The Pennsylvania State University
The Graduate School

TOWARDS SECURE AND PERFORMANT CYBER-PHYSICAL SYSTEMS: ESTIMATION, OPTIMIZATION, AND CONTROL

A Dissertation in
Computer Science and Engineering
by
Yudi Huang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

May 2024

The dissertation of Yudi Huang was reviewed and approved by the following:

Ting He
Associate Professor of the School of Electrical Engineering and Computer Science
Dissertation Advisor
Chair of Committee

Thomas F. La Porta
Evan Pugh Professor
William E. Leonhard Professor of the School of Computer Science and Engineering and Electrical Engineering

Nilanjan Ray Chaudhuri
Associate Professor of the School of Electrical Engineering and Computer Science

Uday V. Shanbhag
Gary and Sheila Bello Chair and Professor in the Harold and Inge Marcus Department of Industrial and Manufacturing Engineering

Chitaranjan Das
Program Head
Distinguished Professor of Computer Science and Engineering

Abstract

Cyber-physical systems (CPSs) have risen to prominence across multiple domains, including power grids, drone networks and transportation systems, owing to their ability to significantly enhance these applications through the integration of modern communication networks. Due to the existence of legacy devices and complex couplings of cyber-physical spaces, secure and efficient management of CPSs imposes formidable challenges. This dissertation aims at systematically exploring the CPSs in the dimensions of *attack recovery*, *system resilience*, *routing control*, and *attack design*, through the lens of **estimation**, **optimization**, and **control**.

In our first piece of work, we study the link state inference problem for *attack recovery* in the smart grid through the lens of **estimation**. We consider a challenging attack form named the joint cyber-physical attacks, where the physical part disconnects links while the cyber part blocks the measurements from the attacked area. We try to fill the gap that the existing works always assume a connected post-attack grid by jointly inferring the link states and the power injection changes due to grid islanding using unattacked observations. We first propose a linear programming-based algorithm for link state estimation for the cases that islanding is possible. We develop a new analysis framework to characterize the accuracy of the proposed algorithm at a finer granularity than existing works. In addition, we extend the analysis into efficient polynomial-time algorithms to verify the correctness of the link state estimation. Finally, we extend the DC-based algorithms to AC model. The evaluations show that we can not only estimate the line states with low error rate, but also verify the most of them.

In our second piece of work, we study a new trade-off between *system resilience* and limited defending budget in smart grid, through the lens of **optimization** of the secured PMU deployment. Specifically, instead of preventing the existence of all attacks through achieving full observability by secured PMUs, we propose to deploy secured PMUs to *limit the attack impact* so that any undetectable false data injection cannot cause overload-induced link tripping. We formulate the problem as a tri-level mixed-integer optimization problem. Then, we propose an alternating optimization framework together with two constraint generation algorithms to solve PPOP optimally. For large grids, we further develop a scalable polynomial-time heuristic algorithm. Finally, we extend our solution to achieve the defense goal under AC power flow model. The numerical evaluations demonstrate that our algorithms can significantly reduce the required number of PMUs.

In our third piece of work, we aim to design *performant* overlay communication

networks for CPSs through the lens of network state **estimation** and routing **control**. Considering the demand of timely delivery of measurements and commands in CPSs, and the trend of communication network virtualization, our objective is to achieve congestion-free overlay routing over an uncooperative underlay network. To this end, we first identified the minimum information necessary for such routing and introduced the first-known polynomial-complexity algorithms to infer this information with guaranteed accuracy. For the special case of symmetric tree-based routing, we developed an alternative algorithm to enhance the inference accuracy. Subsequently, we devised a greedy algorithm to estimate the required capacity information. Numerical evaluations, conducted using the NS3 simulator, demonstrate the effectiveness of our proposed algorithms in avoiding congestion.

In our fourth piece of work, we examined the *attack design* of cross-path attacks in communication networks, through the lens of **estimation** of the shared network components and **optimization** of the attack's impact on the targeted services. We initially developed novel reconnaissance algorithms to consistently reveal the locations and parameters of the shared links through network tomography. Subsequently, we explored two optimization objectives for allocating attack rates to maximally degrade the performance of targeted paths. Extensive evaluations demonstrate that our proposed algorithms can achieve a significantly larger performance impact compared to their non-optimized counterparts.

In the ongoing work, we propose to develop a *performant* decentralized learning framework for CPSs, through the lens of **optimization** of overlay routing and mixing matrix. This endeavor is driven by the goal of enhancing CPSs with machine learning capabilities and the necessity of distributing the training process across decentralized agents. We aim to provide a strategy to minimize the total training time by balancing the trade-off between the per-iteration communication time and the number of iterations for convergence.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Ting He, for her continuous encouragement and selfless help during my Ph.D. study at Penn State. She not only directly teaches me with passion and patience, but also guides me to be a good researcher as a role model. I am really grateful for her help with everything although she is so busy. She is the best advisor I can imagine.

Besides, I would like to thank Prof. Nilanjan Ray Chaudhuri and Prof. Tom La Porta as my co-authors and committee members. I really appreciate their time, help, and also criticism. I would also like to thank Prof. Uday V. Shanbhag for being my committee member. Thanks for the time and help.

I am grateful to my teammates Hanlin Lu, Yilei Lin, Cho-Chun (Daniel) Chiu, Tian Xie, Akash Kumar, and Tingyang Sun. You make my study much more easier and interesting. I am also grateful to my other friends Zhikun Lei, Bo Pan, Dongrui Zeng, Shen Liu, Huaiyuan Zhang, Zhexi Zhu, Junxin Wang, Jie Chen, Junjie Tan, Qian Shi, Ke Liang, Xusheng Zhang, Xiaodong Jia, Jian Zhao, Sitao Zhang, etc. It is great to have them in my life. I want to give my sincerest thanks to PeiJhen Gong. It is one of the luckiest things to have her company during my Ph.D. years.

Last, I would like to thank my parents, Qiang Huang and Qingmei Zhang, for their support. They not only give me birth but also build my personality. Through the best and the worst, through the difficult and the easy, they always respect my thoughts and give me full support. I would also like to give sincere thanks to my grandparents Bairong Zhang and Jinmei Shen, and my aunt Li Zhang for the unconditional love you share with me.

The work in this dissertation was generously supported by the National Science Foundation under awards ECCS-1836827, CNS-1946022, and CNS-2106294. Any opinions, findings, and conclusions or recommendations expressed in this dissertation are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Chapter 1 Introduction

1.1 Background on Cyber-Physical Systems

Cyber-physical systems (CPSs) have widely been recognized as a promising paradigm for transforming industrial systems such as power grid [1], drone networks [2], transportation [3], and health care [4], etc. The opportunities of CPSs lie in the close integration of advanced sensing, communication, computation, and control with the physical components. The advanced communication networks in CPSs help achieve better monitoring and control in the physical spaces.

The current state-of-the-art in CPSs reflects a dynamic interplay between advanced technological innovations and practical applications across various domains. Recent advancements have seen the integration of cutting-edge technologies such as the Internet of Things (IoT), artificial intelligence (AI), and machine learning (ML) with CPSs, driving significant improvements in system responsiveness, adaptability, and efficiency [5]. For instance, in smart grids, AI-driven predictive models are being employed to enhance energy distribution and optimize renewable energy usage, reflecting a shift towards more sustainable and efficient energy systems [6].

Moreover, the advent of 5G technology and beyond promises to revolutionize CPSs by providing ultra-reliable low-latency communication (URLLC), which is crucial for applications requiring real-time data processing and control [7]. This technological leap is expected to unlock new possibilities in autonomous vehicle networks, remote healthcare, and smart cities, where seamless and instantaneous communication is paramount. The integration of these technologies within CPSs facilitates not only enhanced performance but also new functionalities that were previously unattainable, highlighting the evolving nature of CPS capabilities.

Looking ahead, the future of CPSs appears to be geared towards achieving higher

levels of autonomy, intelligence, and inter-connectivity. The convergence of AI and ML with CPSs is set to deepen, with systems becoming increasingly capable of autonomous decision-making, predictive analytics, and self-optimization. This evolution will likely usher in a new era of smart infrastructure, where CPSs can anticipate and respond to changes in their environment in a proactive and efficient manner. Additionally, the focus on enhancing cybersecurity measures and developing robust frameworks for data privacy and ethics in CPSs will remain a critical area of research and development, ensuring the trustworthiness and reliability of these systems.

In essence, the trajectory of CPSs points towards an increasingly interconnected and intelligent ecosystem of cyber and physical components. The ongoing advancements in technology and the continuous integration of these advancements into CPSs suggest a future where these systems play a central role in driving innovation, efficiency, and sustainability across all sectors of society. The challenge lies in harnessing these technologies in a manner that maximizes their potential while mitigating the associated risks, particularly those related to security and privacy.

1.2 Motivations

However, the great opportunities come with greater challenges in system design and management [8], among which we focus on the **secure** and **efficient** management in this dissertation.

The inherent complexity of CPSs arises from their tightly coupled cyber and physical components, which necessitate sophisticated algorithms for real-time data processing and decision-making [9]. This complexity is further magnified by the need for robust measures to defend against increasingly sophisticated attacks. As such, ensuring the security and integrity of CPSs is paramount, particularly given their critical role in infrastructure and their potential impact on public safety and economic stability [10]. The integration of advanced sensing and control technologies enables CPSs to respond dynamically to changing environmental conditions and operational demands, thereby enhancing efficiency and sustainability.

As discussed in [11], securing CPSs requires a systematical defending mechanism, which can be decomposed into four components, as shown in Fig. 1.1. The first component is *prevention*, which aims at postponing the attack onset by reducing information leakage and detecting intrusion. However, there always exist attacks that can bypass the prevention, which leads to the requirements of *resilience*. A resilient CPS is still

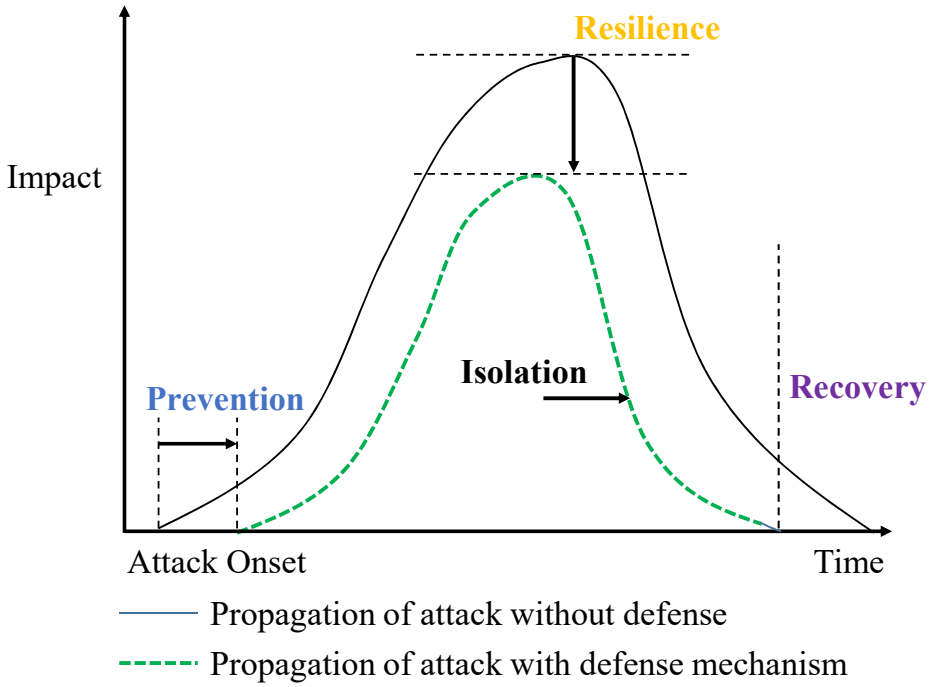


Figure 1.1. A systematical securing solution for CPSs.

functional (in an abnormal state without being broken) under attack by limiting the attack's impact. After observing the existence of attacks that have bypassed the detection mechanism, we need to *isolate* the attacks for further actions. Finally, *recovery* of the system back into its normal state is vital for maintaining a stable system.

