of 26.6 €$/MW$ according to [75], the total profit loss, due to uncertainty of failure location 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.

3.6.3 Preventing the Cascade (*Min-CFA*)

In this paragraph I evaluate the performance of our cascade prevention approach, namely the *Min-CFA* algorithm, with the case in which no cascade countermeasure is available.

I consider the one-way interdependency model described in Section 3.3.2. I assume the communication network gets power from an external source in case of a failure in the power grid. The disruption in the communication network is observed as lack of knowledge and uncontrollability in the power grid.

The performance metric considered in this experiment is the total delivered demand power. Similar to [54], I assume all loads have the same priority and give a high penalty for not being able to satisfy the demand. I 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 I set the disruption percentage of the power grid to $x = 60\%$ and run *Min-CFA* to find the optimal flow assignment. Figure 3.9 shows the total delivered power during different time steps of the algorithms with *Min-CFA* cascade prevention and without it. As shown in the figure, *Min-CFA* can save 54% of the total power that would be delivered if the power grid were not disrupted. On the other hand, if I 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.

In the next set of simulations, I use a continuous cascade prevention, meaning that the decision variable, $P^t_{L_j}$ in Equation 3.4 can decrease continuously. Then, I consider a discrete cascade prevention scenario, where the decision variable, $P^t_{L_j}$ in Equation 3.4 can either be equal to each load's demand power which should be satisfied or set to zero (i.e., loads are turned off); and finally, I 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.

In this experiment I gradually increase the percentage x of randomly failed lines in the power grid and observe the total amount of load served. Figure 3.10 shows the simulation results for the three cases. As shown, the continuous cascade prevention approach saves more power compared to the discrete power optimization and to the case without cascade prevention.

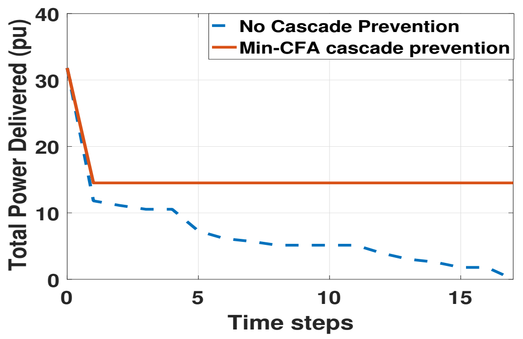


Figure 3.9: Total delivered power (pu) during time when *Min-CFA* cascade prevention algorithm is used and without any cascade prevention in the Italian power grid network.

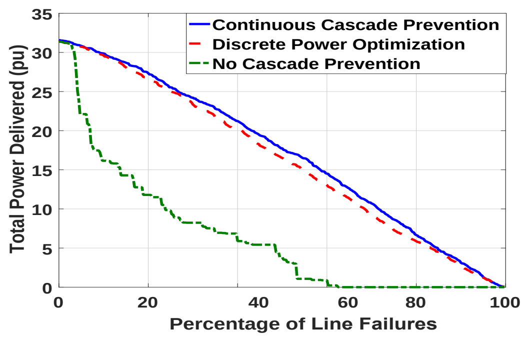


Figure 3.10: Total delivered power (pu) versus the percentage of line failures in the Italian power grid.

Notice that in absence of cascade prevention measures, an initial failure that disrupts more than 60% of the power grid lines is sufficient to make a black out of the entire system, due to a full propagation of the failure.

3.6.4 Recovery Phase (*Max-R*)

In the next set of experiments, I compare the recovery performance of the proposed heuristics (*Max-R-Greedy* and *Max-R-Backward*). Figure 3.11 shows the total delivered power flow over different stages of progressive recovery intervention, when using the two algorithms. As shown, the greedy approach 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

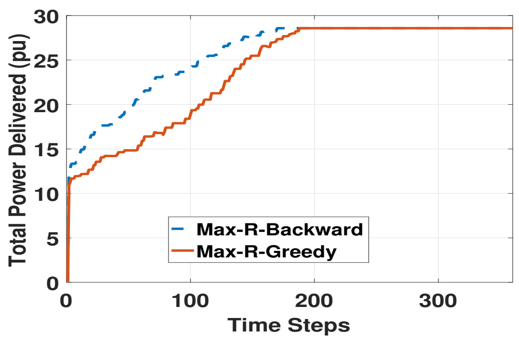


Figure 3.11: Total delivered power (pu) flow over time for *Max-R-Backward* and *Max-R-Greedy* in the Italian power grid.

Table 3.3: Normalized accumulative delivered power for *Max-R-Backward* and *Max-R-Greedy* approaches.

Recovery Resources	*Max-R-Backward*	*Max-R-Greedy*
1	0.8062	0.6791
2	0.9012	0.8377
6	0.9645	0.9435

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-Greedy* approach with larger total area behind the curve in Figure 3.11. I next increase the number of resources at each stage and study the normalized accumulative delivered power for the two recovery approaches. Table 3.3 shows the results of this scenario.

3.7 Conclusion

This chapter studies the problem of mitigating propagating failures and performing progressive recovery interventions to restore the functionality of an interdependent power grid and communication network, under incomplete localization of failures. I formulate an optimization problem to stop the cascading failures and, due to high complexity of the recovery problem, I propose two heuristic approaches (i) a baseline greedy and (ii) a backward heuristic, to restore the power grid functionality.

By means of extensive simulations, I show that since the backward algorithm takes account of the combined impact of repaired component, it significantly outperforms the baseline recovery algorithm in terms of accumulative delivered power.

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 me to study the optimal selection of monitoring paths to balance identifiability and probing cost. Assuming additive link metrics (e.g., delays), I 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, I provide an optimal algorithm for (4) and a $(1-1/e)$-approximation algorithm for (2). I 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). My 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 my solutions can reduce the total probing cost by an order of magnitude while achieving the same monitoring performance [12].

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 [76, 77] 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 [78–82] 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, I show that by carefully selecting the probing paths, I can significantly reduce the amount of probing traffic while achieving the same monitoring performance.

To this end, I 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, I make the following contributions:

1. Based on an existing algorithm that computes all the minimal sets of paths to identify each identifiable link, I convert (1) and (3) to problems similar to the max-coverage problem [83] and the set-cover problem [84], respectively. The conversion allows me to apply the greedy heuristic to these problems. I also propose an iterative branch-and-bound algorithm that treats my problems as integer linear programs (ILPs), and decomposes each problem into smaller meaningful sub-problems to exploit parallelism on a multi-core machine. Using my iterative branch-and-bound algorithm, I can configure the trade-off between the execution time and the optimality gap of the solution.

2. Using techniques from matroid optimization, I 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. I 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. My 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, my solutions can reduce the probing cost by an order of magnitude while achieving the same monitoring performance.

I first discuss the background and motivation behind this work in Section 4.2. In section 4.3, I formulate the four optimization problems. Section 4.4 contains my algorithms and their performance analysis. Section 4.5 shows my evaluation methodology and results. Finally, Section 4.6 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 [73, 85–92]. 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 [85, 86]. 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 [87–89]. In contrast to [87–89], I assume that routes cannot be controlled, as is usually the case in IP networks; in contrast to [85, 86] that focus on the offline cost for deploying monitors, I focus on the online cost for sending probes (i.e., the probing cost).

In the context of overlay networks, Chen et al. [73] 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 [93]. These approaches differ from mine in that they focus on end-to-end performance, while I focus on identifying the performance of individual links.

Zheng et al. [94] introduced a problem similar to my 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. My formulation differs from theirs in that I allow general probing costs for the paths, and do not require coverage of all the links. These differences allow me to model paths with heterogeneous probing costs and further reduce the total cost without losing identifiability. More importantly, the solution in [94] requires the calculation of all the irreducible path sets to identify each of the identifiable

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

links, which has an exponential complexity. In contrast, I show that using rank as a proxy of identifiability gives an efficient solution that provides tight upper/lower bounds on the optimal solution (Theorem 7).

4.2.2 Motivation

I 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. I consider three networks from the Internet Topology Zoo [1, 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), I 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, I describe the 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 I can select from. In my model, I assume IP packets from a source node s to a destination node t are being forwarded using a pre-determined routing algorithm. My formulation and solutions support arbitrary routing algorithms, and the specific algorithm used for evaluation will be specified later (see Section 4.5). I 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 5.1 summarizes the notation used in my formulation.

4.3.2 Measures of Monitoring Performance

I use **identifiability** and **rank** functions to measure the monitoring performance of my 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 3 *[73] 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 $\forall n \in N$ I have $n_i = 0$.*

Therefore, to find the set of identifiable links, $L_I \in L$, I 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 my 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 pre-defined 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, I 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. I 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 5 *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 of Theorem 5 I 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:

$$rref(A_{R,*}) = \left[\begin{array}{c|c} I_{k\times k} & 0_{k\times(|L|-k)} \\ \hline 0_{(n-k)\times k} & M_{(n-k)\times(|L|-k)} \end{array} \right] \tag{4.2}$$

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:

$$rank(A_{R,*}) = k + rank(M) \tag{4.3}$$

It is clear that:

$$\begin{aligned} rank(A_{R,L_1}) &= k, \\ rank(A_{R,L\setminus L_1}) &= rank(M) \end{aligned} \tag{4.4}$$

Therefore,

$$\begin{aligned} rank(A_{R,*}) = k + rank(M) = \\ rank(A_{R,L_1}) + rank(A_{R,L\setminus L_1}) \end{aligned} \tag{4.5}$$

Next, I 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:

$$rref(A_{R,*}) = \left[\begin{array}{c|c} I_{k\times k} & B_{k\times(|L|-k)} \\ \hline C_{(n-k)\times k} & M_{(n-k)\times(|L|-k)} \end{array}\right] \tag{4.6}$$

I 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, I can make them zero by using a sequence of elementary row operations. Note that (4.1) implies that

$$\begin{aligned} rank(rref(A_{R,*})) = rank(&\begin{bmatrix} I_{k\times k} \\ 0_{(n-k)\times k} \end{bmatrix})+ \\ rank(&\begin{bmatrix} B_{k\times(|L|-k)} \\ M_{(n-k)\times(|L|-k)} \end{bmatrix}) \end{aligned} \tag{4.7}$$

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

$$\begin{aligned} rref(A_{R,*}) &= \left[\begin{array}{c|c} I_{k\times k} & B_{k\times(|L|-k)} \\ \hline 0_{(n-k)\times k} & M_{(n-k)\times(|L|-k)} \end{array}\right] \\ &= \left[\begin{array}{c|ccc} I_{k\times k} & b_1 & \dots & b_{|L|-k} \\ \hline 0_{(n-k)\times k} & m_1 & \dots & m_{|L|-k} \end{array}\right], \end{aligned} \tag{4.8}$$

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 $\begin{bmatrix} I_{k\times k} \\ 0_{(n-k)\times(k)} \end{bmatrix}$. Also let $[q_1, q_2, ..., q_{|L|-k}]$ be the columns of $\begin{bmatrix} B \\ M \end{bmatrix}$, where $q_i = \begin{bmatrix} b_i \\ m_i \end{bmatrix}$.

I 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 4 *I claim that*

$$\delta_i = \delta_i' \quad for i = 1, ..., |L| - k, \tag{4.9}$$

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, I note that the left hand side (LHS) and right hand side of Equation (4.7) are as follows:

$$\text{LHS of (4.7):} \quad rank(rref(A_{R,*})) = k + \sum_{i=1}^{|L|-k} \delta_i \tag{4.10}$$

$$\text{RHS of (4.7):} \quad rank(rref(A_{R,*})) = k + \sum_{i=1}^{|L|-k} \delta_i' \tag{4.11}$$

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 $\delta_i < \delta_i'$, (4.10) will be smaller than (4.11), violating Equation (4.7). Thus, $\delta_i = \delta_i', \forall i = 1, ..., |L| - k$. Using the above claim, I 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 (4.9), $q_1 = 0$ 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 (4.8) 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 4, 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 (4.6), 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. I therefore, conclude that the necessary and sufficient condition for $P_R \subseteq P$ to identify a set of links L_1 is $rank(A_{R,*}) = |L_1| + rank(A_{R,L \setminus L_1})$.

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 I have the following routing matrix.

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

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

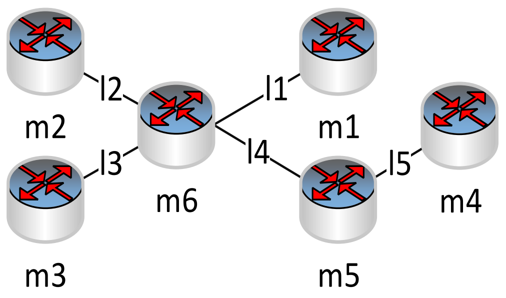


Figure 4.1: A simple network example with 5 links and 4 monitors $\{m_1, \ldots, m_4\}$. Candidate paths: r_{m_1,m_2}, r_{m_1,m_3}, r_{m_1,m_4}, r_{m_2,m_3}, r_{m_2,m_4}, r_{m_3,m_4}.

routing matrix A_R satisfies Theorem 5.

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

Meanwhile, it is also clear that probing these paths suffices to identify l_1, l_2 and l_3. I can solve the identifiable links using Gaussian elimination, where the reduced row echelon form $(rref(A))$ is

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

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 5 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:

$$\begin{aligned} \text{Maximize} \quad & |I(P_R)| & (4.12a) \\ \text{subject to} \sum_{r \in P_R} & k_r \leqslant K, & (4.12b) \\ & P_R \subseteq P, & (4.12c) \end{aligned}$$

where k_r is the probing cost of path r. As a concrete example, I 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, I re-write it as an integer linear programming (*ILP*). The basis of my formulation is the notion of *minimal solutions* (simply called *solutions* in [94]). 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.

I 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 [94]. 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, I index the solutions in $\mathcal{P}$ and denote by S the set of solution indices.

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

$$\begin{aligned}
&\underset{X_l, Y_r, Z_s}{\text{Maximize}} \quad \sum_{l \in L} X_l && && \text{(4.13a)}\\
&\text{subject to } X_l \leqslant \sum_{s: l \in I(P_s)} Z_s, && \forall l \in L, && \text{(4.13b)}\\
&\sum_{r \in P} Y_r \cdot k_r \leqslant K, && && \text{(4.13c)}\\
&Z_s \leqslant Y_r, && \forall s \in S, r \in P_s, && \text{(4.13d)}\\
&X_l, Y_r, Z_s \in \{0,1\}, && \forall l \in L, r \in P, s \in S. && \text{(4.13e)}
\end{aligned}$$

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

Lemma 5 *The ILP optimization problem can be relaxed over the integer variables X_l and Z_s and still gives an optimal integer solution.*

Proof of Lemma 5 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 (4.13-d), it implies that if $Z_s > 0$, I must have $Y_r = 1 \quad \forall r \in P_s$. Therefore, I 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.

Remark: While the problem can be relaxed over X_l and Z_s, finding all minimal solutions has an exponential complexity. Furthermore, similar to [94], optimizing Y_r's is hard to solve. Therefore, I 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, I 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:

$$\begin{aligned} \text{Maximize} \quad & rank(A_R) && (4.14a) \\ \text{subject to} \quad & \sum_{r \in P_R} k_r \leq K, && (4.14b) \\ & P_R \subseteq P. && (4.14c) \end{aligned}$$

The rank function has an important property that makes its maximization easy to solve. To this end, I 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 $P_a, P_b \subseteq P$, I have

$$f(P_a \cup P_b) + f(P_a \cap P_b) \leqslant f(P_a) + f(P_b). \tag{4.15}$$

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 I add elements to P) and submodular, then there is a generic greedy algorithm in [83] 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 6 *[95] 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

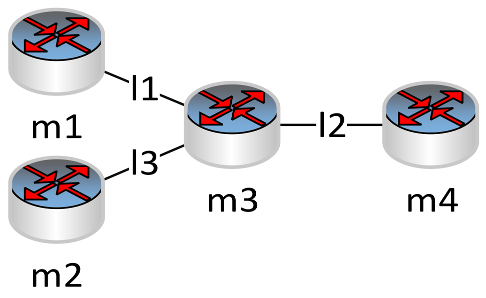


Figure 4.2: An example that shows identifiability is not a submodular or supermodular function.

$I(P_a \cap P_b) = \emptyset$. 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. [94] 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 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 [94] also requires the coverage of non-identifiable links, my 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:

$$\begin{aligned} \text{Minimize} \quad & \sum_{r \in P_R} k_r && (4.16a) \\ \text{subject to} \quad & |I(P_R)| = |I(P)|, && (4.16b) \\ & P_R \subseteq P. && (4.16c) \end{aligned}$$

ILP formulation for Min-Cost-IL: Similar to (4.13), I re-write Min-Cost-IL as an integer linear programming (*ILP*) as follows:

$$\begin{aligned} \underset{Y_r, Z_s}{\text{Minimize}} \quad & \sum_{r \in P} Y_r k_r && && (4.17a) \\ \text{subject to} \quad & 1 \leqslant \sum_{s: l \in I(P_s)} Z_s, && l \in I(P_s), && (4.17b) \\ & Z_s \leqslant Y_r, && \forall s \in S, r \in P_s, && (4.17c) \\ & Y_r, Z_s \in \{0, 1\}, && \forall r \in P, s \in S. && (4.17d) \end{aligned}$$

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, I 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. I 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. I aim to select a subset of paths, $P_R \subseteq P$ such that the rank of both matrices be the same.

$$\begin{aligned} \text{Minimize} \quad & \sum_{r=1}^{|M|(|M|-1)/2} k_r Y_r && (4.18a) \\ \text{subject to} \quad & rank(A_R) = rank(A) && (4.18b) \end{aligned}$$

$$Y_r \in \{0,1\} \tag{4.18c}$$

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, I give different algorithms for the four optimization problems. I propose a greedy heuristic and an iterative branch-and-bound algorithm for the *Max-Cost-IL* and the *Min-Cost-IL* problems. I 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

I 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 [94], I 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:

$$\frac{\text{New Identified Links in } S_i}{\text{Cost of New Paths in } S_i}, \tag{4.19}$$

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 [94] but with two key differences. Unlike [94] that uses uniform cost for all selected paths, I allow an arbitrary cost for each path. Furthermore, unlike [94] that requires

```
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 $IL = \{l \in E\}$
1 $IL = \emptyset$;
2 $P_R = \emptyset$;
3 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
4     Select an un-selected set $S_i = argmax \frac{\text{New Identified Links in } S_i}{\text{Cost of New Paths in } S_i}$ ;
5     for $i = 1$ to New Identified Links($S_i$)    $IL = IL \cup l \quad l \in I(S_i)$;
6     for $r_j \in S_i$
7     $P_R = P_R \cup \{r_j\}$;
8 return $IL$ and $P_R$
```

the selected paths to cover all links, I only require the paths to identify all the identifiable links.

I use a second greedy-based approach that I do not 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 I 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 (4.13, 4.17) allows me to apply general ILP solvers to *Max-IL-Cost* and *Min-Cost-IL*. Specifically, I use an iterative branch-and-bound algorithm [40] to achieve a configurable trade-off between complexity and optimality. For brevity, I explain the algorithm for maximization and minimization works 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 I have the optimal solution. Otherwise, I pick a fractional variable, Y_r, and make two branches by creating one more constraint in the optimization: $Y_r = 0$ or $Y_r = 1$. I 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 I 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. I stop branching once the gap between the incumbent's objective function (LB) and the current upper bound is smaller than a threshold (Gap), or I 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 I have

$$\frac{LB}{OPT} \geq \frac{LB}{LB + Gap}. \tag{4.20}$$

In the second case, there is no guarantee on the approximation ratio but I 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 I have

$$\frac{UB}{OPT} \leq \frac{UB}{UB - Gap}. \tag{4.21}$$

The advantage of this algorithm is its flexibility. I 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, I 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. I show that in terms of the rank objective, Greedy-Min-

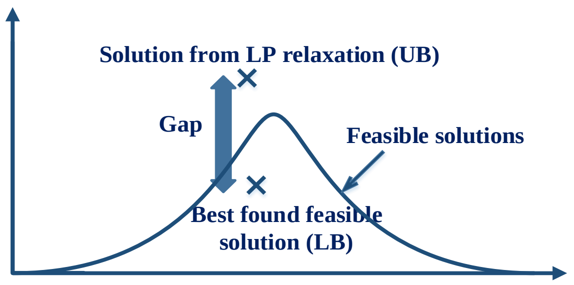

Solution from LP relaxation (UB)
Gap
Feasible solutions
Best found feasible solution (LB)

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

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.

I first review the definition and properties of matroids [70] 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 [96, 97]:

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

I 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.

I am 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 I am 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

I now consider one of the interesting properties of matroids. I 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 [98] 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 6 *[70] 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 [99], 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 7 *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 of Lemma 7 A path set identifies all the identifiable links if and only if it satisfies the conditions of Theorem 5, 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 6.

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 7 *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, I have:

$$K_{LB} \leq K^{opt} \leq K_{UB} \tag{4.22}$$

Proof of Theorem 7 The lower bound is obvious, since I 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 5. 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}.

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 I have more constraint for selecting the first $|L_I|$ paths for A_{*,L_I} than A, I always have

$$k'_1 + k'_2 + ... + k'_{|L_I|} \leq k_1 + k_2 + ... + k_{|L_I|} \tag{4.23}$$

Therefore,

$$\begin{aligned} K_{UB} - K_{LB} &\leq k'_{|L_I|} + ... + k'_{rank(A)} \\ &\leq (rank(A) - |L_I|) * k'_{rank(A)} \end{aligned} \tag{4.24}$$

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, I 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, I 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, I need to select all paths to