Furthermore, the scalability of CPSs presents both a challenge and an opportunity for the design of its communication networks [12]. These networks are crucial for facilitating real-time data exchange and coordination, which become increasingly complex as systems grow. Addressing scalability involves ensuring network robustness against disruptions, achieving low latency for immediate system responsiveness, and securing networks against cyber threats, notably Denial of Service (DoS) attacks [13]. Overlay routing design is particularly significant in this context, offering a strategic approach to optimize communication pathways, thereby enhancing efficiency and resilience [14]. Through adeptly tackling these design challenges, communication networks play a vital role in enabling CPSs to scale effectively, ensuring reliable performance across a spectrum of demands and conditions.

The evolution of CPSs is also driven by advancements in artificial intelligence (AI) and machine learning (ML), which offer the potential to significantly enhance decision-making processes within these systems [15]. By leveraging AI and ML, CPSs can learn from data,

adapt to new situations, and predict future states, thereby improving their efficiency, reliability, and overall performance. However, the integration of AI and ML into CPSs introduces additional layers of complexity, particularly in terms of ensuring the security and efficiency.

In conclusion, the development and deployment of CPSs hold the promise of revolutionizing not only traditional industrial systems but also extending beyond them. The challenges associated with their complexity, scalability, security, and the integration of AI and ML, while significant, are not insurmountable. Overcoming these challenges is both urgent and crucial for realizing the full potential of CPSs.

1.3 An Illustrative Example: Smart Grid

In this subsection, we use the smart grid as an example, as shown in Fig. 1.2, to demonstrate the problems we plan to solve. In Fig. 1.2, the power grid is the physical space to manage, where nodes are connected through power lines. Together with switches and routers, the sensors for monitoring the power grid are connected in the communication layer so that the control center can collect the sensor measurements for further actions such as state estimation and load forecasting. It is worth noting that a large portion of nodes (e.g., routers) for networking may be provided by the third party (e.g., Internet Service Provider, ISP) and thus invisible to the grid operator. We will call the communication network managed by the grid operator the *overlay* since it is layered on the *underlay* ISP's network. That is to say, a direct communication link from the perspective of overlay may be composed of multiple underlay links.

The security threat to smart grid has long been an active research area. For example, adversary may attack the power grid for manipulating the market to obtain a financial gain [16]. The adversary may also be motivated by causing large-scale blackout [17], such as the attacks on the Ukrainian power grid in 2015 [18]. Due to the severe consequences of destabilization, secure control of the CPSs to stay stable is deemed necessary.

The joint cyber-physical attacks as shown in Fig. 1.2 has attracted lots of attention. Specifically, the physical attack refers to the disconnection of links, while the cyber attack can be either information blockage (also named denial-of-service attack) [19] or false data injection [20]. According to [11], a comprehensive solution for CPS security requires *attack prevention*, *impact control*, *attack detection* and *attack recovery*. The first problem in this dissertation to solve is the *attack recovery*, which requires post-attack situation awareness. Since the topological information, i.e., the link states, is vital for attack

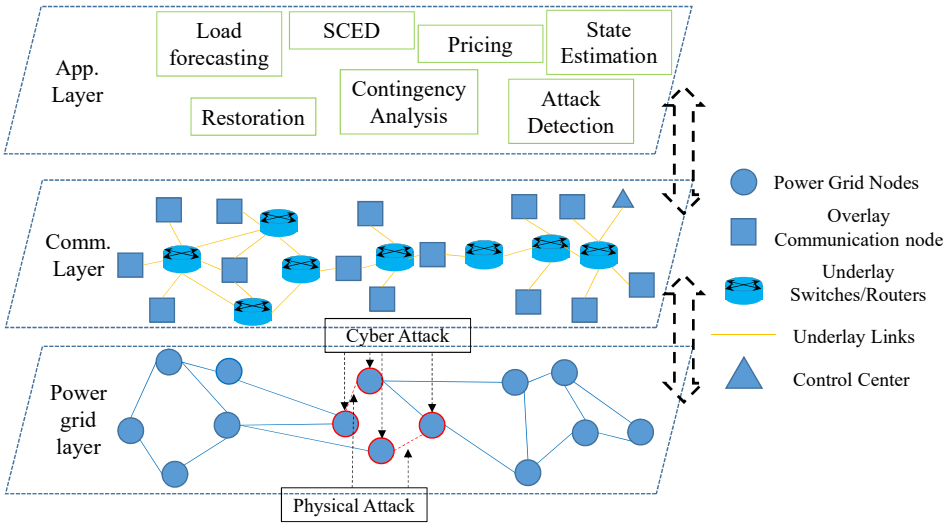


Figure 1.2. An example of smart grid network

recovery, we will study the **link state inference** after joint cyber-physical attacks.

The close integration of advanced networking system [21] introduce new challenges, such as false data injection [20]. The second problem to study is the trade-off between limited security budget and *system resilience* during the attack. It has long been realized that advanced sensing equipment, such as phasor measurement units (PMUs), can help defend against the attacks since they are not subjected to data injections. Many existing works have studied optimal PMU deployment achieving full observability of the grid to eliminate the existence of attacks. However, the high cost prevents PMU from being fully deployed to achieve full observability in the short future, which results in a need for a trade-off between full prevention and impact control under budget constraints. Thus, we propose a new trade-off as well as the associated **optimal deployment** of PMUs under the new defending goal.

The timely transmission of measurements from sensors and commands from the control center is crucial for maintaining the controllability of smart grids [22], which is essential for their efficiency and resilience. The virtualization and softwarization make the congestion-free overlay routing design a challenging problem. Therefore, the third problem addressed in this dissertation concentrates on the **optimal control** of overlay routing over an uncooperative underlay network. One major challenge in operating overlay networks lies in the uncooperative nature of the underlay networks, which implies no direct information of the underlay network. Such lack of information results in possible sub-optimal performance of the overlay applications due to the black-box optimization. We will study the necessary information for achieving congestion-free overlay routing

and algorithms to estimate this information.

The trend of network softwarization and virtualization has fundamentally altered the way we build network systems, and makes modern communication networks appear easier to manage. However, the complex interactions within the networks open the door for new security threats. Thus, in the fourth problem in this dissertation, we study a particular type of DoS attacks, called cross-path attacks, which stem from the sharing of links between high-priority paths and low-priority paths. We will explore the impact of this security threat on the performance degradation of the target service, which is the high-priority communications for CPSs.

Due to the potential of significantly improving the system efficiency, introducing machine learning into CPSs has been widely deemed promising. However, traditional centralized training paradigm based on parameter server may not be suitable for CPSs due to the data privacy requirements. In addition, centralized server is likely to be the communication bottleneck and thus slows down the training. In the ongoing work, we will study the decentralized learning over overlay networks. The objective is to minimize the training time without causing congestion.

1.4 Roadmap and Research Problems

This dissertation has four main chapters addressing four different problems, focusing on*attack recovery*, *system resilience*, *routing control*, and *attack design* problems of the CPSs, through the lens of **estimation**, **optimization**, and **control**. Specifically:

1. Effective defense against cyber-physical attacks in power grids requires accurate damage assessment. While some solutions have been proposed to recover the link states within the attacked area, existing solutions are limited by the assumption of a connected post-attack grid and the lack of verifiable performance guarantees. To fill this gap, we study in Chapter 2 the recovery of link states under a cyber-physical attack that disconnects links within the attacked area while blocking information from that area. In contrast to existing solutions assuming a connected post-attack grid or known post-attack power injections, we consider a more challenging scenario where the attack may partition the grid into islands, which causes unknown changes in power injections. To address this problem, we (i) propose a linear programming-based algorithm to recover the link states within the attacked area under unknown post-attack power injections, (ii) characterize the accuracy of the proposed recovery algorithm under certain conditions, and (iii) develop efficient algorithms to verify

the recovery results using observable information. Our numerical evaluations based on the IEEE 300-bus system and the Polish grid demonstrate that the proposed recovery algorithm is highly accurate in localizing failed links, most of which can be successfully verified by the proposed verification algorithms.

2. Next, we consider the optimal PMU deployment for defending against coordinated cyber-physical attacks to prevent severe consequences in Chapter 3. Existing approaches focus on eliminating the existence of attacks by either securing existing sensors or deploying secured PMUs. In this work, we improve this approach by lowering the defense target from *eliminating attacks* to *preventing outages* and reducing the required number of PMUs. To this end, we formulate the problem of *PMU Placement for Outage Prevention (PPOP)* as a tri-level non-linear optimization problem and transform it into a bi-level mixed-integer linear programming (MILP) problem. Then, we propose an alternating optimization framework to solve PPOP by iteratively adding constraints, for which we develop two constraint generation algorithms. In addition, for large-scale grids, we propose a polynomial-time heuristic algorithm to obtain suboptimal solutions. Finally, we evaluate our algorithm on IEEE 30-bus, 57-bus, 118-bus, and 300-bus systems, which demonstrates the potential of the proposed approach in greatly reducing the required number of PMUs.

3. In Chapter 4, we study the optimal overlay routing control over an uncooperative underlay for timely delivery of measurements and control commands. Overlay network is a non-intrusive mechanism to enhance the existing network infrastructure by building a logical distributed system on top of a physical underlay. A major difficulty in operating overlay networks is the lack of cooperation from the underlay, which is usually under a different network administration. In particular, the lack of knowledge about the underlay topology and link capacities makes the design of efficient overlay routing extremely difficult. In contrast to existing solutions for overlay routing based on simplistic assumptions such as known underlay topology or disjoint routing paths through the underlay, we aim at systematically optimizing overlay routing without causing congestion, by extracting necessary information about the underlay from measurements taken at overlay nodes. To this end, we (i) identify the minimum information for congestion-free overlay routing, and (ii) develop polynomial-complexity algorithms to infer this information with guaranteed accuracy. Our evaluations in NS3 based on real network topologies demonstrate

notable performance advantage of the proposed solution over existing solutions.