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.
1  $P_R^* = \emptyset$;
2  $TotalCost = 0$;
3  sort $P$ in increasing order of cost ;
4  forall $r_i \in P$ do
5      $IncreaseRank_{r_i} = rank(P_R^* \cup \{r_i\}) - rank(P_R^*)$ ;
6      if $IncreaseRank_{r_i} > 0$ then
7          $P_R^* = P_R^* \cup \{r_i\}$ ;
8          $TotalCost = TotalCost + k_i$ ;
9      if $rank(P_R^*) \geq rank(P)$ or $|I(P_R^*)| = |L_I|$ then
10         break ;
11 return $P_R^*, TotalCost$
```

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, I 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 6), I can leverage an existing result for budgeted submodular maximization.

Theorem 8 *[83] 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 [99], 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 9 *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:*

$$I_R \leq I^{opt} \leq min\{rank(A_{R,*}) \cdot \frac{e}{e-1}, |L_I|\}, \tag{4.25}$$

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

Proof of Theorem 9 The lower bound $I_R \leq I^{opt}$ trivially holds due to the optimality of I^{opt}. For the upper bound, I denote by R^* the set of path indices in the optimal solution of Max-IL-Cost. Then by Theorem 5, $I^{opt} \leq rank(A_{R^*,*})$. Meanwhile, by Theorem 8, I have that

$$rank(A_{R^*,*}) \leq rank^{opt} \leq rank(A_{R,*}) \cdot \frac{e}{e-1}, \tag{4.26}$$

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.

4.5 Evaluation

In this section, I consider several scenarios to compare the probing cost of our proposed algorithms compared to the case where I use all feasible probes or the optimal (OPT) brute-force solution. For each scenario, I randomize the results by running 10 different trials, where I vary the random selection of monitors from

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

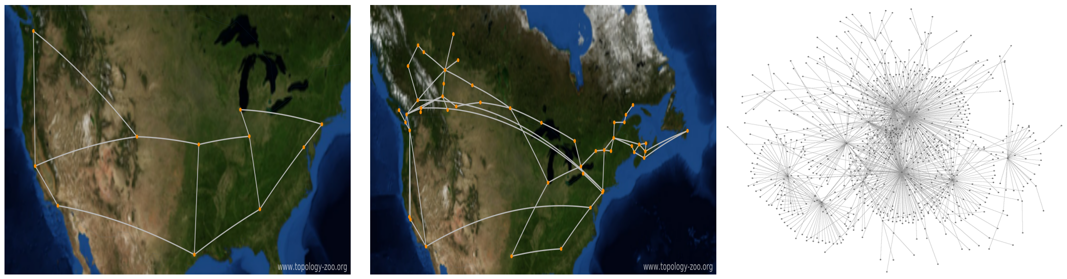
```
Data: A set of uncontrollable paths $P = \{r_1, ..., r_{|M|(|M|-1)/2}\}$, a set of cost
      functions for each path $Cost = \{k_1, ..., k_{|M|(|M|-1)/2}\}$, Limit on the
      probing cost $K$.
Result: A subset of paths $P_{Max} \subseteq P$ that maximizes $rank(P_{Max})$ subject to a
        limited monitoring cost $K$
1  $P_{Max2} = \emptyset$;
2  $TotalCost = 0$;
3  $P_{Max1} = argmax\{rank(P_x) : P_x \subseteq P, |P_x| \leq 3, c(P_x) \leq K\}$;
4  forall $P_g \subseteq P, |P_g| = 3, c(P_g) \leq K$ do
5      $P' = P \setminus P_g$;
6      $P = P_g$ ;
7      while $P' \neq \emptyset$ do
8          forall $r_i \in P'$ do
9              $IncreaseBonus_{r_i} = (rank(P_{Max2} \cup \{r_i\}) - rank(P_{Max2}))/k_i$
10         $r_{MaxIncrease} = argmax_{r_i \in P'} IncreaseBonus_{r_i}$
11         if $k_{MaxIncrease} + TotalCost \leqslant K$ then
12             $P = P \cup \{r_{MaxIncrease}\}$ ;
13             $TotalCost = k_{MaxIncrease} + TotalCost$ ;
14             $P' = P' \setminus (\{r_{MaxIncrease}\} \cup \{r \in P' : k_r > K - TotalCost\})$
15     if $rank(P) > rank(P_{max2})$ then
16         $P_{Max2} = P$
17
18 return $argmax_{P \in \{P_{Max1}, P_{Max2}\}} rank(P)$
```

the entire set of nodes. I implement my low cost monitoring algorithms in python and used the *Gurobi* optimization toolkit, on a 120-core, 2.5 GHz, 4TB RAM cluster [42]. I assume shortest path routing (based on hop count), with ties broken arbitrarily.

I use different network topologies including a small, medium and a large real topology taken from the Internet Topology Zoo [1, 5]. I also consider AS28717 (CAIDA) topology taken from the CAIDA (Center for Applied Internet Data Analysis) resource collection [100]. The network topologies used in my evaluation is shown in Figure 4.4. Table 5.2 shows the characteristics of the topologies used for the evaluation.

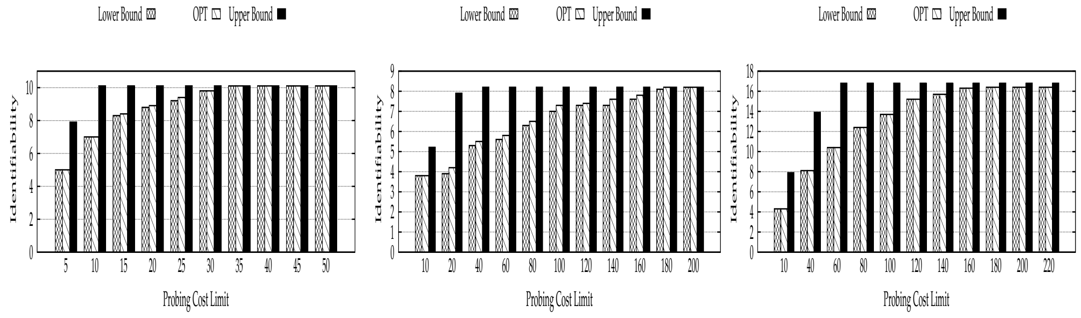
Table 4.3: Network characteristics used in our evaluation.

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.

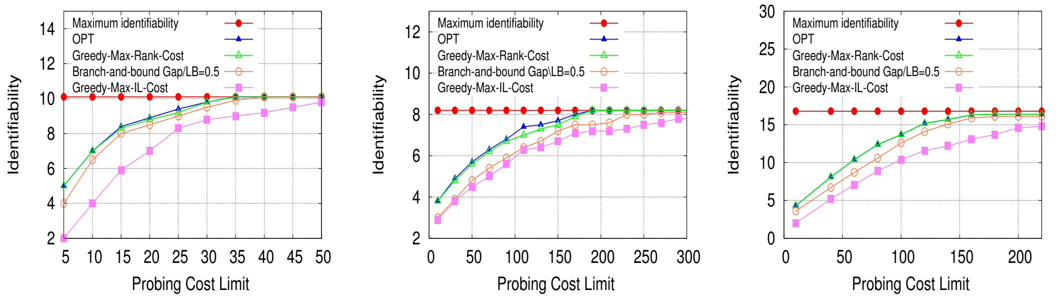


(a) Abilene. (b) BellCanada. (c) CAIDA.

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).

4.5.1 Identifiability Maximization

In the first set of simulations, I consider the impact of probing cost limit on the number of identifiable links. I first compare the upper and lower bound of Theorem 9, 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. I note that the optimal number of identifiable links is



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

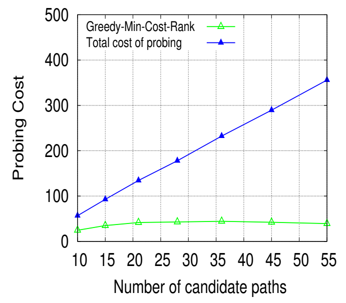
Figure 4.6: Number of identifiable links as a function of limit on the probing cost in a) Abilene (9 monitors), b) BellCanada (10 monitors) and c) CAIDA topology (9 monitors).

always upper bounded by the minimum of (i) maximum identifiability and (ii) the upper bound in Theorem 9. 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, I note that the lower bound is closer to the optimal and gives a tighter bound in terms of number of identifiable links.

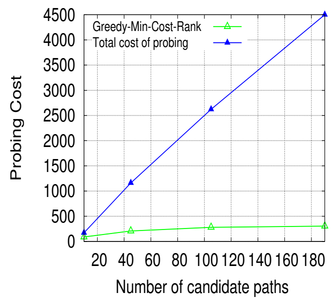
I next compare the number of identifiable links in Greedy-Max-IL-Cost heuristic (Algorithm 6), Greedy-Max-Rank, and the optimal case (OPT). I use gurobi optimization toolkit to solve the ILP problem formulation (equation 4.12). I also use my iterative branch-and-bound algorithm and stop the search when $Gap \leq 0.5 \cdot LB$. I recall that, the larger the gap is, the lower is the number of iterations of the optimization algorithm is and therefore I have an approximation of the solution which is farther from optimal. Figures 4.6a, 4.6b and 4.6c show scenarios where I increase the limit on the probing cost of monitors for the Abilene, BellCanada and CAIDA topology. As I 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

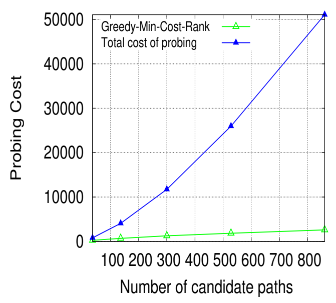
In the next set of simulations, I evaluate the performance of *Greedy-Min-Cost-Rank* algorithm that preserves rank and the greedy-based heuristics that preserve



(a) Abilene.

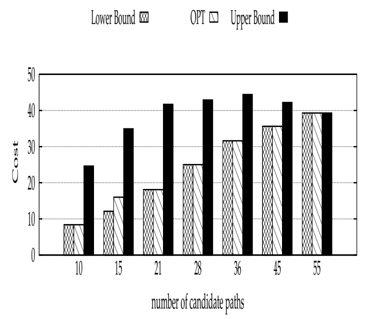


(b) BellCanada.

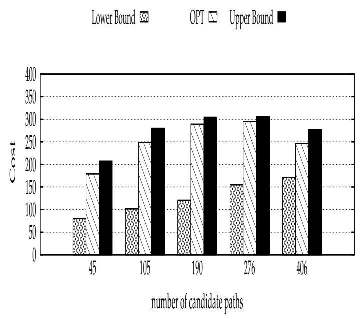


(c) CAIDA.

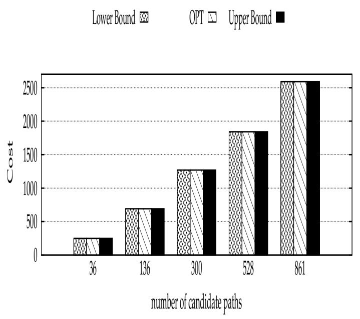
Figure 4.7: Probing cost vs number of feasible probing paths in a) Abilene (5-11 monitors), b) BellCanada (10-29 monitors) and c) CAIDA (9-42 monitors) topology.



(a) Abilene.

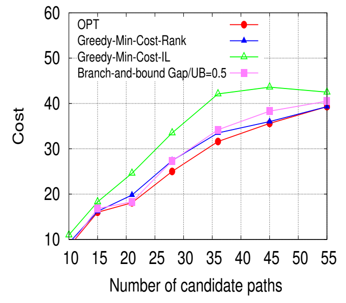


(b) BellCanada.

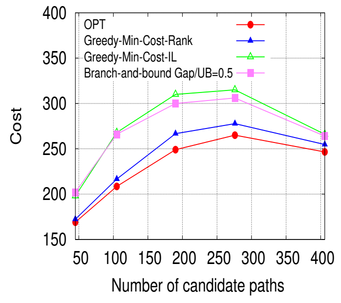


(c) CAIDA.

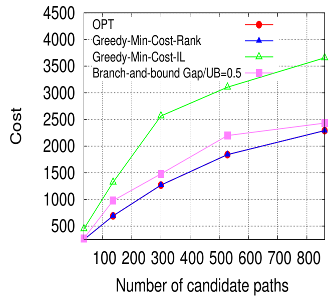
Figure 4.8: Upper and lower bound on the probing cost as a function of number of candidate paths in a) Abilene (5-11 monitors), b) BellCanada (10-29 monitors) and c) CAIDA (9-42 monitors) topology.



(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.

identifiability. I consider Abilene, BellCanada and CAIDA topology and run Greedy-Min-Cost-Rank and Greedy-Min-Cost-IL algorithms that preserve rank and identifiability respectively. I also run my branch-and-bound formulation and stop branching once $Gap \leq 0.5 \cdot UB$. In the first set of experiments, I increase the number of candidate paths and evaluate the cost saving of my Greedy-Min-Cost-Rank algorithm with respect to the case where I 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 my Greedy-Min-Cost-Rank algorithm significantly reduces the cost. I 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 7. Figure 4.8 shows the upper and lower bound on the probing cost as I 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.

I 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 I increase the number of candidate paths. As shown, my 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. I 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). I proposed an optimal greedy-based approach for (4) and proposed a $(1 - 1/e)$-approximation algorithm for (2). My experimental analysis reveals that, compared to several greedy approaches, our rank-based optimization performs better in terms of identifiability and probing cost. Furthermore, my solution can reduce the total probing cost by an order of magnitude while achieving the same monitoring performance.

Chapter 5

A Minimally Disruptive Rule Update in Software Defined Networking

This chapter addresses the problem of re-routing existing flows in a software defined network to enable the admission of new flows, while minimizing the disruption of existing flows under link capacity and Quality of Service (QoS) constraints. I show that the update of routing rules for an existing flow can cause packet loss and flow disruption. I aim to find paths for all active flows (including existing and new flows) to minimize the total disruption time or the number of disrupted flows. I formulate the problem as an integer linear programming problem and show that it is NP-Hard. I propose two randomized rounding algorithms with bounded congestion and demand loss to solve this problem. In addition to preliminary experiments on an SDN testbed, I performed a large-scale simulation study to evaluate my proposed approaches on real network topologies. Extensive simulation results show that my two random rounding approaches have a disruption cost close to the optimal while having a low congestion factor and a low demand loss.

5.1 Introduction

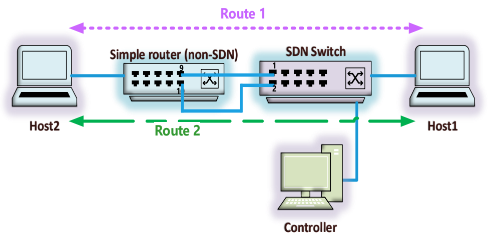
Software Defined Networks (SDN), which decouple the control plane and data plane, provide a powerful tool for network management, traffic engineering, and network policy enforcement. Decoupling the control functions and data plane brings significant advantages, including routing flexibility, being vendor agnostic, and centralized control and programmability [101, 102].

While SDN provides a rich framework for packet forwarding, forwarding rules may change 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 changing network conditions, which requires fast and safe update of the flow table entries, without causing major disruptions to existing flows.

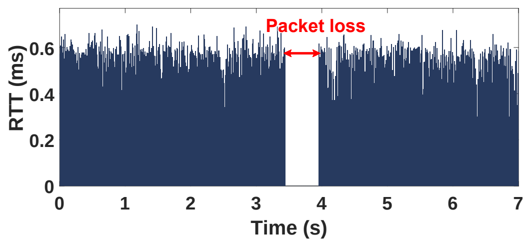
Updates of flow forwarding rules in SDN are the main source of service disruption. Rule updates occur very frequently in SDN due to the inherent flow dynamism, and to topology changes. Rule updates can disrupt the existing traffic by causing packet losses, delays, and security holes in the system [103–106].

While most prior works on SDN updates focus on providing consistent updates that protect against loops and policy violations, very few consider the disruption of communications experienced by the existing flows during the update. In general, it is not always possible to find an update schedule that (i) preserves policy consistency, (ii) avoids congestion during the update, and (iii) satisfies all the demands. In this chapter, I first provide consistency using two existing approaches: (1) the two-phase update approach [103], which I call the ***synchronous*** update approach, and (2) the sequential update approach [107], which I call the ***asynchronous*** update approach. In the synchronous update approach, all updates need to wait for the slowest switch to complete the update, while in the asynchronous update approach, each flow gets an update independent of other flows, which privileges update time at the expense of temporary congestion and lack of consistency. I then show the trade-off between (ii), (iii) and disruption cost.

Figure 5.1 shows an experiment that characterizes the disruption due to flow rule update in my evaluated SDN testbed which is shown in Figure 5.1a. I first install rules on a 24-port Brocade (ICX 6610) SDN switch to connect two hosts via a (non-SDN) router using *Route 1*. Then, I update the flow table in the Brocade switch to use *Route 2*. During the update, I send ICMP packets from *Host1* to *Host2* to measure the round trip time (RTT). Figure 5.1b shows RTT measurements, before, during and after the rerouting. The results show that rerouting the *Host1* $\rightarrow$ *Host2* flow from *Route 1* to *Route 2* disrupts the flow for about 500 *ms*, i.e.