4. Then, we consider the impact of a particular DoS attack paradigm on modern communication networks in Chapter 5. While softwarization and virtualization technologies make modern communication networks appear easier to manage, they also introduce highly complex interactions within the networks that can cause unexpected security threats. In this work, we study a particular security threat due to the sharing of links between high-security paths and low-security paths, which enables a new type of DoS attacks, called cross-path attacks, that indirectly attack a set of targeted high-security paths (target paths) by congesting the shared links through a set of attacker-controlled low-security paths (attack paths). While the feasibility of such attacks has been recently demonstrated in the context of SDN, their potential performance impact has not been characterized. To this end, we develop an approach for designing an optimized cross-path attack under a constrained total attack rate, consisting of (i) novel reconnaissance algorithms that can provide consistent estimates of the locations and parameters of the shared links via network tomography, and (ii) efficient optimization methods to design the optimal allocation of attack rate over the attack paths to maximally degrade the performance of the target paths. The proposed attack has achieved a significantly larger performance impact than its non-optimized counterparts in extensive evaluations based on multiple network settings, signaling the importance of addressing such intelligent attacks in network design.

5. At last in Chapter 6, we will discuss the limitations and potential extensions of the works in this dissertation. Furthermore, we will demonstrate some details of optimal overlay network design for decentralized learning as an illustrative example. Since the training time is determined by both the per-iteration communication delay and the number of interactions for convergence, we propose to find the optimal trade-off between them to minimize the total training wall-clock time.

Chapter 2
Link State Estimation under Cyber-Physical Attacks

2.1 Introduction

Modern power grids are interdependent cyber-physical systems consisting of a power transmission system (power lines, substations, etc.) and an associated control system (Supervisory Control and Data Acquisition - SCADA and Wide-Area Monitoring Protection and Control - WAMPAC) that monitors and controls the states of the power grid. This interdependency raises a legitimate concern: what happens if an attacker attacks both the physical grid and its control system simultaneously? The resulting attack, known as a *joint cyber-physical attack*, can cause large-scale blackouts, as the cyber attack can blindfold the control system and thus make the physical attack on the power grid more damaging. For example, one such attack on Ukraine's power grid left 225,000 people without power for days [23].

One of the challenges in dealing with such attacks is that in case the measurements (e.g., breaker status) within the attacked area are blocked by the cyber attacks, the control center is unable to accurately identify the damage caused by the physical attack (e.g., which lines are disconnected) and hence unable to efficiently schedule the repairing/restoration. To address this challenge, solutions [19, 24–27] have been proposed to recover the state and topology of the power grid inside the attacked area under either direct-current (DC) or alternating-current (AC) power flow models. However, existing works are based on the limiting assumption that either the grid remains connected after attack, or the post-attack power injections at all the buses are known, which is often violated by large-scale attacks. In addition, most of the existing solutions lack the ability to verify the correctness of the recovered state in real time, which can result in a costly

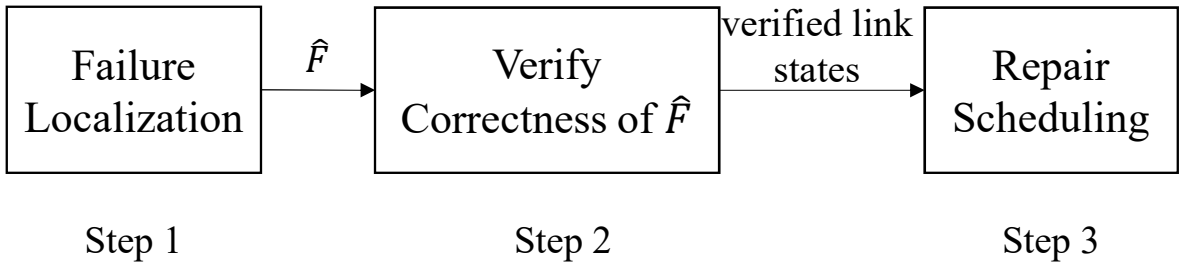


Figure 2.1. The defense system in smart grid. The proposed methods focus on **Topology Recovery** and **Topology Verification**.

waste of time and resources during restoration due to false alarms.

As shown in Fig. 2.1, the defense system in the smart grid is usually composed of multiple subsystems [11], which can be categorized into pre-attack and post-attack subsystems. The pre-attack subsystem aims to prevent the attack from taking effect. For example, the prevention subsystem will reduce the information leakage, while the detection subsystem intends to detect the invasion. For the advanced attacks that can bypass all pre-attack modules, efficient recovery from the attacks, which is the focus of the work, is desired. A key step before repairing/restoration is to learn the current topology through line state estimation. In this work, we address this problem under a joint cyber-physical attack, where the cyber attack blocks information from an attacked area while the physical attack disconnects lines (i.e., transmission lines) within the area, with a focus on scenarios where the physical attack causes islanding and hence unknown changes in the power injections within the attacked area. Our first goal is to compute an estimate $\hat{F}$ of the failed lines within the attacked area (**Topology Recovery**). Then, in contrast to existing works [24, 25, 28, 29], we add a novel step called **Topology Verification** before repairing/restoration, to guide resource dispatch during repairing/restoration by avoiding the cost due to false alarms.

2.1.1 Related Work

Power grid state estimation, as a key functionality for supervisory control, has been extensively studied in the literature [30, 31]. Secure state estimation under attack is of particular interest [20, 32]. Specifically, the attackers can launch denial-of-service attacks [11, 19] or false data injection attacks [19, 20, 26, 32, 33] so that the control center cannot correctly estimate the phase angles [34] or/and the topology [35] of the power grid. Recently, joint cyber-physical attacks are gaining attention due to their stealthiness and severe consequences [19, 26].

The resilience of the power grid to attacks requires both pre-attack prevention [36] and post-attack restoration, the latter being the focus of this work. To facilitate the restoration, several approaches have been proposed for detecting failed lines. In [28, 29], the problem was formulated as a mixed-integer program, which becomes computationally inefficient when multiple lines fail. The problem was formulated as a sparse recovery problem over an overcomplete representation in [24, 25], where the combinatorial sparse recovery problem was relaxed to a linear programming (LP) problem. Machine learning-based recovery strategy was studied in [27]. All works discussed above assume DC power flow model. Recently, the detection of failed link caused by attack was studied under the AC power flow model [37–39]. Although the above works can successfully identify failures in some cases, *they all assume a connected post-attack grid and their recovery results cannot be verified in real time*. The works closest to ours are [19, 39], which established graph-theoretic conditions to guarantee the recovery accuracy. Our work differs from [19, 39] in the following aspects: (1) they still assume the grid to remain connected after attacks, while our solution is applicable to a partitioned post-attack grid where the power injections within the attacked area are unknown; (2) their conditions only characterize when all the line states can be identified correctly, while we provide recovery conditions at a finer granularity of individual lines. Such a finer granularity allows us to verify the states of a subset of lines (in Step 2 in Fig. 2.1) when the conditions in [19, 39] are not satisfied.

Besides blocking information as considered in this work, other types of cyber attacks are also possible, e.g., injecting false data into measurements. We refer readers to [17, 33, 40] and the references therein for details and leave failure localization under such attacks to future work.

The recovery from the joint cyber-physical attacks considered in this work is more challenging than traditional failure detection [41], since the jointly launched cyber attack will block the information from the attacked area and thus obfuscate the locations of the physical attack. Moreover, line failures due to naturally occurring faults typically do not occur at the same time and are mostly self-clearing, i.e., they rarely lead to line disconnection. On the contrary, a coordinated physical attack could take place simultaneously at multiple places, which may even lead to islanding.

2.1.2 Summary of Contributions

We aim at estimating the power grid state within an attacked area from which measurements have been blocked, with the following contributions:

1. Motivated by an observation that the existing method and condition for recovering the phase angles within the attacked area, previously developed for the case of connected post-attack grid, remain valid in the case of islanding, we focus on the recovery of the line states (i.e., breaker status of lines) within the attacked area using the phase angles, for which under the DC power flow model we develop an LP-based algorithm that allows for unknown changes in the power injections within the attacked area (due to islanding).

2. We establish conditions under which the accuracy of the proposed algorithm is guaranteed, which are further developed into verifiable conditions that can be tested using observable information.

3. Based on the above conditions, we develop a polynomial-time algorithm to verify the correctness of the estimated line states. We further provide an algorithm for verifying the states of potentially more lines based on the line states verified by the previous algorithm.

4. We extend the DC-based failed line detection and line state verification algorithms to their AC-based variants.

5. Our evaluations on real grid topologies show that the proposed recovery algorithm is highly accurate in localizing the failed lines with very few false alarms, and most of the failed lines can be successfully verified by the proposed verification algorithms.

Roadmap. Section 2.2 presents our models and problem formulation. Under the DC power flow model, Section 2.3 presents the proposed algorithm for localizing failed lines and its performance analysis. Section 2.4 presents conditions and algorithms for verifying the correctness of the estimated line states. In Section 2.5, we extend the DC-based line state detection and verification algorithms to the AC model. Section 2.6 evaluates our solutions on real grid topologies, and Section 2.7 concludes the paper. **All appendices can be found in the supplementary file** (proofs in Appendix 2.8.1).

2.2 Problem Formulation

Notation. The main notations are summarized in Table 2.1. Moreover, given a subgraph X of G, V_X and E_X denote the subsets of nodes/lines in X, and $\boldsymbol{x}_X$ denotes the subvector

of a vector $\boldsymbol{x}$ containing elements corresponding to X. Similarly, given two subgraphs X and Y of G, $\boldsymbol{A}_{X|Y}$ denotes the submatrix of a matrix $\boldsymbol{A}$ containing rows corresponding to X and columns corresponding to Y. For a set A, $\mathbb{I}_C = 1$ if condition C holds and $\mathbb{I}_C = 0$ otherwise. We use $\boldsymbol{\Lambda}_{(\cdot)} \in \{0,1\}^{m\times n}$ with one nonzero element in each row to select entries from a vector such that $\boldsymbol{\Lambda}_{(\cdot)}\boldsymbol{x}$ is a subvector of $\boldsymbol{x}$. For a vector $\boldsymbol{x}$, $[\boldsymbol{x}]$ denotes a diagonal matrix with $\boldsymbol{x}$ on the main diagonal. For a complex-valued number x, we use $\mathrm{Re}(x)$ and $\mathrm{Im}(x)$ to denote its real and imaginary part, respectively.

2.2.1 Power Grid Model

We model the power grid as a connected undirected graph $G = (V, E)$, where V is the set of nodes (buses) and E the set of lines (transmission lines). Each line $e = (s, t)$ is associated with a *reactance* r_{st} ($r_{st} = r_{ts}$) and a state $\in \{\text{operational}, \text{failed}\}$ (assumed to be operational before attack). Each node v is associated with a phase angle θ_v and an active power injection p_v. The phase angles $\boldsymbol{\theta} := (\theta_v)_{v\in V}$ and the active power injections $\boldsymbol{p} := (p_v)_{v\in V}$ are related by

$$\boldsymbol{B}\boldsymbol{\theta} = \boldsymbol{p}, \tag{2.1}$$

where $\boldsymbol{B} := (b_{uv})_{u,v\in V} \in \mathbb{R}^{|V|\times|V|}$ is the *admittance matrix*, defined as:

$$B_{uv} = \begin{cases} 0 & \text{if } u \neq v, (u,v) \notin E, \\ -1/r_{uv} & \text{if } u \neq v, (u,v) \in E, \\ -\sum_{w\in V\setminus\{u\}} b_{uw} & \text{if } u = v. \end{cases} \tag{2.2}$$

By arbitrarily assigning an orientation for each line, the topology of G can also be represented by the *incidence matrix* $\boldsymbol{D} \in \{-1, 0, 1\}^{|V|\times|E|}$, whose (i, j)-th entry is defined as

$$D_{ij} = \begin{cases} 1 & \text{if line } e_j \text{ comes out of node } v_i, \\ -1 & \text{if line } e_j \text{ goes into node } v_i, \\ 0 & \text{otherwise.} \end{cases} \tag{2.3}$$

It is worth noting that the proposed algorithms and analysis are not restricted to any specific orientation assignment.