(a) Testbed setup.



(b) Measured disruption.

Figure 5.1: Disruption caused by re-routing the existing flow to accommodate new flows.

ICMP packets are lost during the 500 ms period.

Motivated by these observations, I propose a minimally disruptive rule update framework that minimizes such disruptions while satisfying flow demands under link capacity and QoS constraints. To this end, this chapter makes the following main **contributions**:

- I formulate the minimally disruptive network update (*Min-touch*) problem as an integer linear programming (ILP) problem that minimizes the flow disruption and show that it is NP-Hard.

- I propose two randomized rounding algorithms, *RR-Cong* and *RR-Demand*, which aim at solving *Min-touch* from different perspectives. *RR-Cong* solves *Min-touch* with a bounded amount of congestion, and *RR-Demand* provides a bounded demand loss.

- I present a simulation-based evaluation of my proposed approach based on real network topologies and demonstrate the advantage of my proposed approach in terms of disruption cost, compared to existing update approaches that aim at minimizing the routing cost.

The remainder of this chapter is organized as follows. Section 5.2 discusses the background and related works. In section 5.3, I explain the *Min-touch* optimization problem and show that it is NP-Hard. Section 5.4 describes my algorithms. Section 5.5 shows my evaluation methodology and results. Section 5.6 concludes the chapter with a summary.

5.2 Background and Related Work

In the SDN paradigm, the controller monitors and controls network elements and defines forwarding rules via protocols such as the OpenFlow [108, 109]. 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 a set of actions. Openflow switches are required to support *output*, *drop* and *group* actions [109]. Packets are matched based on criteria defined in the rules of the flow tables and forwarded according to the *output* action in the corresponding entry of the flow table. Packets whose output action are not specified should be dropped. The *group* action processes packets based on the specified group to support multi-path routing [105, 109].

While SDN has great potential to dramatically simplify network management, rule updates are an inseparable part of SDN network management and occur very frequently in SDN due to arrival of new flows, termination of existing flows, and topology changes. Service disruption and inconsistencies can occur during the updates leading to degraded QoS or interruption of existing services. The problem of mitigating undesirable behaviors during an update has been studied by several works [103, 106, 110–118].

Reitblatt et al. propose per-packet and per-flow consistency to overcome instabilities caused by network re-configuration [103, 112]. Per-packet consistency ensures that each packet follows either the old rules or the new rules. It can be implemented by maintaining both sets of rules in the TCAM memory of the switches, a.k.a the *two-phase update* approach. Per-flow consistency guarantees that all packets in the same flow will follow the same rule version. The two-phase update approach doubles the total precious TCAM memory usage during the update, and the update time can be long due to the straggler switch which delays other installed rules from becoming active [103]. However, the two-phase update approach provides strong consistency and avoids transient congestion during the update phase.

To reduce the TCAM memory usage during rule updates, prior works studied sequential update schedules that preserve consistency [107, 110, 114, 119, 120]. Vissicchio et al. proposed a sequential update schedule for rule replacements and additions that preserves forwarding policies [110]. While the algorithm works well for a single flow update, the assumption of each flow being independent of other flows is limiting. The algorithm may cause high packet loss during the update period due to congestion since it does not consider link capacity constraints. Further, the algorithm does not consider the routing of flows and assumes that the routing problem is already solved.

Ludwig et al. proposed an update schedule that preserves waypoint enforcement and the loop-freedom property. They showed that there exists scenarios where both properties cannot be satisfied simultaneously [107]. The algorithm considers the update of a single flow and does not consider the conflict of flows. Vissicchio et al. proposed an SDN update algorithm that prevents inconsistencies and preserves forwarding policies [110]. Xu et al. proposed a joint optimization of update scheduling and route selection [121]. Mizrahi et al., leveraged a Time Precision Protocol (TPP) to avoid inconsistencies in SDN global updates while minimizing the transition time [111]. Also, Katta et al. introduced an incremental network update method with a trade-off between the update time and the TCAM space overhead [122]. Wang et al. proposed an update ordering approach that constructs a dependency graph to avoid potential congestion and minimize the risk of deadlocks [123].

The proposed minimally disruptive update framework is built on the two existing prior works (two-phase update approach and sequential update approach) that provide consistency with different trade-offs between the update time and consistency level. I note that while the update time of the sequential update approach is lower than the two-phase update approach, it does not provide as strong a consistency level as the two-phase update approach. In contrast to the aforementioned prior works that focused on implementing a given set of new rules, I focus on the computation of new rules itself, while trying to minimize the cost of implementing these rules using one of the existing update approaches.

5.3 Problem Formulation

5.3.1 Network Model

I consider the problem of minimizing the service disruption due to rule updates on TCAM-based SDN switches from a traffic engineering perspective. Given a capacitated undirected graph $G = (V, E)$, where V represents the SDN switches and E is the set of communication links connecting the switches. I also assume having a set of flow demands H, where each flow h has a source s_h, a destination t_h, and d_h units of demand. Each link $(i, j) \in E$ has a capacity of c_{ij}. I aim at satisfying all demands while minimizing the disrupted services. To focus on minimizing the disruption, I assume the demands have been filtered by proper admission control, such that there exists at least one feasible solution which can satisfy all flows. To differentiate existing and newly arrived flows, I use H_{old} to denote the set of exiting flows, and H_{new} to denote the set of new flows. I have $H = H_{old} \cup H_{new}$.

Table 5.1 summarizes the **notations** used in my formulation.

5.3.2 Optimization Problems

I propose the minimally disruptive update problem (***Min-touch***) to route new and existing flows in the network with minimum impact on the existing flows. I present concrete formulations for two versions of Min-touch for ***synchronous*** and ***asynchronous*** update approaches, respectively. The synchronous version assumes that the two phase update approach [103] will be used to implement the updates, in which all flows wait for the straggler switch to finish the update. The asynchronous version assumes that the sequential update approach [107, 110] will be used to implement the updates, where each flow is updated independently of the other flows.

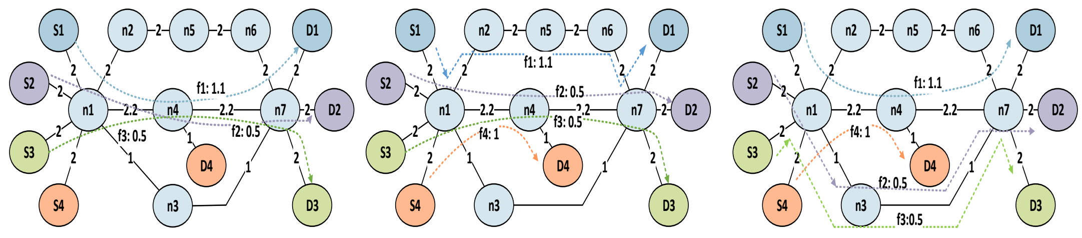
• **Objective function:** I aim to minimize the disruption cost which is formulated with $\sum_{h \in H} \delta_h \cdot g_h(T_h)$, where δ_h is a binary variable which is 1 if flow H is re-routed, and 0 otherwise, $g_h(T_h)$ denotes the cost of disruption for flow h, and T_h denotes the disruption time of flow h. As a concrete example, I will consider linear cost, $g_h(T_h) = w_h \cdot T_h$, where w_h is the priority of minimizing disruption for flow h. As explained below, in a synchronous update approach, all flows will experience the same disruption time, i.e. $T_h = T$, $\forall h \in H$, where T is the time taken to update the straggler switch. In an asynchronous update approach, each flow gets disrupted only once and does not need to wait for the straggler switch to finish, i.e. $T_h = 1$, $\forall h \in H$.

Example: To clarify the trade-off between the disruption cost and number of

Table 5.1: Notations used in my formulations.

Notation	Explanation
$G(V, E)$	an undirected graph where V represents the set of nodes and E is the set of links
H	the graph of all flow demands $H = (V_h, E_h)$ where $E_h = \{(s_1, t_1), ..., (s_h, t_h)\}$
H_{old}	set of flow demands that exist before the update
H_{new}	set of new flow service demands
c_{ij}	capacity of each link $(ij) \in E$
$\tilde{c}_{ij}$	residual capacity of link $(ij) \in E$
H_r	set of flow demands that should be re-routed according to the new setting
b_i^h	the amount of flow generated/consumed by node i
k_{ij}^h	the current routing decision to use link (i, j) for flow h (when $k_{ij}^h = 1$), or not ($k_{ij}^h = 0$)
x_{ij}^h	the new routing decision to use link (i, j) for flow h (when $x_{ij}^h = 1$), or not ($x_{ij}^h = 0$)
δ_h	binary variable that specifies if flow h is re-routed in the new setting ($\delta_h = 1$), or not ($\delta_h = 0$)
θ_h	binary variable that specifies if an existing flow h is not re-routed but crosses an updated switch (when $\theta_h = 1$), or does not cross an updated switch (when $\theta_h = 0$)
t_i^h	a binary variable that specifies if switch i gets updated by flow h (when $t_i^h = 1$), or not, when ($t_i^h = 0$)
d_h	amount of demand flow for flow h
$\tilde{d}_h$	unrouted demand of flow h
w_h	priority of flow h
T_h	disruption time for flow h
Ω_h	QoS constraint for flow h
Ω	global QoS constraint for all flows
τ_h	duration time of flow h
$g_h(T_h)$	disruption cost for flow h

hops, consider a simple example shown in Figure 5.2. The figure shows a network with seven switches. Initially three source-destination flows, $\{f_1 : S1 - D1, f_2 : S2 - D2, f_3 : S3 - D3\}$, exist in the network with $\{1.1, 0.5, 0.5\}$ units of demand correspondingly. Consider an update where a new flow f_4 with 1 unit of demand arrives at the system. To admit the new flow, I can either (i) re-route f_1 through the upper links without disrupting f_2 and f_3, as in the first target state in Figure 5.2b,



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

Figure 5.2: Two ways to update rules from the original state (a) to target states that (b) increases the total hop count, or (c) causes the disruption of existing flow.

with a total of 17 hops, or (ii) re-route f_2 and f_3 through the lower links without disrupting f_1 and route the new flow through $S4 - n1 - n4 - D4$ which requires fewer hops (15) in the second target state in Figure 5.2c, but disrupts more flows (f_2 and f_3).

The decision on which solution to use, can be based on either minimizing the number of flows that are disrupted, or minimizing the total disruption time of the rerouted flows. In order to update the network from the original state to the target state, the controller updates the flow table in the corresponding switches. Depending on the size of the flow tables and the number of rules that must be installed, the update on each switch can take several hundreds of milliseconds to complete, causing a long disruption of flow f_1 in the first case or f_2 and f_3 in the second case.