As illustrated in Fig. 2.2, we assume that the grid is organized as a composition of multiple areas. Each area has a hybrid deployment of remote terminal units (RTUs) and

phasor measurement units (PMUs), which are responsible for collecting measurements. The measurements will be communicated to Supervisory Control and Data Acquisition (SCADA) or Wide Area Monitoring Protection and Control (WAMPAC) system for power grid management. As envisioned by [42], we consider a heterogeneous smart grid with several operators where multiple communication networks and protocols coexist. More specifically, the measurements and control instructions can be communicated through traditional fiber optic cables, power lines [43] or wireless links [44]. Due to the wide range of communication media, heterogeneous protocols (such as DNP3 [45], IEEE 1901 FFT-OFDM [43], etc.) can coexist.

2.2.2 Attack Model

As illustrated in Fig. 2.2, we consider an adversary who launches joint cyber-physical attacks during the interval between two consecutive state estimations for a specific area $H = (V_H \subseteq V, E_H \subseteq E)$. We assume that the attacks have successfully bypassed both prevention and detection measures.

The physical part will disconnect a set F ($|F| > 0$) of lines within H by either manipulating the breaker status or cutting the power lines.

The cyber part will take the form of Denial-of-Service (DoS) attacks to block the measurements within H. In other words, although the control center can observe the information in $\bar{H} = (V_{\bar{H}} \subseteq V, E_{\bar{H}} \subseteq E)$, information within the attacked area H, especially the post-attack topology and power injections, is blocked. Such DoS attacks can be achieved by destroying communication media (wireless or wired link), congesting the communication network (e.g., through telephonic floods), or remotely wiping the data in servers [23].

One of the motivations behind such joint attacks is to cause service disruption for consumers, especially critical infrastructures. Specifically, a large-scale line disconnection due to physical attacks can cause islanding and the associated load shedding, which can cause disruption of service to affected customers, and in the worst case, the collapse of the whole island in absence of generation. It is worth noting that the role of cyber attack considered in this work is not to hide the existence of physical attacks as considered in false data injection [17, 33], but to hinder the recovery process [23]. More specifically, repairing/restoration usually requires the knowledge of topology [46], which is normally available at the control center through binary breaker status measurements. However, the DoS attack will block such information and thus make the repairing/restoration scheduling challenging.

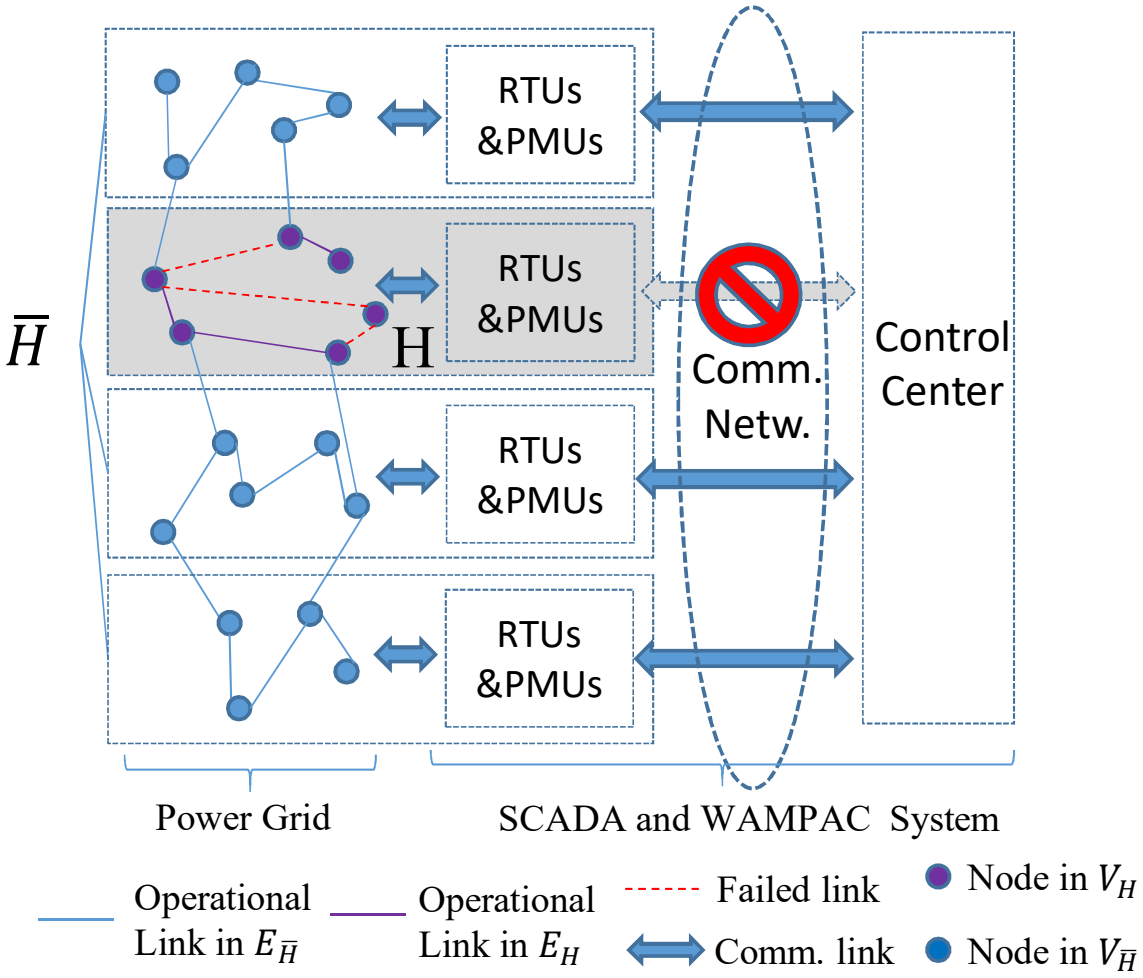


Figure 2.2. A cyber-physical attack that blocks information from the attacked area H while disconnecting certain lines within H.

Formally, we denote x' as the post-attack counterpart of the pre-attack value x. Before attack, the control center has full access to measurements on power injections, line flows, and the information of breaker status (the topology $G = (E, V)$). The physical attack will change G to $G' = (V, E')$, where $E'_{\bar{H}} = E_{\bar{H}}$ while $E'_H \neq E_H$, as illustrated in Fig. 2.2. The cyber attack will block the information within $H = (V_H, E_H)$. To schedule the repairing/restoration to recover from the attack, we need to recover the topology information E'_H or equivalently detecting the failed lines F.

This work focuses on "Topology Recovery" (detecting $\hat{F}$) and "Topology verification" during the recovery from a general class of attacks that result in power line disconnection and information loss.

Table 2.1. Notations

Notation	Description
$G = (V, E)$	power grid
H, $\bar{H}$	attacked/unattacked area
F , E_o	set of failed/operational lines after attack
$\boldsymbol{B}$, $\boldsymbol{D}$	admittance/incidence matrix
$\boldsymbol{\theta}$, $\boldsymbol{\theta}'$	phase angles before/after attack
$\boldsymbol{p}$, $\boldsymbol{p}'$	active power injections before/after attack
$\boldsymbol{\Gamma}$	$[\frac{1}{r_e}]_{e\in E}$ (r_e: reactance of line e)
$\boldsymbol{\Delta}$	change in active power injections
$\tilde{\boldsymbol{D}}$	hypothetical post-attack power flows (2.5)
η	rounding threshold in FLD
$S_U, f_{U,g}, f_{U,1(0)}$	definitions related to hyper-node (2.16)

2.2.3 Voltage Recovery Problem

Our goal is to recover the post-attack state within H, based on the grid state before the attack (e.g., $\boldsymbol{B}$ and $\boldsymbol{\theta}$) and the information from the unattacked area $\bar{H}$ after the attack (e.g., $\boldsymbol{\theta}'_{\bar{H}}$). In contrast to the previous works, we consider cases where the attack may partition the grid into multiple islands, which can cause changes in active power injections to maintain the supply-demand balance in each island. Let $\boldsymbol{\Delta} = (\Delta_v)_{v\in V} := \boldsymbol{p} - \boldsymbol{p}'$ denote the change in active power injections due to such supply-demand balance, where $\Delta_v > 0$ if v is a generator bus and $\Delta_v \leq 0$ otherwise.

We observe in [47, Theorem A.1] that the existing condition in [19, Theorem 1] for recovering the phase angles $\boldsymbol{\theta}'_H$ in a connected post-attack grid remains valid under an attack that possibly partitions the grid into islands, which allows $\boldsymbol{\theta}'_H$ to be recovered through solving a linear system. Thus, we assume $\boldsymbol{\theta}'_H$ to be known in the sequel to focus on the recovery of the line states within H, i.e., the localization of F.

Remark: Alternatively, $\boldsymbol{\theta}'_H$ can be obtained from secured PMUs as assumed in [48–50]. We refer to Appendix 2.8.3 for details, which which also provides guidelines on placing secured PMUs for recovering $\boldsymbol{\theta}'_H$.

2.3 Localizing Failed lines with Unknown Active Power Injections

Although providing theoretical guarantees, the existing solutions for localizing failed lines in [19, 39] assume either a connected post-attack grid or a known $\boldsymbol{\Delta}_H$, which cannot be guaranteed in practice. To address this limitation, we will first present an algorithm extended from [19] that can jointly estimate the failed lines F and the changes in power injections $\boldsymbol{\Delta}_H$ (Section 2.3.1), and then analyze the algorithm's accuracy in estimating F under unknown $\boldsymbol{\Delta}_H$ (Section 2.3.2).

2.3.1 Algorithm

Our algorithm extends the failure localization algorithm in [19] (which assumes $\boldsymbol{\Delta}_H = \mathbf{0}$) by formulating the $(F, \boldsymbol{\Delta}_H)$ joint estimation problem as the following optimization.

Constraints: Let $\boldsymbol{x} \in \{0,1\}^{|E|}$ be an indicator vector such that $x_e = 1$ if and only if $e \in F$. Due to $\boldsymbol{B} = \boldsymbol{D}\boldsymbol{\Gamma}\boldsymbol{D}^T$ (see Table 2.1 for the definitions), we can write the post-attack admittance matrix as $\boldsymbol{B}' = \boldsymbol{B} - \boldsymbol{D}\boldsymbol{\Gamma}[\boldsymbol{x}]\boldsymbol{D}^T$, which together with $\boldsymbol{\Delta} = \boldsymbol{p} - \boldsymbol{p}'$ implies

$$\boldsymbol{\Delta}_H = \boldsymbol{B}_{H|G}(\boldsymbol{\theta} - \boldsymbol{\theta}') + \boldsymbol{D}_H\boldsymbol{\Gamma}_H[\boldsymbol{D}_{G|H}^T\boldsymbol{\theta}']\boldsymbol{x}_H, \tag{2.4}$$

where $\boldsymbol{D}_{G|H} \in \{-1,0,1\}^{|V|\times|E_H|}$ is the submatrix of the incidence matrix $\boldsymbol{D}$ only containing the columns corresponding to lines in H[1]. For simplicity, we define

$$\tilde{\boldsymbol{D}} := \boldsymbol{D}\boldsymbol{\Gamma}[\boldsymbol{D}^T\boldsymbol{\theta}']. \tag{2.5}$$

For line $e_k = (i,j)$, $\tilde{D}_{i,k} = -\tilde{D}_{j,k} = \frac{\theta'_i - \theta'_j}{r_{ij}}$, which indicates the post-attack power flow on line e_k if it is operational.

In addition, $\boldsymbol{\Delta}_H$ is subject to the following constraints:

$$p_v \geq \Delta_v \geq 0, \ \ \forall v \in \{u \,|u \in V_H, p_u > 0\}, \tag{2.6a}$$

$$p_v \leq \Delta_v \leq 0, \ \ \forall v \in \{u \,|u \in V_H, p_u \leq 0\}, \tag{2.6b}$$

$$\mathbf{1}^T\boldsymbol{\Delta} = 0, \tag{2.6c}$$

[1]Because we focus on the post-attack recovery stage as shown in Fig. 2.1, we implicitly assume in (2.4) that the grid has reached the post-attack steady state. This means that each island is assumed to have reached a steady state in the case of islanding.

which ensure that (i) the bus type will remain the same after attack, (ii) the load will not increase after islanding, and (iii) the total power is balanced. It is worth noting that (2.6c) is ensured by (2.4), which implies that $\mathbf{1}^T\boldsymbol{\Delta}_H - \mathbf{1}^T\boldsymbol{B}_{H|G}(\boldsymbol{\theta}-\boldsymbol{\theta}') = (\mathbf{1}^T\tilde{\boldsymbol{D}}_H)\boldsymbol{x}_H = 0$ since $\mathbf{1}^T\tilde{\boldsymbol{D}}_H = \mathbf{0}$ by definition (2.5). This implies that any $\boldsymbol{\Delta}_H$ satisfying (2.4) will satisfy $\mathbf{1}^T\boldsymbol{\Delta}_H = \mathbf{1}^T\boldsymbol{B}_{H|G}(\boldsymbol{\theta}-\boldsymbol{\theta}') = \mathbf{1}^T\boldsymbol{\Delta}_H^*$ ($\boldsymbol{\Delta}_H^*$: the ground-truth power injection changes in H), and thus satisfy (2.6c). Hence, we will omit (2.6c) in the sequel.

Objective: The problem of failure localization aims at finding a failed line set $\hat{F}$ that is as close as possible to the ground-truth set F, while satisfying all the constraints. The solution is generally not unique, e.g., if both endpoints of a line $l \in E_H$ are disconnected from $\bar{H}$ after the attack, then the states of l will have no impact on any observable variable, and hence cannot be determined. To resolve this ambiguity, we set our objective as using the fewest failed lines to satisfy all the constraints. This idea has been applied to failure localization in power grid in various forms [19, 24, 25]. Mathematically, the problem is formulated as

$$\text{(P0)} \quad \min_{\boldsymbol{x}_H, \boldsymbol{\Delta}_H} \ \mathbf{1}^T\boldsymbol{x}_H \tag{2.7a}$$

$$\text{s.t.} \quad (2.4), (2.6a) - (2.6b), \tag{2.7b}$$

$$x_e \in \{0,1\}, \quad \forall e \in E_H, \tag{2.7c}$$

where the decision variables are $\boldsymbol{x}_H$ and $\boldsymbol{\Delta}_H$. Although binary linear programming is generally hard, (P0) is only a special case and hence needs to be analyzed separately.

Lemma 2.3.1. *The optimization (P0) is NP-hard.*

By relaxing the integer constraint (2.7c), (P0) is relaxed into

$$\text{(P1)} \quad \min_{\boldsymbol{x}_H, \boldsymbol{\Delta}_H} \ \mathbf{1}^T\boldsymbol{x}_H \tag{2.8a}$$

$$\text{s.t.} \quad (2.4), (2.6a) - (2.6b), \tag{2.8b}$$

$$\mathbf{0} \le \boldsymbol{x}_H \le \mathbf{1}. \tag{2.8c}$$

where $\mathbf{0} \le \boldsymbol{x}_H \le \mathbf{1}$ denotes element-wise inequality. To take into account the error on recovered $\boldsymbol{\theta}'$, we can add a noise term in (2.6a)-(2.6b). For example, by denoting ϵ as a tunable noise term, (2.6a) becomes $p_v + \epsilon \ge \Delta_v \ge -\epsilon$. We assume $\epsilon = 0$ in the rest of the paper to focus on the recovery from the information loss caused by the attack. The problem (P1) is a linear program (LP) which can be solved in polynomial time. Based on (P1), we propose an algorithm for localizing the failed lines, called ***F**ailed **L**ine **D**etection*

(FLD) as given in Alg. 1, where the input parameter $\eta \in (0, 1)$ is a threshold for rounding the factional solution of $\boldsymbol{x}_H$ to an integral solution ($\eta = 0.5$ in our experiments). We will illustrate how to set η at the end of Section 2.4.

Algorithm 1: **F**ailed **L**ine **D**etection (FLD)

Input: $\boldsymbol{B}, \boldsymbol{p}, \boldsymbol{\Delta}_{\bar{H}}, \boldsymbol{\theta}, \boldsymbol{\theta}', \boldsymbol{D}, \eta$
Output: $\hat{F}$
1 Solve the problem (P1) to obtain $\boldsymbol{x}_H$;
2 Return $\hat{F} = \{e : x_e \geq \eta\}$.

2.3.2 Analysis

We now analyze when FLD can correctly localize the failed lines, the results of which will lay the groundwork for the verifiable conditions in the next section. In the sequel, $\boldsymbol{\Delta}_H^*$ denotes the ground-truth power injection changes in H and $\boldsymbol{x}_H^*$ denotes the ground-truth failure indicators.

According to (2.6), we decompose V_H into $V_{H,L}$ for nodes with $p_v \leq 0$ and $V_{H,S}$ for the rest. Define $E_o \subseteq E_H$ as the set of lines that operate normally after failure, and $F \subseteq E_H$ as the failed lines. We make the following assumption:

Assumption 1. *As in [19], we assume that for each line $(s, t) \in E_H$, $\theta'_s \neq \theta'_t$, as otherwise the line will carry no power flow and hence its states cannot be identified*[2]*.*

First, we simplify (P1) into an equivalent but simpler optimization problem. To this end, we combine the decision variables $\boldsymbol{\Delta}_H$ and $\boldsymbol{x}_H$ of (P1) into a single vector $\boldsymbol{y}_H = [\boldsymbol{\Delta}_H^T, \boldsymbol{x}_H^T]^T \in \mathbb{R}^{(|E_H|+|V_H|)}$ (where $[A, B]$ denotes horizontal concatenation), and explicitly represent the solution to $\boldsymbol{y}_H$ that satisfies (2.4). Notice that (2.4) can be written as $[\boldsymbol{I}_{|V_H|}, -\tilde{\boldsymbol{D}}_H]\boldsymbol{y}_H = \boldsymbol{B}_{H|G}(\boldsymbol{\theta} - \boldsymbol{\theta}')$ ($\boldsymbol{I}_{|V_H|}$: the $|V_H| \times |V_H|$ identity matrix). The ground-truth solution $\boldsymbol{y}_H^* = [(\boldsymbol{\Delta}_H^*)^T, (\boldsymbol{x}_H^*)^T]^T$ certainly satisfies (2.4). Next, consider the null space of $[\boldsymbol{I}_{|V_H|}, -\tilde{\boldsymbol{D}}_H]$, whose dimension is $|E_H|$. It is easy to verify that $[\tilde{\boldsymbol{d}}_e^T, \boldsymbol{u}_e^T]^T$ ($e \in E_H$) are $|E_H|$ independent vectors spanning the null space of $[\boldsymbol{I}_{|V_H|}, -\tilde{\boldsymbol{D}}_H]$, where $\tilde{\boldsymbol{d}}_e$ is the column vector of $\tilde{\boldsymbol{D}}_H$ corresponding to line e, and $\boldsymbol{u}_e$ is a unit vector in $\mathbb{R}^{|E_H|}$

[2]This assumption essentially means that we will ignore the existence of such lines in failure localization. Specifically, in the case of islanding, if an island has a total blackout (because of not containing load/generation or frequency collapse), then lines in this island will be excluded from failure localization.

with the e-th element being 1 and the other elements being 0. Therefore, any pair of $(\boldsymbol{\Delta}_H, \boldsymbol{x}_H)$ satisfying (2.4) can be expressed as

$$\boldsymbol{y}_H = \begin{bmatrix} \boldsymbol{\Delta}_H \\ \boldsymbol{x}_H \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}_H^* \\ \boldsymbol{x}_H^* \end{bmatrix} + \sum_{e \in E_H} c_e \begin{bmatrix} \tilde{\boldsymbol{d}}_e \\ \boldsymbol{u}_e \end{bmatrix}, \tag{2.9}$$

where c_e's are the coefficients. Based on the decomposition of V_H into $V_{H,L}$ and $V_{H,S}$, $\tilde{\boldsymbol{D}}_H$ and $\boldsymbol{\Delta}_H$ can be written as

$$\tilde{\boldsymbol{D}}_H = \begin{array}{c} \\ V_{H,L} \\ V_{H,S} \end{array} \begin{array}{c} \begin{array}{cc} E_o & F \end{array} \\ \begin{bmatrix} \tilde{\boldsymbol{D}}_{H,L,o} & \tilde{\boldsymbol{D}}_{H,L,F} \\ \tilde{\boldsymbol{D}}_{H,S,o} & \tilde{\boldsymbol{D}}_{H,S,F} \end{bmatrix} \end{array}, \tag{2.10a}$$

$$\boldsymbol{\Delta}_H = \begin{array}{c} V_{H,L} \\ V_{H,S} \end{array} \begin{bmatrix} \boldsymbol{\Delta}_{H,L} \\ \boldsymbol{\Delta}_{H,S} \end{bmatrix}. \tag{2.10b}$$