In the synchronous update approach the flow rules in $n1, n2 \cdots, n7$ would all be updated simultaneously. Traffic will not flow in the disrupted flows until the last switch, the straggler switch, has completed its updates. In this case, route consistency is guaranteed, but the disruption of all flows depends on the slowest switch. In the asynchronous update approach, each flow gets an update independent of the other flows. In the example of Figure 5.2, the paths of each flow are updated in parallel, starting from the last switch in the flow. Thus, in order to update f_1 in the first case, $n7$, $n6$, $n5$ and $n2$ are updated first, while f_1 continues to use its old path, and then $n1$ gets updated causing a disruption to flow f_1.

• **Input parameters:** Let b_i^h be the amount of flow h generated by node i which is $b_i^h = d_h$ if i is the source node ($i = s_h$), and $b_i^h = -d_h$, if i is a destination node ($i = d_h$) and $b_i^h = 0$ otherwise. I denote k_{ij}^h to be the existing routing decision

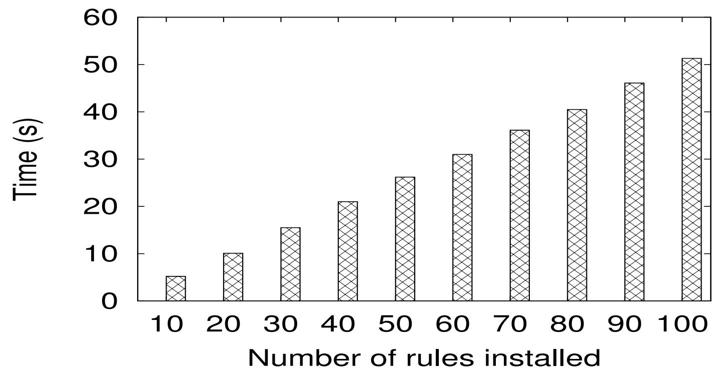


Figure 5.3: Total rule installation time on a 24-port Borocade (ICX 6610) SDN switch.

to use link (i, j) for flow h if $k_{ij}^h = 1$ or not $k_{ij}^h = 0$. Further, let Ω_h be the QoS threshold for flow h in terms of number of hops the solution increases the paths by over the shortest path solution and Ω be the global QoS constraint for all flows in terms of total number of hops the solution increases the paths by over the shortest path solution for all flows.

• **Decision variables:** Let x_{ij}^h be the new routing decision variable to use link (i, j) for flow h when $x_{ij}^h = 1$ or not $x_{ij}^h = 0$. I note that flow $h \in H_{old}$ is re-routed if and only if $\exists (i, j) \in E$ such that $x_{ij}^h = 1$ and $k_{ij}^h = 0$. As I already mentioned when describing the objective function, the variable δ_h represents the decision to re-route flow h in the new setting ($\delta_h = 1$) or not ($\delta_h = 0$). As the updates on each switch occurs sequentially, I can only update one flow rule on each switch at a time. I use T to denote the maximum number of updates between all switches. I also introduce an auxiliary variable t_i^h to specify if switch i gets updated by flow h when $t_i^h = 1$ or not $t_i^h = 0$.

5.3.2.1 Minimally Disruptive Synchronous Update (Min-touch Synch)

I formulate the minimally disruptive update (*Min-touch Synch*) optimization problem as follows:

$$\text{minimize} \quad \sum_{h \in H} \delta_h \cdot w_h \cdot T_h$$

$$\text{subject to} \sum_{h\in H}(x_{ij}^h + x_{ji}^h)\cdot d_h \le c_{ij}, \;\; \forall (i,j)\in E \tag{5.1a}$$

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

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

$$\sum_{(i,j)\in E}(x_{ij}^h + x_{ji}^h) \le \Omega_h, \;\; \forall h\in H \tag{5.1d}$$

$$\sum_{h\in H}\sum_{(i,j)\in E}(x_{ij}^h + x_{ji}^h) \le \Omega, \tag{5.1e}$$

$$t_i^h \ge \frac{\sum_{j\in V}[x_{ij}^h + k_{ij}^h - 2k_{ij}^h x_{ij}^h]}{|V|} \;\; \forall i\in V, \, \forall h\in H \tag{5.1f}$$

$$\sum_{h'\in H} t_i^{h'} \le T_h, \;\; \forall i \in V \;\; \forall h\in H \tag{5.1g}$$

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

Constraint 5.1a specifies that the fraction of flow that will be routed through link (i, j) has to be smaller than or equal to the capacity of that edge. Constraint 5.1b shows the flow balance, 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 5.1c ensures that δ_h is set to 1 when flow h is re-routed. Constraint 5.1d is the per flow QoS constraint which ensures that each flow's total number of hops in the current routing is not greater than a threshold Ω_h. Constraint 5.1e is the global QoS constraint which ensures that the summation of total number of hops for all flows does not exceed a threshold Ω. Constraints 5.1f ensures that t_i^h will be 1 if there is a rule update on switch i for flow h, and will ne 0 otherwise. Constraint 5.1g shows the maximum update time is dominated by the straggler switch which takes the most time to update. In words, formulation (5.1) aims at minimizing the weighted total disruption time such that all the flows can be routed subject to link capacity and QoS constraints.

In a synchronous update approach, I provide consistency using the two-phase update approach as proposed in [103], and send a batch update to all switches. In this approach all flows have to wait for the straggler switch to finish updating. Therefore, the disruption times for all flows are the same and are equal to the update time of the straggler switch, i.e. $T_h = T \; \forall h \in H$.

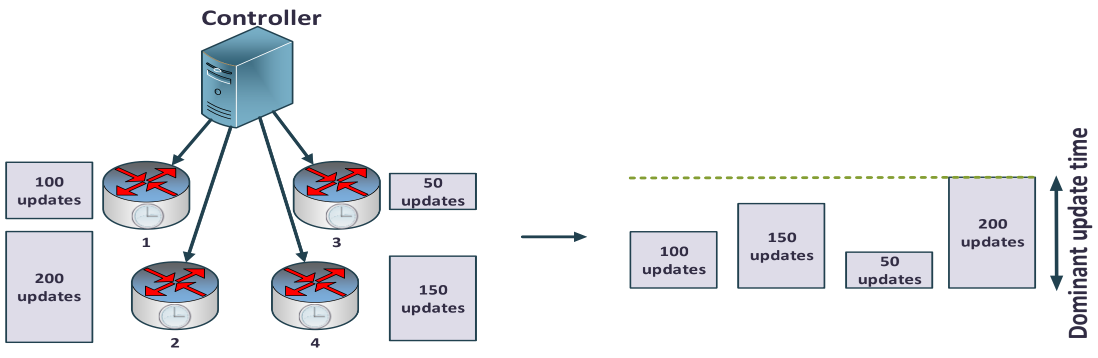


Figure 5.4: Dominant update time.

Figure 5.3 shows flow installation time on a 24-port Brocade (ICX 6610) commercial SDN switch. As shown, when I increase the number of rules to install on the switch from 10 to 100, the rule installation time increases linearly from 5.2 (s) to 51.3 (s). In an example, shown in Figure 5.4, the controller sends an update to 4 SDN switches in parallel, each with a different number of rules to be installed. The dominant update time is the switch which takes the most time to install the scheduled update [124–128] which is switch 2 in this example.

Linearization: I consider a linear cost of disruption for each flow $g_h(T_h) = w_h \cdot T_h$, and define a new variable $\Delta_h = T_h \cdot \delta_h$ that shows the disruption cost for each flow due to the update. Since the objective function is non-linear, I add the following three constraints to make the optimization problem linear. The objective function is to minimize the total disruption cost, i.e. $\sum_{h \in H} w_h \cdot \Delta_h$.

$$\Delta_h \leq \delta_h \cdot |H|, \quad \forall h \in H \tag{5.2a}$$

$$\Delta_h \geq 0, \quad \forall h \in H \tag{5.2b}$$

$$\Delta_h \leq T_h, \quad \forall h \in H \tag{5.2c}$$

$$\Delta_h \geq T_h - |H| \cdot (1 - \delta_h), \quad \forall h \in H \tag{5.2d}$$

5.3.2.2 Minimally Disruptive Asynchronous Update (Min-touch Asynch)

In an asynchronous update approach, each flow gets an update independent of the other flows. Assuming the same per-rule update time on all switches, the disruption time for all flows is the same, i.e. $T_h = 1$, $\forall h \in H$, and the goal is to minimize the total disruption cost. I formulate the minimally disruptive update (*Min-touch Asynch*) optimization problem as follows:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{h \in H} \delta_h \cdot w_h \\
\text{subject to} \quad & (5.1a - 5.1e), && (5.3a) \\
& x_{ij}^h, \delta_h \in \{0,1\}, \;\; \forall (ij) \in E, \;\; \forall h \in H \;\; \forall i \in V && (5.3b)
\end{aligned}
$$

In words, formulation (5.3) aims at minimizing the weighted number of disrupted flows such that all the flows can be routed subject to link capacity and QoS constraints.

I have the following results for both Min-touch Synch and Min-touch Asynch.

Theorem 10 ***No Cycle Removal****: No cycle removal is needed in the Min-touch optimization problem.*

Proof of Theorem 10 I prove by contradiction that any optimal solution containing cycles has an equivalent optimal solution without cycles. Suppose that the optimal solution contains cycles, by removing the cycles from the optimal solution I have a solution with fewer updated switches which potentially has fewer selected edges x_{ij}s; and therefore fewer affected flows that needs to be re-routed. Therefore, the new optimal solution by removing the cycles is no worse than the optimal solution with cycles.

Theorem 11 ***NP-Hardness****: The problem Min-touch is NP-Hard.*

Proof of Theorem 11 I prove that the Min-touch problem is NP-hard by giving a reduction from the well-known edge-disjoint paths problem (EDP). In EDP, the goal is to find a routing schedule which can simultaneously connect a set of source/destination pairs using edge-disjoint paths in G. EDP is NP-hard even when restricted to planar graphs [136]. An instance of EDP is specified by a set of demand pairs $\{(s_1, t_1), ..., (s_h, t_h)\}$. The goal is to find a collection of edge disjoint paths $P_1, ..., P_h$ such that for $1 \leq i \leq h$, P_i is a path from s_i to t_i.

In the following I show how I can build, in polynomial time, an instance of Min-touch problem whose solution corresponds to the solution of the EDP problem given above. I create one unit of demand flow for each demand pair in the supply graph. I also assume that all edges in G have unit capacity. I assume that there exists no current flows in the network and therefore, the problem is to satisfy all new demands using edge-disjoint paths. I set the QoS threshold for each flow equal to the number of edges in the network ($\Omega_h = |E|$), and the global QoS to be $\Omega = |E| \cdot |H|$. Thus, the QoS constraints are set large enough such that it does not imply any restriction on the edge-disjoint paths found. Therefore, EDP problem changes to an instance of Min-touch problem. I can therefore conclude the reducibility of the EDP problem to Min-touch, and consequently show that the problem Min-touch is NP-hard.

5.3.2.3 Baseline: Min-Edge-Cost

I note that, to avoid flow disruptions, my minimum disruptive update approach may increase the path lengths of flows. Therefore, to evaluate the increase in the path length in my approach, I formulate a baseline problem, *Min-Edge-Cost*, that finds a set of paths with the minimum number of hops to route the flows. This problem of finding a minimum cost route for unsplittable flows is shown to be NP-Hard [6, 129]. Nevertheless, this baseline, when solved optimally, provides a lower bound on the routing cost (measured by hop count).

$$\text{minimize} \sum_{(ij) \in E} \sum_{h \in H} (x_{ij}^h + x_{ji}^h)$$

$$\sum_{h \in H} (x_{ij}^h + x_{ji}^h) \cdot d_h \leq c_{ij}, \quad \forall (i,j) \in E \tag{5.4a}$$

$$\sum_{j \in V} x_{ij}^h = \sum_{k \in V} x_{ki}^h + sign(b_i^h), \quad \forall i \in V, \quad \forall h \in H \tag{5.4b}$$

$$x_{ij}^h \in \{0, 1\}, \quad \forall (ij) \in E, \quad \forall h \in H \tag{5.4c}$$

Later, in Section 5.5, I show that *Min-Edge-Cost* can perform very poorly in terms of disruption to the existing flows but provides an optimal solution in terms of number of hops used in my solution.

5.4 Minimum Disruptive Update Algorithms

The Min-touch problem is hard to solve optimally and efficiently as proved in Theorem 11. Therefore, in this section, I propose two randomized rounding algorithms that each aims at solving the problem from a different perspective. Both algorithms are applicable to both versions of Min-touch (i.e., Min-touch Synch and Min-touch Asynch).

5.4.1 Randomized Rounding Algorithms

In the following, I first propose a randomized rounding approach with bounded congestion (*RR-Cong*). Next, I propose a randomized rounding approach with a bounded amount of demand loss (*RR-Demand*). In my analysis I make use of the following version of Chernoff Hoeffding bound.

Lemma 8 ***Chernoff Hoeffding bound [130]*** *Let $X = \sum_{i=1}^{n} X_i$ be the summation of n independent random variables $X_i \in [0,1]$ with $E(X_i) \leq \mu_i$ and $\sum \mu_i = \mu$. Then, $\forall \epsilon > 0$,*

$$Pr\left(\sum_{i=1}^{n} X_i \geq (1+\epsilon)\mu\right) \leq e^{\frac{-\epsilon^2\mu}{2+\epsilon}} \tag{5.5}$$

5.4.1.1 Randomized Rounding with Bounded Congestion

I first propose a randomized rounding approach that bounds the maximum amount of congestion on the links. Algorithm 9 shows different steps of RR-Cong algorithm. The algorithm first solves the LP-relaxation of the Min-touch problem whose objective value will be lower than the optimal (line 1). The LP-relaxation of the problem gives fractional solution $\tilde{x}_{ij}^{h}$ indicating the fraction of flow h routed on link (i,j). I then convert this solution into a set of routing paths $\mathcal{P}_h$ for each flow and the fraction y_p^h of flow h routed on each path $p \in \mathcal{P}_h$ (line 2).

The conversion from the link-level fractional solution in the LP-relaxation to the

Algorithm 9: RR-Cong algorithm for *Min-touch* problem