Let $\tilde{\boldsymbol{D}}_{H,L} := [\tilde{\boldsymbol{D}}_{H,L,o}, \tilde{\boldsymbol{D}}_{H,L,F}]$, $\tilde{\boldsymbol{D}}_{H,S} := [\tilde{\boldsymbol{D}}_{H,S,o}, \tilde{\boldsymbol{D}}_{H,S,F}]$, and $\boldsymbol{c} := (c_e)_{e \in E_H} \in \mathbb{R}^{|E_H|}$. Since $\boldsymbol{\Delta}_{H,L}$ and $\boldsymbol{\Delta}_{H,S}$ are constrained differently in (2.6a) and (2.6b), we introduce $\boldsymbol{\Lambda}_L = [\boldsymbol{I}_{|V_{H,L}|}, \boldsymbol{0}]$ and $\boldsymbol{\Lambda}_S = [\boldsymbol{0}, \boldsymbol{I}_{|V_{H,S}|}]$ such that $\boldsymbol{\Delta}_{H,L} = \boldsymbol{\Lambda}_L \boldsymbol{\Delta}_H$, $\boldsymbol{\Delta}_{H,S} = \boldsymbol{\Lambda}_S \boldsymbol{\Delta}_H$, $\tilde{\boldsymbol{D}}_{H,L} = \boldsymbol{\Lambda}_L \tilde{\boldsymbol{D}}_H$ and $\tilde{\boldsymbol{D}}_{H,S} = \boldsymbol{\Lambda}_S \tilde{\boldsymbol{D}}_H$. According to (2.9), for FLD to correctly localize the failed lines, it suffices to have $x_e^* + c_e \geq \eta$ for all $e \in F$ and $x_e^* + c_e < \eta$ for all $e \in E_o$. Equivalently, it suffices to ensure that the optimal solution $\hat{\boldsymbol{c}}$ to the following optimization problem satisfies $\hat{c}_e \geq \eta - 1$ for all $e \in F$ and $\hat{c}_e < \eta$ for all $e \in E_o$:

$$\begin{aligned}
\min_{\boldsymbol{c}} \quad & \boldsymbol{1}^T \boldsymbol{c} && \text{(2.11a)} \\
\text{s.t.} \quad & \tilde{\boldsymbol{D}}_{H,L} \boldsymbol{c} \leq -\boldsymbol{\Delta}_{H,L}^*, && \text{(2.11b)} \\
& -\tilde{\boldsymbol{D}}_{H,L} \boldsymbol{c} \leq -(\boldsymbol{\Lambda}_L \boldsymbol{p}_H - \boldsymbol{\Delta}_{H,L}^*), && \text{(2.11c)} \\
& -\tilde{\boldsymbol{D}}_{H,S} \boldsymbol{c} \leq \boldsymbol{\Delta}_{H,S}^*, && \text{(2.11d)} \\
& \tilde{\boldsymbol{D}}_{H,S} \boldsymbol{c} \leq \boldsymbol{\Lambda}_S \boldsymbol{p}_H - \boldsymbol{\Delta}_{H,S}^*, && \text{(2.11e)} \\
& -\boldsymbol{c} \leq \boldsymbol{x}_H^*, && \text{(2.11f)} \\
& \boldsymbol{c} \leq \boldsymbol{1} - \boldsymbol{x}_H^*, && \text{(2.11g)}
\end{aligned}$$

where (2.11a) is equivalent to (2.8a), (2.11b)-(2.11c) correspond to (2.6b), (2.11d)-(2.11e) correspond to (2.6a), (2.11f)-(2.11g) correspond to (2.8c), and the change of variables $\boldsymbol{x}_H, \boldsymbol{\Delta}_H$ into $\boldsymbol{c}$ based on (2.9) ensures the satisfaction of (2.4). This equivalent formulation

of (P1) will help to simplify our analysis by eliminating the equality constraint (2.4). *For notational simplicity, we will omit the subscript $_H$ in the sequel* unless it causes confusion.

Next, we use (2.11) to analyze the accuracy of FLD. Let $\hat{F}$ be the failed line set returned by FLD. We first define $Q_m = F \setminus \hat{F}$ as the set of missed failed lines, and $Q_f = \hat{F} \setminus F$ as the set of operational lines that are falsely detected as failed. Note that according to (2.11), a failed line $e \in F$ is missed if and only if $\hat{c}_e < \eta - 1$. Similarly, an operational line $e \in E_o$ is falsely detected as failed if and only if $\hat{c}_e \geq \eta$. To express this in a vector form, we define $\boldsymbol{W}_m \in \{0,1\}^{|Q_m| \times |E_H|}$ as a binary matrix, where for each $i = 1, \ldots, |Q_m|$, $(W_m)_{i,e} = 1$ if the i-th missed line is line e and thus we have $\boldsymbol{W}_m \hat{\boldsymbol{c}} \leq (\eta - 1)\mathbf{1}$. Similarly, $\boldsymbol{W}_f \in \{0,1\}^{|Q_f| \times |E_H|}$ is defined such that $(W_f)_{i,e} = 1$ if the i-th false-alarmed line is line e, which leads to $-\boldsymbol{W}_f \hat{\boldsymbol{c}} \leq -\eta \mathbf{1}$. For ease of presentation, we define

$$\boldsymbol{A}_D^T := [\tilde{\boldsymbol{D}}_L^T, -\tilde{\boldsymbol{D}}_L^T, -\tilde{\boldsymbol{D}}_S^T, \tilde{\boldsymbol{D}}_S^T] \in \mathbb{R}^{|E_H| \times 2|V_H|}, \tag{2.12a}$$

$$\boldsymbol{A}_x^T := [-\boldsymbol{I}_{|E_H|}, \boldsymbol{I}_{|E_H|}] \in \mathbb{R}^{|E_H| \times 2|E_H|}, \tag{2.12b}$$

$$\boldsymbol{W}^T := [\boldsymbol{W}_m^T, -\boldsymbol{W}_f^T] \in \mathbb{R}^{|E_H| \times (|Q_m| + |Q_f|)}, \tag{2.12c}$$

$$\boldsymbol{g}_D^T := [-(\boldsymbol{\Delta}_L^*)^T, (-\boldsymbol{p}'_L)^T, (\boldsymbol{\Delta}_S^*)^T, (\boldsymbol{p}'_S)^T], \tag{2.12d}$$

$$\boldsymbol{g}_x^T := [(\boldsymbol{x}^*)^T, \mathbf{1}^T - (\boldsymbol{x}^*)^T] \in \mathbb{R}^{1 \times 2|E_H|}, \tag{2.12e}$$

$$\boldsymbol{g}_w^T := [(\eta - 1)\mathbf{1}^T, -\eta \mathbf{1}^T] \in \mathbb{R}^{1 \times (|Q_m| + |Q_f|)}, \tag{2.12f}$$

where $\boldsymbol{p}'_L = \boldsymbol{p}_L - \boldsymbol{\Delta}_L^*$ and $\boldsymbol{p}'_S = \boldsymbol{p}_S - \boldsymbol{\Delta}_S^*$ denote the post-attack active power injections at $V_{H,L}$ and $V_{H,S}$. Then the constraints in (2.11) can be written as $[\boldsymbol{A}_D^T, \boldsymbol{A}_x^T]^T \boldsymbol{c} \leq [\boldsymbol{g}_D^T, \boldsymbol{g}_x^T]^T$, and the optimal solution must satisfy $\boldsymbol{W}\boldsymbol{c} \leq \boldsymbol{g}_w$. The following observation is the foundation of our analysis.

Lemma 2.3.2. *A line $e \in F$ cannot be missed by FLD if for $Q_m = \{e\}$ and $Q_f = \emptyset$, there is a solution $\boldsymbol{z} \geq \mathbf{0}$ to*

$$[\boldsymbol{A}_D^T, \boldsymbol{A}_x^T, \boldsymbol{W}^T, \mathbf{1}]\boldsymbol{z} = \mathbf{0}, \tag{2.13a}$$

$$[\boldsymbol{g}_D^T, \boldsymbol{g}_x^T, \boldsymbol{g}_w^T, \mathbf{0}]\boldsymbol{z} < 0. \tag{2.13b}$$

Similarly, a line $e' \in E_o$ cannot be falsely detected as failed by FLD if there exists a solution $\boldsymbol{z} \geq \mathbf{0}$ to (2.13) *where $\boldsymbol{W}$ is constructed according to $Q_f = \{e'\}$ and $Q_m = \emptyset$.*

The proof is by contradiction: if $e \in F \setminus \hat{F}$, then for $\boldsymbol{W}$ corresponding to $Q_m = \{e\}$ and $Q_f = \emptyset$, there must be no $\boldsymbol{z} \geq \mathbf{0}$ satisfying (2.13); similar argument holds for $e' \in E_o$

by assuming $e' \in \hat{F} \setminus F$. See detailed proof in Appendix 2.8.1.

For ease of presentation, we will introduce a few notations as follows. Denote $\tilde{\boldsymbol{D}}_u$ as the row in $\tilde{\boldsymbol{D}}$ corresponding to node u, and $\tilde{D}_{u,e}$ as the entry in $\tilde{\boldsymbol{D}}_u$ corresponding to line e. Recall that as defined in (2.5), if $e = (u, v)$, then $\tilde{D}_{u,e} = (\theta'_u - \theta'_v) r_{uv}^{-1}$. We decompose the l.h.s of (2.13a) into $\boldsymbol{A}_D^T \boldsymbol{z}_D + \boldsymbol{A}_x^T [\boldsymbol{z}_{x-}, \boldsymbol{z}_{x+}] + \boldsymbol{W}_m^T \boldsymbol{z}_{w,m} + \boldsymbol{W}_f^T \boldsymbol{z}_{w,f} + z_* \mathbf{1}$ such that its row corresponding to line e can be written as

$$\sum_{u \in V_H} \left(\tilde{D}_{u,e} z_{D,u} - \tilde{D}_{u,e} z_{D,-u} \right) + \left(z_{x+,e} - z_{x-,e} \right) + \mathbb{I}_{Q_m}(e) z_{w,m,e} - \mathbb{I}_{Q_f}(e) z_{w,f,e} + z_*. \quad (2.14)$$

Similarly, the l.h.s of (2.13b) can be expanded into

$$\sum_{u \in V_H} \left(g_{D,u} z_{D,u} + g_{D,-u} z_{D,-u} \right) + \sum_{e \in E_H} [z_{x+,e}(1 - x_e^*) + z_{x-,e} x_e^*] + \boldsymbol{g}_w^T \boldsymbol{z}_w + z_*, \quad (2.15)$$

where $\boldsymbol{g}_w^T \boldsymbol{z}_w = \sum_{e \in E_H} [\mathbb{I}_{Q_m}(e) z_{w,m,e}(\eta - 1) - \mathbb{I}_{Q_f}(e) z_{w,f,e} \eta]$, $g_{D,u} := -\Delta_u^*$ and $g_{D,-u} := -p'_u$ if $p_u \le 0$, whereas $g_{D,u} := p'_u$ and $g_{D,-u} := \Delta_u^*$ if $p_u > 0$. Then, a solution $\boldsymbol{z} \ge \mathbf{0}$ satisfies (2.13) if $\forall e \in E_H$, we have (2.14) equal to 0 and (2.15) less than 0.

Equipped with Lemma 2.3.2, we are ready to explicitly characterize what types of lines are guaranteed to be correctly identified. Specifically, we will show that a line will satisfy the conditions in Lemma 2.3.2 if its endpoints satisfy certain conditions. To make our conditions as general as possible, we introduce a generalization of node called *hyper-node* as follows (a single node is also a hyper-node):

Definition 2.3.1. *A set of nodes $U \subseteq V_H$ is a hyper-node if they induce a connected subgraph before attack.*

We define a few properties of a hyper-node U. Define E_U as the set of lines with exactly one endpoint in U, i.e, $E_U := \{e | e = (s, t) \in E_H, s \in U, t \notin U\}$. If $E_U \cap F \neq \emptyset$, we define

$$\tilde{D}_{U,e} := \sum_{u \in U} \tilde{D}_{u,e}, \quad (2.16a)$$

$$S_U := \{e \in E_U \setminus F | \exists l \in E_U \cap F, \tilde{D}_{U,l} \tilde{D}_{U,e} > 0\}, \quad (2.16b)$$

$$f_{U,0} := \max_{e \in S_U} |\tilde{D}_{U,e}|, \text{ where } f_{U,0} := 0 \text{ if } S_U = \emptyset, \quad (2.16c)$$

$$f_{U,1} := \min_{e \in E_U \cap F} |\tilde{D}_{U,e}|, \quad (2.16d)$$

$$f_{U,g} := \begin{cases} \sum_{u \in U} g_{D,u} & \text{if } \exists l \in E_U \cap F, \tilde{D}_{U,l} < 0, \\ \sum_{u \in U} g_{D,-u} & \text{otherwise.} \end{cases} \quad (2.16e)$$

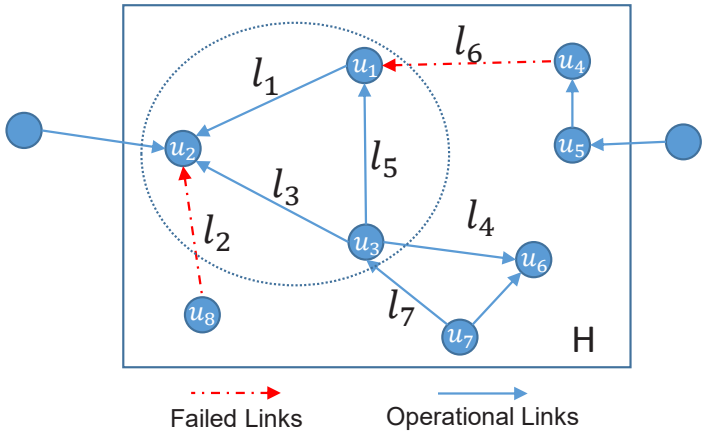


Figure 2.3. An example of hyper-node (arrow denotes the direction of a power flow or a hypothetical power flow).

Example 1. *Consider an attacked area H as shown in Fig. 2.3, where blue circles denote nodes (buses) while the direction of each line indicates the direction of power flow*[3]*. Suppose that $F = \{l_2, l_6\}$ and all nodes are load buses. Nodes u_1, u_2 and u_3 form a hyper-node U, where $E_U = \{l_2, l_4, l_6, l_7\}$, $S_U = \{l_7\}$, $f_{U,0} = |\tilde{D}_{U,l_7}|$, $f_{U,1} = \min\{|\tilde{D}_{U,l_2}|, |\tilde{D}_{U,l_6}|\}$ and $f_{U,g} = -\sum_{v\in U} \Delta_v^*$. $\tilde{D}_{U,l_1} = \tilde{D}_{u_1,l_1} + \tilde{D}_{u_2,l_1} = 0$ since $l_1 \notin E_U$, while $\tilde{D}_{U,l_2} = \tilde{D}_{u_2,l_2} \neq 0$ since $l_2 \in E_U$.*

Based on these definitions and Lemma 2.3.2, we are ready to present a condition under which a failed line $l \in F$ will not be missed by FLD (see proof in Appendix 2.8.1).

Theorem 2.3.1. *A failed line $l \in F$ will be detected by FLD, i.e., $l \in \hat{F}$, if there exists at least one hyper-node (say U) such that $l \in E_U$, for which the following conditions hold:*

1. $\forall e, l \in E_U \cap F$, $\tilde{D}_{U,e}\tilde{D}_{U,l} > 0$,
2. $S_U = \emptyset$, *and*
3. $f_{U,g} + (\eta - 1)|\tilde{D}_{U,l}| < 0$.

Based on similar arguments, the following condition can guarantee that an operational line $l \in E_o$ will not be falsely detected by FLD ($l \notin \hat{F}$) (see proof in Appendix 2.8.1). For notational simplicity, we extend the definition of $f_{U,g}$ to a hyper-node U with $E_U \cap F = \emptyset$:

$$f_{U,g} := \begin{cases} \sum_{u\in U} g_{D,u} & \text{if } \exists l \in E_U \setminus F, \tilde{D}_{U,l} > 0, \\ \sum_{u\in U} g_{D,-u} & \text{otherwise.} \end{cases} \tag{2.17}$$

[3]These may be hypothetical power flows, as a failed line carries no flow.

Theorem 2.3.2. *An operational line $l \in E_o$ will not be detected (as failed) by FLD, i.e., $l \notin \hat{F}$, if there exists at least one hyper-node (say U) such that $l \in E_U$, for which the following conditions hold:*

1. $\forall l, l' \in E_U \cap E_o : \tilde{D}_{U,l}\tilde{D}_{U,l'} > 0$,
2. $S_U = \emptyset$ *if* $E_U \cap F \neq \emptyset$, *and*
3. $f_{U,g} - \eta|\tilde{D}_{U,l}| < 0$.

Remark: Theorems 2.3.1 and 2.3.2 provide sufficient conditions for FLD to correctly identify the states of a line l based on the direction and magnitude of "power flows" around a hyper-node U at the "endpoint" of l (i.e., $l \in E_U$): The (hypothetical) power flows on all the lines of the same state (failed or operational) around U should be in the same direction, i.e., all going into or out of U (condition 1); all lines of different states around U should have opposite (hypothetical) power flow directions (condition 2); the magnitude of the (hypothetical) power flow on the line of interest (i.e., l) should be sufficiently large (condition 3).

2.4 Verifying Estimated Line States

Although the conditions in Section 2.3.2 can guarantee the accuracy of FLD, they are not testable in practice since the ground-truth (F and $\mathbf{\Delta}^*$) is required. In this section, we will develop conditions requiring only observable information such that they can be verified during operation.

To this end, we first notice that Δ_u for some $u \in V_H$ can be recovered directly. For example, if the adjustment of power injections after islanding follows *proportional load shedding/generation reduction* [51, 52][4], then we have the following observations (see proof in Appendix 2.8.1):

Lemma 2.4.1. *Let $N(v; \bar{H})$ denote the set of all the nodes in $\bar{H}$ that are connected to node v via lines in $E \backslash E_H$. Then under the assumption of proportional load shedding/generation reduction, Δ_v for $v \in V_H$ can be recovered unless $N(v; \bar{H}) = \emptyset$ or every $u \in N(v; \bar{H})$ is of a different type from v with $\Delta_u = 0$.*

[4]Under this assumption, either the load or the generation (but not both) will be reduced upon the formation of an island. Moreover, if nodes u and v are in the same island and of the same type (both load or generator), then $p'_u/p_u = p'_v/p_v$. More details are given in Appendix 2.8.2.

Define $U_B \subseteq V_H$ as the node set such that $\forall u \in U_B$, Δ_u has been recovered (e.g., through Lemma 2.4.1). Our key observation is that for any hyper-node U, $\tilde{D}_{U,l}$ for any $l \in E_U$ can be computed with the knowledge of $\boldsymbol{\theta}'$, and $f_{U,g}$ can be upper-bounded by

$$\hat{f}_{U,g} := \sum_{u \in U \cap U_B} f_{u,g} + \sum_{u \in U \setminus U_B} |p_u|, \tag{2.18}$$

where $f_{u,g}$ is defined in (2.16e) for $U = \{u\}$. Since $f_{u,g}$ is known for nodes in U_B and p_u (power injection at u before attack) is also known, $\hat{f}_{U,g}$ is computable. We now show how to use this information to verify the results of FLD based on Lemma 2.3.2 and Theorems 2.3.1–2.3.2. We note that our solution remains valid even if $U_B = \emptyset$, which only affects the tightness of the upper bound $\hat{f}_{U,g}$.

2.4.1 Verification without Knowledge of Ground Truth

We first tackle the lines whose states can be verified without any knowledge of the ground truth line states.

2.4.1.1 Verifiable Conditions

The basic idea is to rule out the other possibility by constructing *counterexamples* to the theorems in Section 2.3.2 if the estimated line state is incorrect.

lines in 1*-edge cuts:* If line $e = (u_1, u_2)$ forms a *cut* of H, i.e., $(V_H, E_H \setminus \{e\})$ contains more connected components than H, then by breadth-first search (BFS) starting from u_1 and u_2 respectively without traversing e, we can construct two hyper-nodes U_1 and U_2 such that $E_{U_1} = E_{U_2} = \{e\}$ and thus $S_{U_1} = S_{U_1} = \emptyset$. For example, in Fig. 2.3, line $e := l_6$ is a 1-edge cut, and thus $U_1 := \{u_4, u_5\}$ and $U_2 := V_H \setminus U_1$ satisfy this condition. Then the following verifiable conditions are directly implied by Theorems 2.3.1–2.3.2:

Corollary 2.4.0.1. *If* $e \in \hat{F}$ *and* $\min\{\hat{f}_{U_1,g}, \hat{f}_{U_2,g}\} - \eta|\tilde{D}_{U_1,e}| < 0$*, then we can verify* $e \in F$*. If* $e \in E_H \setminus \hat{F}$ *and* $\min\{\hat{f}_{U_1,g}, \hat{f}_{U_2,g}\} + (\eta - 1)|\tilde{D}_{U_1,e}| < 0$*, then we can verify* $e \in E_H \setminus F$*.*

Proof. If $e \in \hat{F}$ and $\min\{\hat{f}_{U_1,g}, \hat{f}_{U_2,g}\} - \eta|\tilde{D}_{U_1,e}| < 0$, then e must have failed, since otherwise e would have been estimated as operational according to Theorem 2.3.2. Similarly, if $e \in E_H \setminus \hat{F}$ and $\min\{\hat{f}_{U_1,g}, \hat{f}_{U_2,g}\} + (\eta - 1)|\tilde{D}_{U_1,e}| < 0$, then e must be operational, since otherwise e would have been estimated as failed according to Theorem 2.3.1. Note that as our verification is based on contradiction, $\hat{f}_{U_i,g}$ should be computed as if $e \in E_H \setminus F$ to verify $e \in \hat{F}$ and vice-versa. □

lines in 2-edge cuts: If lines $e_1, e_2 \in E_H$ together form a cut of H but each individual line does not, then by BFS starting from the endpoints of e_1 (or e_2) without traversing e_1 or e_2, we can construct two hyper-nodes U_1, U_2 such that $E_{U_1} = E_{U_2} = \{e_1, e_2\}$. For example, as $e_1 := l_4$ and $e_2 := l_7$ form a 2-edge cut of H in Fig. 2.3, $U_1 := \{u_6, u_7\}$ and $U_2 := V_H \setminus U_1$ satisfy this condition. Moreover, any pair of lines in a cycle C form a 2-edge cut if they are not in any other cycle in H, e.g., any pair of lines in the cycle $\{l_1, l_3, l_5\}$ satisfy this condition. Based on this observation, we provide the following conditions for verifying the states of such lines

Theorem 2.4.1. *Consider a hyper-node U with $E_U = \{e_1, e_2\}$ and $e_1, e_2 \in E_H \setminus \hat{F}$. If $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} < 0$, then e_1, e_2 are guaranteed to both belong to $E_H \setminus F$ if*

1. $\hat{f}_{U,g} + (\eta - 1)\min\{|\tilde{D}_{U,e_1}|, |\tilde{D}_{U,e_2}|\} < 0$, *and*
2. $\eta < 1 - \min\{\frac{\hat{f}_{U,g}+|\tilde{D}_{U,e_1}|}{|\tilde{D}_{U,e_2}|}, \frac{\hat{f}_{U,g}+|\tilde{D}_{U,e_2}|}{|\tilde{D}_{U,e_1}|}\}$.