```
1 Solve LP-relaxation of Min-touch problem.
2 Convert the solution into a set of routing path $\mathcal{P}_h$ for each flow $h$ and the
    routing fraction $y_p^h$ for each path $p \in \mathcal{P}_h$.
3 for each flow $h \in H$ do
4     Independently choose a path $p \in \mathcal{P}_h$ in the converted solution with
        probability $y_p^h$.
5     if $min_{(i,j)\in p}(c_{ij}) < d_h$ then
6         drop flow $h$.
7     else
8         route a flow of rate $d_h$ over $p$ .
```

path-level fractional solution is performed as follows. For each source-destination pair s_h-t_h I first find a path p from s_h to t_h using breadth first search. Let y denote the bottleneck residual capacity of p. I then route y units of flow on path p, i.e. $y_p^h = y$, and update the residual capacities by subtracting y from the residual capacities of links on the path p. I repeat these steps until I cannot find any more paths with positive residual capacity from s_h to t_h. I then randomly select a path p for flow h based on the probability of that path, y_p^h, from the LP-relaxation solution (lines 3-4). If the minimum capacity of the selected path is smaller than d_h, I drop flow h, and otherwise, I route a flow of rate d_h over the selected path p (lines 5-8).

I now analyze the performance of RR-Cong in terms of the amount of congestion. Given a link (i, j) I first define a set of independent random variables $\{f_{ij}^h | h \in H\}$ where f_{ij}^h shows the amount of flow h that crosses a link (i, j):

$$\text{If } \ d_h > c_{ij} : \qquad f_{ij}^h = 0 \qquad \text{with probability 1,}$$

and

$$\text{If } \ d_h \le c_{ij} : \quad f_{ij}^h = \begin{cases} d_h & \text{with probability } (\tilde{x}_{ij}^h + \tilde{x}_{ji}^h), \\ 0 & \text{otherrwise.} \end{cases}$$

The random variables $\{f_{ij}^h | h \in H\}$ are mutually independent due to the independent rounding in line 4. The expected amount of flow routed through link (i, j) is computed as follows:

$$\mathbb{E}\Big[\sum_{h\in H}(f_{ij}^h)\Big] = \sum_{h\in H}\mathbb{E}(f_{ij}^h) \le \sum_{h\in H} d_h\cdot(\tilde{x}_{ij}^h + \tilde{x}_{ji}^h) \le c_{ij}$$

I define $\mu_h := \frac{d_h\cdot(\tilde{x}_{ij}^h+\tilde{x}_{ji}^h)}{c_{ij}} + \zeta$, where,

$$\zeta := \frac{1}{|H|}\left(1 - \frac{1}{c_{ij}}\sum_{h\in H} d_h\cdot(\tilde{x}_{ij}^h + \tilde{x}_{ji}^h)\right) \ge 0 \tag{5.7}$$

Then,

$$\mathbb{E}\Big[\frac{f_{ij}^h}{c_{ij}}\Big] \le \mu_h, \quad \text{and,} \qquad \sum_{h\in H}\mu_h = 1 \tag{5.8}$$

Theorem 12 ***RR-Cong congestion bound*** *Under RR-Cong, the probability of violating the link capacity constraint for any link (i,j) by a factor of $1+5log(|E|)$ is not more than $1/|E|^2$, i.e.,*

$$Pr\left(\exists(i,j)\in E:\ \sum_{h\in H} f_{ij}^h \ge (1+5log(|E|))\cdot c_{ij}\right) \le 1/|E|^2 \tag{5.9}$$

Proof of Theorem 12 I consider the normalized random variable f_{ij}^h/c_{ij} which resides in the interval $[0,1]$. Assuming $|E|\ge 2$ ((5.9) holds trivially for $|E|=1$), I apply the Chernoff Hoeffding bound to the variable f_{ij}^h/c_{ij} with $\epsilon = 5log(|E|)$ as follows:

$$\begin{aligned} Pr\left(\sum_{h\in H}\frac{f_{ij}^h}{c_{ij}} \ge (1+5log(|E|))\right) &\le e^{\frac{-25log^2(|E|)}{2+5log(|E|)}} \\ &\le e^{-3log(|E|)} = 1/|E|^3 \end{aligned}$$

Using the union bound for $|E|$ random variables, the probability that the capacity of any link is violated by a factor of at least $1+5log(|E|)$ is no more than $1/|E|^2$.

Remark: I note that RR-Cong does not drop any flow if the capacity of all links are greater than or equal to the maximum demand, i.e. $min_{(i,j)\in E}(c_{ij}) \ge max_{h\in H}(d_h)$, although some links may be congested.

Algorithm 10: RR-Demand algorithm for *Min-touch* problem

```
1  Solve LP-relaxation of Min-touch problem with the additional constraint
     $\tilde{x}_{ij}^h \leq \frac{1}{log(|E|)} \quad \forall (i,j) \in E, h \in H$.
2  Convert the solution into a set of routing path $\mathcal{P}_h$ for each flow $h$ and the
     routing fraction $y_p^h$ for each $p \in \mathcal{P}_h$.
3  for each flow $h \in H$ do
4      Independently choose a path $p \in \mathcal{P}_h$ in the converted solution with
         probability $y_p^h \cdot log(|E|)$.
5      if $min_{(i,j) \in p}(c_{ij} < d_h)$ then
6          drop flow h.
7      else
8          route $\frac{1}{6log(|E|)}$ fraction of flow $h$ over $p$ and update the residual capacities
             $\tilde{c}_{ij} \ \forall (i,j) \in p$.
9  for each flow $h \in H$ and its selected path $p_h$ do
10     Increase the flow rate by $min(\tilde{d}_h, min_{(i,j) \in p_h}(\tilde{c}_{ij}))$, where $\tilde{d}_h$ is unrouted
         demand of flow $h$.
```

5.4.1.2 Randomized Rounding with Bounded Demand Loss

In this section, I propose a randomized rounding approach (RR-Demand) that does not cause link capacity violation with high probability, but may drop a percentage of demands. Algorithm 10 shows different steps of the algorithm. RR-Demand first solves the LP-relaxation of the problem with an additional constraint that each fractional link solution can only route $\frac{1}{log(|E|)}$ fraction of each flow (line 1). This solution is then converted into a set of routing paths for each flow as explained before (line 2). I then randomly choose a path $p \in \mathcal{P}_h$ with probability $y_p^h \cdot log(|E|)$ and route $\frac{1}{6log(|E|)}$ fraction of flow h over p if the bottleneck capacity of p is at least d_h; otherwise I drop flow h (lines 3- 8). After trying to route $1/(6 \cdot log(|E|)$ of demand for each flow, if the chosen path for flow h has any spare capacity, I increase the flow rate for this flow and update the residual capacities, until there is no more spare residual capacity for any of the flows (lines 9-10).

I show that using RR-Demand, the probability of violating any link capacity constraint is negligible. Let me define a set of independent random variables f_{ij}^h, each denoting the amount of flow h routed on link (i,j), as follows:

$$\text{If } \; d_h > c_{ij} : \qquad f_{ij}^h = 0 \qquad \text{with probability 1,}$$

and If $d_h \leq c_{ij}$:

$$f_{ij}^h = \begin{cases} \frac{d_h}{6 \cdot log(|E|)} & \text{with probability } log(|E|) \cdot (\tilde{x}_{ij}^h + \tilde{x}_{ji}^h), \\ 0 & \text{otherrwise.} \end{cases}$$

Theorem 13 ***RR-Demand congestion bound*** *Under RR-Demand, the probability of violating the link capacity constraint for any link (i,j) is not more than $1/|E|^2$, i.e.,*

$$Pr\left(\exists (i,j) \in E : \sum_{h \in H} f_{ij}^h \geq c_{ij}\right) \leq 1/|E|^2 \tag{5.11}$$

Proof of Theorem 13 I consider the normalized random variable $\frac{6 \cdot f_{ij}^h}{c_{ij}}$ which resides in the interval $[0, \frac{1}{log(|E|)}]$. The expected value is computed as follows:

$$\mathbb{E}\Big[\sum_{h \in H} (\frac{6 \cdot f_{ij}^h}{c_{ij}})\Big] = \sum_{h \in H} \mathbb{E}(\frac{6 \cdot f_{ij}^h}{c_{ij}}) \leq$$
$$\sum_{h \in H} \frac{1}{c_{ij}} \frac{d_h}{log(|E|)} \cdot log(|E|) \cdot (\tilde{x}_{ij}^h + \tilde{x}_{ji}^h)$$
$$\leq \sum_{h \in H} \frac{d_h}{c_{ij}} \cdot (\tilde{x}_{ij}^h + \tilde{x}_{ji}^h) \leq 1$$

I then apply the Chernoff Hoeffding bound to the variable $6 \cdot \frac{f_{ij}^h}{c_{ij}} \cdot log(|E|)$, where $\mu = log(|E|)$ and $\epsilon = 5$, as follows:

$$Pr\left(\sum_{h \in H} \frac{6 \cdot f_{ij}^h}{c_{ij}} \cdot log(|E|) \geq (1+5) log(|E|)\right) \leq e^{\frac{-25 log(|E|)}{2+5}}$$
$$< e^{-3 log(|E|)} = 1/|E|^3$$

Using the union bound for $|E|$ random variables, the probability that the capacity of any link is violated is no more than $1/|E|^2$.

Table 5.2: Network characteristics used in my evaluation.

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

Therefore, by reducing the amount of routed flow on a chosen path by a factor of $(6 \cdot log(|E|))$, RR-Demand satisfies link capacity constraints with high probability, and if the minimum link capacity is greater than or equal to the maximum demand, i.e. $min_{(i,j)\in E}(c_{ij}) \geq max_{h\in H}(d_h)$, I can satisfy a total demand of at least $\sum_{h\in H} d_h/(6 \cdot log(|E|))$.

5.5 Evaluation

In this section, I evaluate my algorithms, the baseline, and the optimal solution, under several scenarios, to compare the number of disrupted flows, congestion, straggler's update time, demand loss and total number of hops. For each scenario, I randomize the results by running 20 different trials, where I vary the random selection of demand pairs from the entire set of nodes. I implement my algorithms in python and used the *Gurobi* optimization toolkit, on a 120-core, 2.5 GHz, 4TB RAM cluster [42].

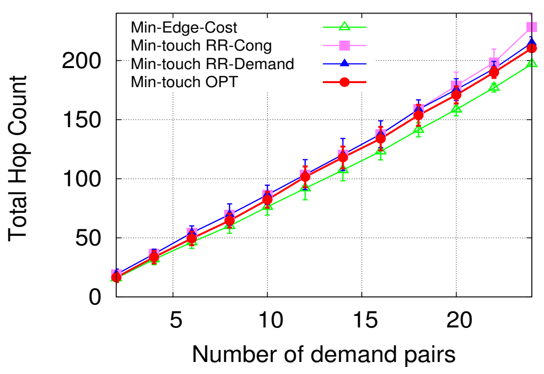
I use real Internet Service Provider (ISP) topologies including the BellCanada topology taken from the Internet Topology Zoo [1, 5] and AS28717 topology taken from the CAIDA (Center for Applied Internet Data Analysis) dataset [100]. Table 5.2 shows the characteristics of the topologies used for the evaluation. Unless otherwise specified, in all experiments, half of the demand pairs are existing flows and the rest are new flows; also, to set the QoS constraints (Ω and Ω_h) in my evaluation, I allow a 20% QoS deviation from the shortest path routing. For most of experiments, I only present results on the BellCanada topology below due to space limitations, but similar observations have been made on the CAIDA topology. For evaluation of QoS constraints in Section 5.5.3 I use the CAIDA topology, because it provides many more paths between demand pairs.

5.5.1 Synchronous Updates

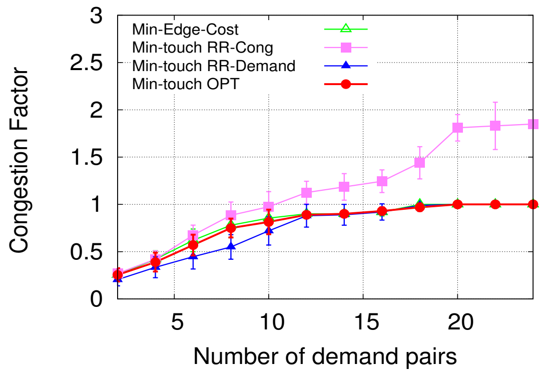
In the first set of experiments, I use the BellCanada topology where each link's capacity is chosen randomly in the interval $[20, 50]$. Each demand pair has a flow rate of three. I define the *congestion factor* to be the maximum amount of flow divided by the capacity of the link for all links in the network. In cases where none of the links are overloaded, the congestion factor is less than or equal to one. I then increase the number of demand pairs from 2 to 24 and compare (i) total number of hops, (ii) congestion factor, (ii) demand loss, and (iv) objective function in Min-Edge-Cost, RR-Cong, RR-Demand and OPT. Figure 5.5 shows the experimental results for this scenario. As shown, Min-Edge-Cost routes all the flows with the minimum number of total hops but its total disruption cost is higher than the other algorithms by a factor of three. This is due to the fact that Min-Edge-Cost does not consider the disruption cost and the final solution reroutes almost all existing flows to minimize the number of hops. I note that my random rounding approaches are within 20% of shortest path in terms of path length, and significantly minimize the disruption cost. In addition, RR-Demand satisfies all the flows but its congestion factor can increase by a factor of two, whereas RR-Demand loses 5% of the demands when the number of demand pairs is 24.

5.5.2 Asynchronous Updates

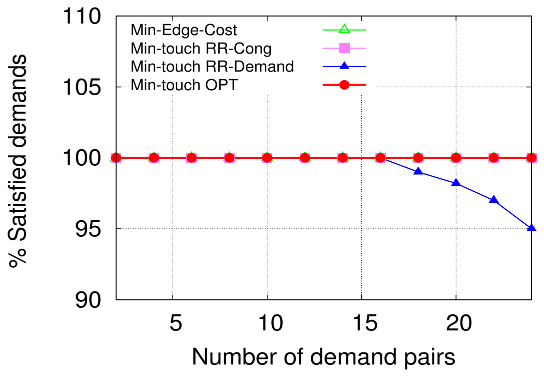
In this section, I use the same topology as the previous experiment with the same units of flows. Similar to the previous experiment, I increase the number of demand pairs from 2 to 24 and compare (i) total number of hops, (ii) Congestion factor, (ii) demand loss, and (iv) objective function in Min-Edge-Cost, RR-Cong, RR-Demand and OPT for the asynchronous update approach. Figure 5.6 shows the experimental results for this scenario. It is observed the number of disrupted flows in Min-Edge-Cost is higher than other algorithms by a factor of two. In addition, while the number of disrupted flows in RR-Cong and RR-Demand are close to the optimal, they have either a higher congestion factor or less satisfied demands.



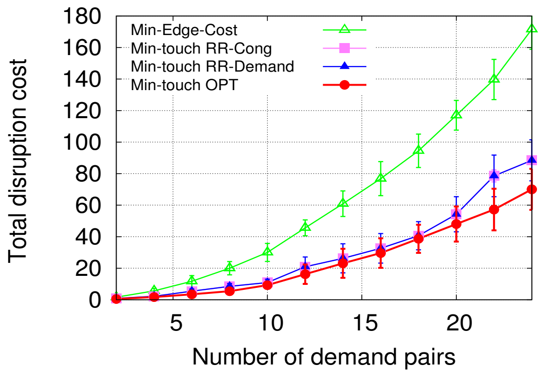
(a) Total number of hops.



(b) Congestion factor.



(c) Satisfied demands.

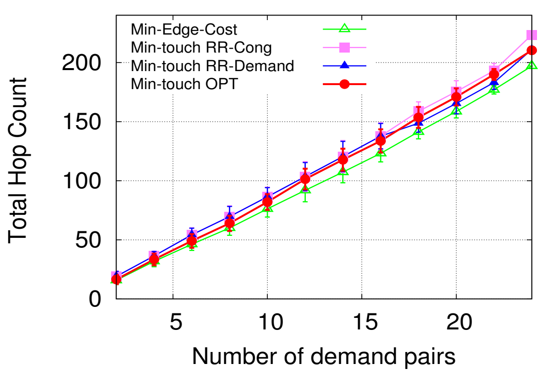


(d) Objective function.

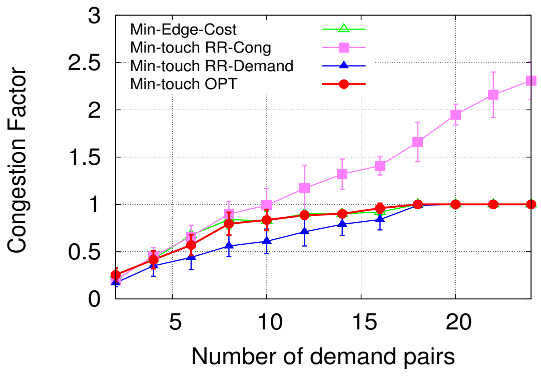
Figure 5.5: Comparison of Min-Edge-Cost (baseline), RR-Cong, RR-Demand (proposed), and OPT (super-polynomial time): BellCanada topology, synchronous update.

5.5.3 Quality of Service Constraint

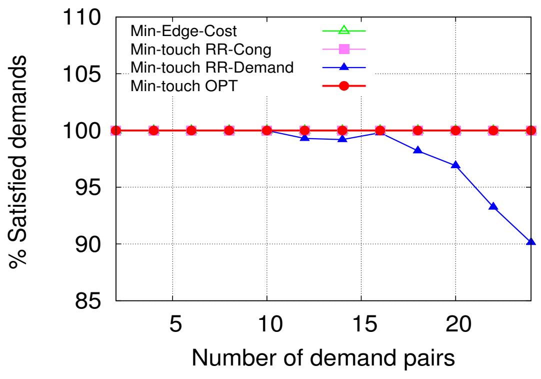
I next evaluate the impact of QoS constraint on the average number of hops. I use the CAIDA topology with 20 demand pairs where the flow rate of each demand pair is set to 3, and each link's capacity is chosen randomly in the interval [20, 50]. I first find the minimum number of hops to route each flow using shortest path routing and increase the QoS deviation from shortest path from 10% to 60%. Figure 5.7 shows the experimental results for this scenario for both the synchronous and asynchronous update approach. As shown, by relaxing the QoS constraint the total number of hops increases while the straggler's update time decreases. Therefore, by changing the QoS threshold, I can configure my choice of trade-offs



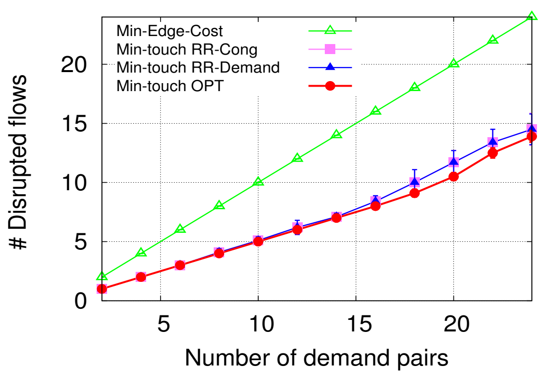
(a) Total number of hops.



(b) Congestion factor.



(c) Satisfied demands.



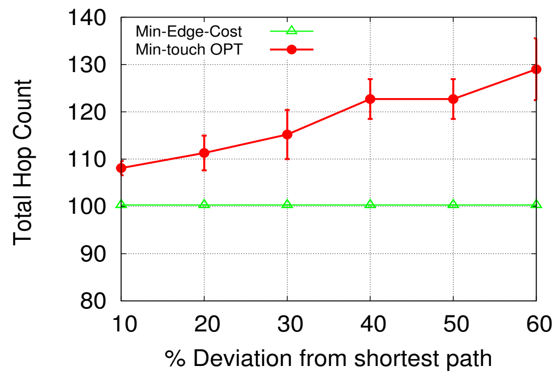
(d) Objective function.

Figure 5.6: Comparison of Min-Edge-Cost (baseline), RR-Cong, RR-Demand (proposed), and OPT (super-polynomial time): BellCanada topology, asynchronous update.

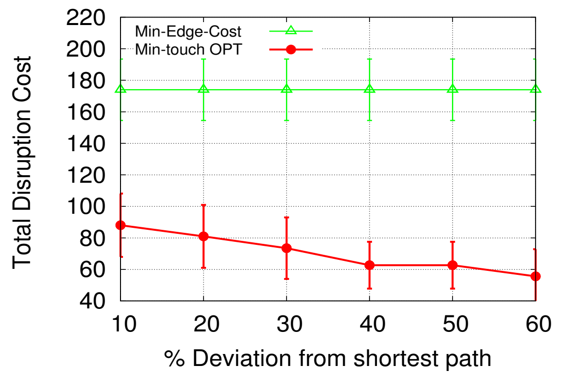
between having shorter paths or lower disruption cost.

5.5.4 Prioritizing Short-lived flows

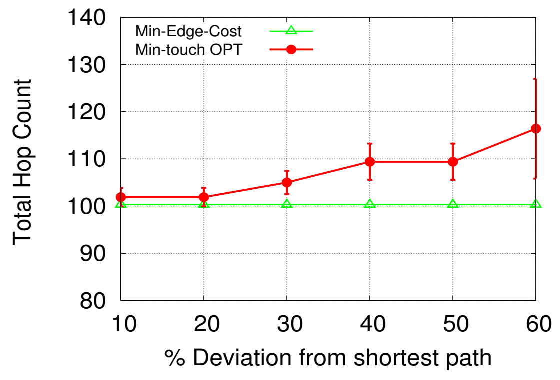
In the next set of experiments, I consider a network with two types of flows: (i) short-lived flows, and (ii) long-lived flows. According to [131], 80% of the flows in a 1500-node production data center are short-lived flows that last less than ten seconds, and 20% are long-lived. I used the same data and assume that on average, the flow duration (τ_h) of long-lived flows is 100 times larger than short-lived flows, i.e. $\tau_h^{long-lived} = 100\tau_h^{short-lived}$. I assume the priority of each flow to be a decreasing function of flow duration, $w_h = 1/\tau_h$.



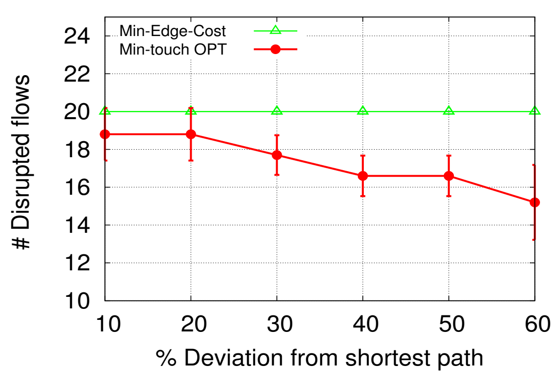
(a) Total number of hops (synchronous).



(b) Disruption cost (synchronous).



(c) Total number of hops (asynchronous).



(d) Disrupted flows (asynchronous).

Figure 5.7: Trade-off between hop count and disruption cost/number of disrupted flows as I increase the QoS threshold in synchronous and asynchronous update approach: AS28717 topology.

I use the BellCanada topology where each link's capacity is chosen randomly in the interval [20, 50]. I consider 20 demand pairs where the flow rate of each demand pair is set to 5. Figure 5.8 shows the percentage of disrupted short-lived and long-lived flows for the synchronous and asynchronous update problem. As shown, by prioritizing the short-lived flows, the algorithm disrupts fewer short-lived flows in the both synchronous and asynchronous approaches.

5.5.5 Update Times

In this section, I use the same topology as the previous experiment with the same number of old and new flows where the flow rate of each demand pair is set to 3. I

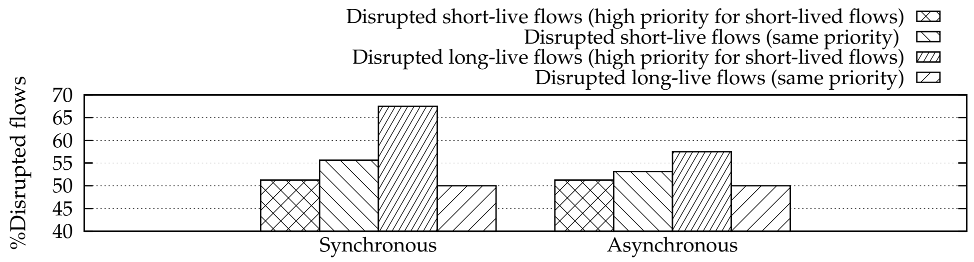


Figure 5.8: Percentage of disrupted flows in the Asynchronous and Synchronous update problem, BellCanada topology.

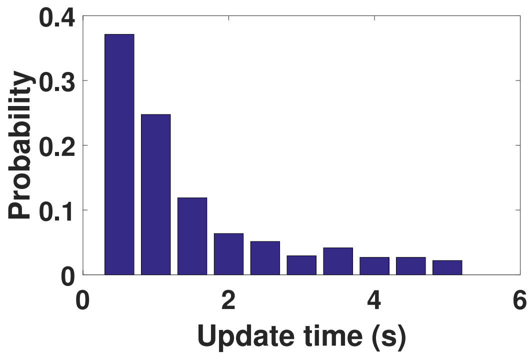


Figure 5.9: Distribution of update times for all switches in the synchronous update approach, BellCanada topology.

consider the update time among all switches in the synchronous update approach. Figure 5.9 shows the probability distribution function of the update time among all switches, i.e. $\sum_{h\in H} t_i^h \cdot T \; \forall i \in V$, where T is the update time for installing a single rule on the evaluated Brocade SDN switch which is around 500 (ms). It is observed that half of the switches have very small update time while very few switches have higher update time. For the synchronous update approach, the slowest bottleneck switch severely increases the disruption time. Significant variations in rule update times are also reported in [119, 132].

5.6 Conclusion

This chapter studies one of the important limitations of software defined networking - fast and safe network reconfiguration with minimum disruptions to the existing flows. I formulate the problem as an integer linear programming and show that it is

NP-Hard. I considered two versions of the problem: (i) Min-touch synchronous, and (ii) Min-touch asynchronous, which provide different tradeoffs between consistency and update time. I conducted a thorough theoretical study of the Min-touch problem and proposed two efficient randomized rounding-based algorithms that solve the *Min-touch* problem with bounded approximation on congestion and demand loss. Experimental results on real network topologies demonstrated the effectiveness of the proposed approaches in terms of disruption cost, congestion and demand loss. The results indicate that my proposed approaches have a disruption cost close to the optimal while having a low congestion factor and a low demand loss. Further, by changing the QoS threshold, I can configure my choice of trade-offs between hop count and disruption cost.

Chapter 6

Conclusion and Future Work

In this chapter, I first summarize my research contributions toward exploring different modeling, monitoring, scheduling and re-configuration techniques for network recovery from massive failures. I then outline several interesting future directions and some of the challenges and open problems that is aimed to be further explored.

6.1 Summary of Contributions

In this dissertation, I provided comprehensive solutions to recover a network after massive disruption. I 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 were in conflict with each other and I studied the trade-off among them.

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

- **Chapter 2:** In this chapter, I proposed a progressive network recovery under uncertain knowledge of damages. The problem was formulated as a mixed integer linear programming (MILP) optimization 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. I proposed several algorithms to find a feasible solution set at each iteration of the algorithm. My results show that ISR outperforms the state-of-the-art ISP algorithm with a configurable choice of trade-off between the execution time, number of repairs and demand loss.

- **Chapter 3:** I showed that the inter-connectivity and dependency between different elements makes complex networks more vulnerable to failure [10]. I studied the inter-dependency between a power grid and a communication network. I proposed 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. I then formulated 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. The recovery phase aims at maximizing the restored accumulative flow. I showed that the recovery problem (*Max-R*) is NP-Hard and proposed heuristic recovery strategies that work under partial knowledge of damage locations. I proposed a consistent failure set algorithm (*CFS*) to locate the failures.

- **Chapter 4:** I studied the optimal selection of monitoring paths to balance identifiablity and cost [12]. To this end, I 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. I showed 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). I proposed an optimal greedy-based approach for (4) and proposed a

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

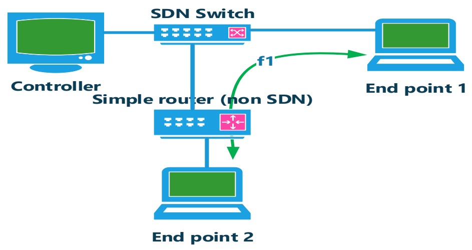
- **Chapter 5:** I studied a minimally disruptive update problem in software defined networks. I proposed two randomized rounding algorithms with bounded approximation on congestion and demand loss. Extensive experimental results on real network topologies shows the effectiveness of the proposed approach in terms of disruption cost, congestion and demand loss.

6.2 Future Directions

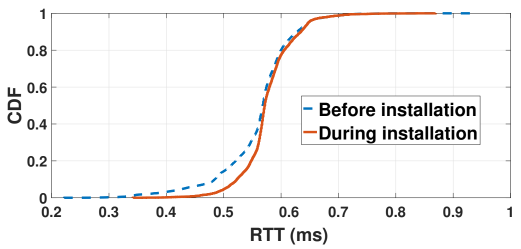
I provided several solutions to network recovery after large-scale failures in a communication network under uncertainty, and interdependent networks including a communication and a power grid. The recovery approach and failure detection mechanism with incomplete information is one of the first steps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

6.2.1 Power Grid and Communication Network Interdependency

My detection mechanism and recovery approach with incomplete information presented in chapter 3 opens up avenues of new research on improving power grid reliability and resiliency under incomplete or noisy information. Future research directions include: (1) more accurate power grid representation (e.g. AC power flow model), (2) more accurate communication network models for SCADA (e.g. including power-line carrier) and WAMPAC, and (3) mitigating the disruptions to electric power grids caused by malicious attacks.



(a) Implemented Testbed.



(b) Disruption.

Figure 6.1: Disruption caused by delaying the existing flow due to rule updates on a switch traversed by the existing flow.

6.2.2 Minimally Disruptive Network Updates

In chapter 5 I discussed a method to update flow rules in software defined networking that minimizes the disruption cost of re-routed flows. In addition to the disruption caused by re-routing existing flows, I also experienced another type of disruption which is the disruption caused by interrupting the existing flows due to rule updates on a switch traversed by the existing flow. For example, in a motivating example shown in Figure 6.1a, updating the SDN switch can also cause a delay to the existing flow f_1 which is not being re-routed, but traverses the updated switch.

Figure 6.1 shows a motivating experiment for this type of disruption. I first install rules on a 24-port Brocade (ICX 6610) SDN commercial switch and connect two end points. I then install 250 new rules on the SDN swicth and measure the round trip time (RTT) delays on the existing flow, $f1$, while installing the new rules. Figure 6.1 shows the cumulative distribution function of the RTT of the existing flow before and during installing the new rules.

Since this type of disruption is very small and does not always exist in my experiment and is very vendor-specific, I didn't consider it in my Min-touch problem formulation. However, as a future research direction, one can work on an optimization problem that minimizes both types of disruptions as follows:

$$\begin{aligned} \text{minimize} \quad & w_{high} \sum_{h \in H} \delta_h + w_{low} \sum_{h \in H} \theta_h \\ \text{subject to} \quad & \delta_h + \theta_h < 2, \quad \forall h \in H \qquad (6.1a) \end{aligned}$$

$$\sum_{h \in H} (x_{ij}^h + x_{ji}^h) \cdot d_h \leq c_{ij}, \ \ \forall (i,j) \in E \tag{6.1b}$$

$$\sum_{j \in V} x_{ij}^h = \sum_{k \in V} x_{ki}^h + sign(b_i^h), \ \ \forall i \in V, \ \ \forall h \in H \tag{6.1c}$$

$$\delta_h \geq \frac{\sum\limits_{(i,j) \in E} [x_{ij}^h + k_{ij}^h - 2k_{ij}^h x_{ij}^h]}{|E|} \ \ \forall h \in H \tag{6.1d}$$

$$\delta_h + \theta_h \geq \frac{\sum\limits_{j \in V} x_{ij}^h}{|V|} + t_i - 1, \ \ \forall i \in V, \forall h \in H \tag{6.1e}$$

$$t_i \geq \frac{\sum\limits_{j \in V} \sum\limits_{h \in H} [x_{ij}^h + k_{ij}^h - 2k_{ij}^h x_{ij}^h]}{|V||H|} \ \ \forall i \in V \tag{6.1f}$$

$$t_i \leq \sum_{j \in V} \sum_{h \in H} (x_{ij}^h + k_{ij}^h - 2k_{ij}^h x_{ij}^h), \ \ \forall i \in V \tag{6.1g}$$

$$t_i, x_{ij}^h, \delta_h, \theta_h \in \{0,1\}, \ \ \forall (ij) \in E, \ \ \forall h \in H \ \ \forall i \in V \tag{6.1h}$$

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 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 Section 5.1, I 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 6.1b 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 6.1c 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 6.1d ensures that δ_h is set to 1 when flow h is re-routed. Constraint 6.1e together with the objective ensures that θ_h will be 1 if flow h is not re-routed but crosses and updated switch. Whenever it is possible to choose between the two different categories of "affected" flows, the rerouted flows (captured by $\delta_h = 1$) and the non-rerouted, crossing altered switches flows (captured by $\theta_h = 1$), the objective function will set θ_h to 1, due to the lower weight in the linear combination for the minimization of the number of affected flows.

6.3 Summary

This chapter first summarizes the main contributions of this dissertation and then outlines several interesting open problems and ongoing research.

Bibliography

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

[2] 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*.

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

[4] 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.

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

[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., N. Bartolini, H. Khamfroush, T. He, N. R. Chaudhuri, and T. La Porta (2018) "Mitigation and Recovery from Cascading Failures in Interdependent Networks under Uncertainty," *IEEE Transactions on Control of Network Systems.*

[12] 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).*

[13] ——— (2018) "Parsimonious tomography: Optimizing cost-identifiability trade-off for probing-based network monitoring," *ACM SIGMETRICS Performance Evaluation Review.*

[14] 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.

[15] (September 13, 2017) "Communications Status Report for Areas Impacted by Hurricane Irma," *Federal Communications Commission (FCC).*

[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] 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).*

[18] 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.

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

[20] Todimala, A. and B. Ramamurthy (2006) "Approximation algorithms for survivable multicommodity flow problems with applications to network design," in *Proceedings IEEE INFOCOM.*

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

[22] 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.

[23] ——— (2011) "Risk of attack coefficient effect on availability of Ad-hoc networks," in *Consumer Communications and Networking Conference (CCNC)*, IEEE.

[24] Farhat, F., M. R. Pakravan, M. Salmasizadeh, and M. R. Aref (2010) "Locally multipath adaptive routing protocol resilient to selfishness and wormholes," in *International Conference on Information Security Practice and Experience*, Springer.

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

[26] Barford, P., N. Duffield, A. Ron, and J. Sommers (2009) "Network performance anomaly detection and localization," in *Proceedings IEEE INFOCOM*.

[27] Bartolini, N., T. He, and H. Khamfroush (2017) "Fundamental limits of failure identifiability by boolean network tomography," in *Proceedings IEEE INFOCOM*.

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

[29] Hojjatinia, S., K. Bekiroglu, and C. M. Lagoa (2018) "Parsimonious Volterra System Identification," *arXiv preprint arXiv:1804.07239*.

[30] Ravazzi, C., S. Hojjatinia, C. M. Lagoa, and F. Dabbene (2018) "Randomized opinion dynamics over networks: influence estimation from partial observations," *arXiv preprint arXiv:1804.07220*.

[31] Hojjatinia, S., C. M. Lagoa, and F. Dabbene (2017) "A Method for Identification of Markovian Jump ARX Processes," *IFAC-PapersOnLine*.

[32] ——— (2018) "Identification of Switched ARX Systems from Large Noisy Data Sets," *arXiv preprint arXiv:1804.07411*.

[33] Tootaghaj, D. Z., N. Bartolini, H. Khamfroush, and T. La Porta (2017) "Controlling Cascading Failures in Interdependent Networks under Incomplete Knowledge," in *SRDS*, IEEE.

[34] 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.

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

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

[37] Tootaghaj, D. Z., A. Sampson, T. Mytkowicz, and K. S. McKinley (2017) "High Five: Improving Gesture Recognition by Embracing Uncertainty," *arXiv preprint arXiv:1710.09441*.

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

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

[40] 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.

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

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

[43] Gao, J., S. V. Buldyrev, H. E. Stanley, and S. Havlin (2012) "Networks formed from interdependent networks," *Nature physics*.

[44] 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.

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

[46] 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.

[47] 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.*

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

[49] 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.

[50] Sen, A., A. Mazumder, J. Banerjee, A. Das, and R. Compton (2014) "Identification of k most vulnerable nodes in multi-layered network using a new model of interdependency," in *IEEE INFOCOM WKSHPS).*

[51] 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.

[52] 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.

[53] 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).*

[54] 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.*

[55] 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.*

[56] 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.*

[57] 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.

[58] Bertsch, J., C. Carnal, D. Karlson, J. McDaniel, and K. Vu (2005) "Wide-area protection and power system utilization," *Proceedings of the IEEE*.

[59] Begovic, M., D. Novosel, D. Karlsson, C. Henville, and G. Michel (2005) "Wide-area protection and emergency control," *Proceedings of the IEEE*.

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

[61] 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.

[62] Soltan, S., D. Mazauric, and G. Zussman (2017) "Analysis of failures in power grids," *IEEE Transactions on Control of Network Systems*.

[63] Soltan, S. and G. Zussman (2017) "Power Grid State Estimation after a Cyber-Physical Attack under the AC Power Flow Model," *Proc. IEEE PES-GM*, **17**.

[64] Bienstock, D. (2016) *Electrical Transmission System Cascades and Vulnerability: An Operations Research Viewpoint*, vol. 22, SIAM.

[65] Shao, J., S. V. Buldyrev, S. Havlin, and H. E. Stanley (2011) "Cascade of failures in coupled network systems with multiple support-dependence relations," *Physical Review E*.

[66] Huang, Z., C. Wang, S. Ruj, M. Stojmenovic, and A. Nayak (2013) "Modeling cascading failures in smart power grid using interdependent complex networks and percolation theory," in *IEEE ICIEA*.

[67] Huang, X., J. Gao, S. V. Buldyrev, S. Havlin, and H. E. Stanley (2011) "Robustness of interdependent networks under targeted attack," *Physical Review E*.

[68] 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*.

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

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

[71] Andersson, G. (2004) "Modelling and analysis of electric power systems," *EEH-Power Systems Laboratory, Swiss Federal Institute of Technology (ETH), Zürich, Switzerland.*

[72] Biggs, N. (1993) *Algebraic graph theory*, Cambridge university press.

[73] 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.

[74] 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.*

[75] 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.

[76] 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.*

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

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

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

[80] 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.

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

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

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

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

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

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

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

[88] 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)*.

[89] 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*.

[90] Xie, Q., A. Yekkehkhany, and Y. Lu (2016) "Scheduling with multi-level data locality: Throughput and heavy-traffic optimality," in *The 35th Annual IEEE International Conference on Computer Communications (INFOCOM)*.

[91] Yekkehkhany, A., A. Hojjati, and M. H. Hajiesmaili (2018) "GB-PANDAS:: Throughput and heavy-traffic optimality analysis for affinity scheduling," *ACM SIGMETRICS Performance Evaluation Review*.

[92] Yekkehkhany, A. (2017) "Near data scheduling for data centers with multi levels of data locality," *arXiv preprint arXiv:1702.07802*.

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

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

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

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

[97] 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*.

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

[99] Strang, G. (2011) “Introduction to linear algebra,” .

[100] “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.

[101] 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*.

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

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

[104] 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.*

[105] 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.

[106] 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.

[107] Ludwig, A., S. Dudycz, M. Rost, and S. Schmid (2016) “Transiently Secure Network Updates.” in *Sigmetrics*.

[108] 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*.

[109] "OpenFlow Switch Specification." `https://www.opennetworking.org/` .

[110] Vissicchio, S. and L. Cittadini (2016) "Flip the (flow) table: Fast lightweight policy-preserving sdn updates," in *INFOCOM*, IEEE.

[111] Mizrahi, T. and Y. Moses (2016) "Software defined networks: It's about time," in *INFOCOM*, IEEE.

[112] Reitblatt, M., N. Foster, J. Rexford, and D. Walker (2011) "Consistent updates for software-defined networks: Change you can believe in!" in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, ACM.

[113] Lazaris, A., D. Tahara, X. Huang, E. Li, A. Voellmy, Y. R. Yang, and M. Yu (2014) "Tango: Simplifying SDN control with automatic switch property inference, abstraction, and optimization," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, ACM.

[114] Mahajan, R. and R. Wattenhofer (2013) "On consistent updates in software defined networks," in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, ACM.

[115] Canini, M., P. Kuznetsov, D. Levin, and S. Schmid (2015) "A distributed and robust sdn control plane for transactional network updates," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*, IEEE.

[116] Dudycz, S., A. Ludwig, and S. Schmid (2016) "Can't touch this: Consistent network updates for multiple policies," in *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, IEEE.

[117] Amiri, S. A., A. Ludwig, J. Marcinkowski, and S. Schmid (2016) "Transiently consistent sdn updates: Being greedy is hard," in *International Colloquium on Structural Information and Communication Complexity*, Springer.

[118] Vissicchio, S., O. Tilmans, L. Vanbever, and J. Rexford (2015) "Central control over distributed routing," *ACM SIGCOMM Computer Communication Review.*

[119] Jin, X., H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer (2014) "Dynamic scheduling of network updates," in *ACM SIGCOMM Computer Communication Review*, ACM.

[120] Wen, X., C. Diao, X. Zhao, Y. Chen, L. E. Li, B. Yang, and K. Bu (2014) "Compiling minimum incremental update for modular SDN languages," in *Proceedings of the third workshop on Hot topics in software defined networking*, ACM.

[121] Xu, H., Z. Yu, X. Li, C. Qian, L. Huang, and T. Jung (2016) "Real-time update with joint optimization of route selection and update scheduling for SDNs," in *IEEE ICNP*.

[122] Katta, N. P., J. Rexford, and D. Walker (2013) "Incremental consistent updates," in *HotSDN*, ACM.

[123] Wang, W., W. He, J. Su, and Y. Chen (2016) "Cupid: Congestion-free consistent data plane update in software defined networks," in *INFOCOM*, IEEE.

[124] Farhat, F., D. Tootaghaj, Y. He, A. Sivasubramaniam, M. Kandemir, and C. Das (2016) "Stochastic modeling and optimization of stragglers," *IEEE Transactions on Cloud Computing*.

[125] Tootaghaj, D. Z., F. Farhat, M. Arjomand, P. Faraboschi, M. T. Kandemir, A. Sivasubramaniam, and C. R. Das (2015) "Evaluating the combined impact of node architecture and cloud workload characteristics on network traffic and performance/cost," in *Workload characterization (IISWC), 2015 IEEE international symposium on*, IEEE, pp. 203–212.

[126] Tootaghaj, D. Z. (2015) "Evaluating cloud workload characteristics," *Master thesis*.

[127] Farhat, F., D. Z. Tootaghaj, A. Sivasubramaniam, M. T. Kandemir, and C. R. Das (2014) *Modeling and optimization of straggling mappers*, *Tech. rep.*, Technical report, Technical Report CSE-14-006, Pennsylvania State University.

[128] Farhat, F., D. Z. Tootaghaj, and M. Arjomand (2016) "Towards Optimizing Data Computing Flow in the Cloud," *arXiv preprint arXiv:1607.04334*.

[129] 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 Networking*.

[130] Hoeffding, W. (1963) "Probability inequalities for sums of bounded random variables," *Journal of the American statistical association*.

[131] Kandula, S., S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken (2009) "The nature of data center traffic: measurements & analysis," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, ACM.

[132] Ferguson, A. D., A. Guha, C. Liang, R. Fonseca, and S. Krishnamurthi (2013) "Participatory networking: An API for application control of SDNs," in *ACM SIGCOMM computer communication review.*

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

[134] 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).*

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

[136] Chekuri, C., S. Khanna, and F. B. Shepherd (2006) "An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow," *Theory of computing.*

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.