If $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} > 0$, then we can verify:

1. $e_1 \in E_H \setminus F$ *if* $(1-\eta)|\tilde{D}_{U,e_1}| > \hat{f}_{U,g} + |\tilde{D}_{U,e_2}|$,
2. $e_2 \in E_H \setminus F$ *if* $(1-\eta)|\tilde{D}_{U,e_2}| > \hat{f}_{U,g} + |\tilde{D}_{U,e_1}|$.

Theorem 2.4.2. *Consider a hyper-node U with $E_U = \{e_1, e_2\}$ and $e_1 \in \hat{F}, e_2 \in E_H \setminus \hat{F}$. If $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} > 0$, then the states of e_1, e_2 are guaranteed to be correctly identified if*

1. $\hat{f}_{U,g} - \eta|\tilde{D}_{U,e_1}| < 0$, $\hat{f}_{U,g} + (\eta - 1)|\tilde{D}_{U,e_2}| < 0$, *and*
2. *either* $\eta > \frac{\hat{f}_{U,g}+|\tilde{D}_{U,e_2}|}{|\tilde{D}_{U,e_1}|}$ *or* $\eta < 1 - \frac{\hat{f}_{U,g}+|\tilde{D}_{U,e_1}|}{|\tilde{D}_{U,e_2}|}$.

If $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} < 0$, then we can verify:

1. $e_1 \in F$ *if* $\eta|\tilde{D}_{U,e_1}| > \hat{f}_{U,g} + |\tilde{D}_{U,e_2}|$,
2. $e_2 \in E_H \setminus F$ *if* $(1-\eta)|\tilde{D}_{U,e_2}| > \hat{f}_{U,g} + |\tilde{D}_{U,e_1}|$.

Theorem 2.4.3. *Consider a hyper-node U with $E_U = \{e_1, e_2\}$ and $e_1, e_2 \in \hat{F}$. Then, we can verify:*

1. $e_1 \in F$ *if* $\eta|\tilde{D}_{U,e_1}| > \hat{f}_{U,g} + |\tilde{D}_{U,e_2}|$,
2. $e_2 \in F$ *if* $\eta|\tilde{D}_{U,e_2}| > \hat{f}_{U,g} + |\tilde{D}_{U,e_1}|$.

While in theory such verifiable conditions can also be derived for lines in larger cuts, the number of cases will grow exponentially. We also find 1–2-edge cuts to cover the majority of lines in practice (see Fig. 2.11).

2.4.1.2 Verification Algorithm

Based on Theorems 2.4.1–2.4.3, we develop an algorithm *Verification Of sTatEs (VOTE)* (Alg. 2) for verifying the line states estimated by FLD, which can be applied to lines in 1–2-edge cuts. Here, E_a denotes the set of all the lines in 1-edge cuts of H, while $\mathcal{E}_c$ denotes the set of 2-edge cuts. In the algorithm, lines in E_a are tested before lines in $\mathcal{E}_c$ since it is easier to extend the knowledge of U_B based on the test results for E_a. As for the complexity, we first note that the time complexity of each iteration is $\mathcal{O}(|E_H| + |V_H|)$ due to BFS. Then, it takes $\mathcal{O}(|E_H|)$ iterations to verify E_a and $\mathcal{O}(|E_H|^2)$ iterations for $\mathcal{E}_c$, which results in a total complexity of $\mathcal{O}(|E_H|^2(|E_H| + |V_H|))$.

Algorithm 2: **V**erification **O**f s**T**at**E**s (VOTE)

Input: $\tilde{\boldsymbol{D}}, \boldsymbol{p}, \boldsymbol{\Delta}_{\bar{H}}, U_B, \eta, E_a, \mathcal{E}_c, \hat{F}$
Output: E_v

$E_v \leftarrow \emptyset$; `/* verifiable lines */`
foreach $e = (u_1, u_2) \in E_a$ **do**
  Construct hyper-nodes U_1 and U_2 such that $E_{U_1} = E_{U_2} = \{e\}$;
  if $e \in \hat{F}$ **then**
    Add e to E_v if $\min\{\hat{f}_{U_1,g}, \hat{f}_{U_2,g}\} - \eta|\tilde{D}_{U_1,e}| < 0$;
  else
    Add e to E_v if $\min\{\hat{f}_{U_1,g}, \hat{f}_{U_2,g}\} + (\eta - 1)|\tilde{D}_{U_1,e}| < 0$;
  if *e is verified to be in $E_H \setminus F$* **then**
    Add u_i to U_B if Δ_{u_i} $(i = 1, 2)$ can be recovered through Lemma 2.4.1;
foreach $\{e_1, e_2\} \in \mathcal{E}_c$ **do**
  Construct hyper-nodes U_1 and U_2 such that $E_{U_1} = E_{U_2} = \{e_1, e_2\}$;
  Test the satisfaction of Lemma 2.4.1, 2.4.2, or 2.4.3 for U_1 and U_2, respectively;
  Add e_i $(i = 1, 2)$ to E_v if it is verified;

2.4.2 Verification with Partial Knowledge of Ground Truth

VOTE assumes no knowledge of the ground-truth line states, even if the states of some lines are already verified, e.g., line set E_v verified by VOTE. We observe that such partial knowledge of the ground truth can be exploited for better approximation of the unknown terms in (2.13) and thus verifying lines where VOTE fails. Following this idea, we propose a follow-up step designed to verify the states of additional lines in $E_H \setminus E_v$.

2.4.2.1 Verifiable Conditions

Recall that the idea for verifying the correctness of $e \in \hat{F}$ (or $e \in E_H \setminus \hat{F}$) is to construct a solution to (2.13) as if $e \in E_H \setminus F$ (or $e \in F$). Specifically, it can be shown that for a line $e \in \hat{F}$, if there exists $\boldsymbol{z} \geq \mathbf{0}$ for (2.13) where $\boldsymbol{W}$ is constructed for $Q_f = \{e\}$ and $Q_m = \emptyset$, then e is guaranteed to have failed since otherwise it must have been estimated to be operational. The challenge is the unknown $\boldsymbol{g}_D$, $\boldsymbol{g}_x$, and $\boldsymbol{g}_w$ due to unknown F and $\boldsymbol{\Delta}_H^*$. To tackle this challenge, we approximate these parameters by their worst possible values (in terms of satisfying (2.13)), which leads to the following result (see proof in Appendix 2.8.1).

Theorem 2.4.4. *Given a set E_v of lines with known states, we define $\hat{\boldsymbol{g}}_D \in \mathbb{R}^{2|V_H|}$ and $\hat{\boldsymbol{g}}_x \in \mathbb{R}^{2|E_H|}$ as follows:*

$$\hat{g}_{D,u} = \begin{cases} g_{D,u} \text{ if } u \in U_B \\ |p_u| \text{ if } u \notin U_B \end{cases} \quad \hat{g}_{x,e} = \begin{cases} g_{x,e} \text{ if } e \in E_v \\ 1 \text{ if } e \notin E_v \end{cases} \tag{2.19}$$

and define $\hat{g}_{D,-u}$ and $\hat{g}_{x,-e}$ similarly. Then, a line $l \in \hat{F}$ is verified to have failed if there exists a solution $\boldsymbol{z} \geq \mathbf{0}$ to

$$[\boldsymbol{A}_D^T, \boldsymbol{A}_x^T, \boldsymbol{w}^T, \mathbf{1}]\boldsymbol{z} = \mathbf{0}, \tag{2.20a}$$

$$[\hat{\boldsymbol{g}}_D^T, \hat{\boldsymbol{g}}_x^T, g_w, \mathbf{0}]\boldsymbol{z} < 0, \tag{2.20b}$$

where $\boldsymbol{w} \in \{0,1\}^{|E_H|}$ is defined to be $\boldsymbol{W}_f$ with $Q_f = \{l\}$, and $g_w := -\eta$. Similarly, a line $e \in E_H \setminus \hat{F}$ is verified to be operational if $\exists \boldsymbol{z} \geq \mathbf{0}$ that satisfies (2.20)*, where $\boldsymbol{w} \in \{0,1\}^{|E_H|}$ is defined to be $\boldsymbol{W}_m$ with $Q_m = \{e\}$, and $g_w := \eta - 1$.*

2.4.2.2 Verification Algorithm

All the elements in (2.20) are known, and thus the existence of a solution can be checked by solving an LP. Based on this result, we propose an algorithm *VOTE with Partial Ground truth (VOTE-PG)* (Alg. 3) for verifying the estimated states of the remaining lines, which iteratively updates E_v. Each iteration of VOTE-PG involves solving $O(|E_H|)$ LPs, each of which has a time complexity that is polynomial[5] in the number of decision variables ($|E_H|$) and the number of constraints ($|V_H| + |E_H|$) [53]. Since VOTE-PG has at most $|E_H|$ iterations, the total time complexity of VOTE-PG is polynomial in $|E_H|$

[5]The exact order of the polynomial depends on the specific algorithm used to solve the LP [53].

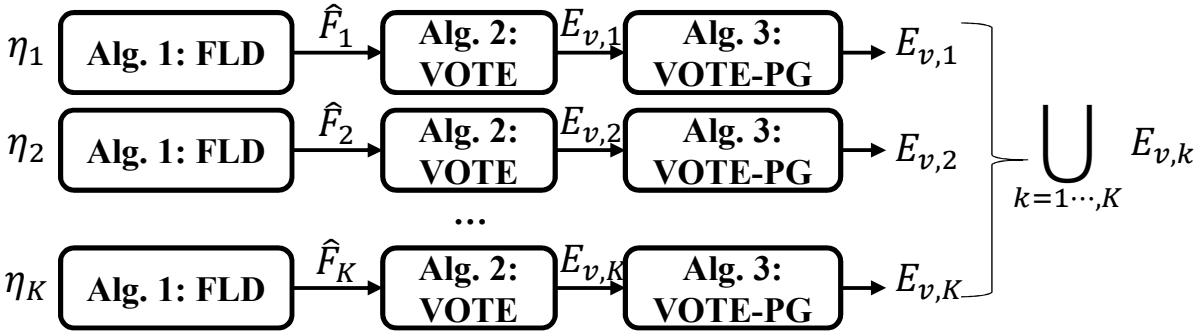


Figure 2.4. Guidelines for applying the proposed algorithms.

and $|V_H|$.

Algorithm 3: VOTE with **P**artial **G**round truth (VOTE-PG)

Input: $\tilde{\boldsymbol{D}}, \boldsymbol{p}, \boldsymbol{\Delta}_{\bar{H}}, U_B, \eta, E_H, E_v, \hat{F}, \hat{\boldsymbol{g}}_D, \hat{\boldsymbol{g}}_x$

1 **while** $E_H \setminus E_v \neq \emptyset$ **do**
2 $\bar{E}_v \leftarrow E_v$;
3 **foreach** $e \in E_H \setminus E_v$ **do**
4 **if** $\exists \boldsymbol{z} \geq \boldsymbol{0}$ *satisfying* (2.20) *for* e **then**
5 $\bar{E}_v \leftarrow \bar{E}_v \cup \{e\}$;
6 Update $\hat{\boldsymbol{g}}_x$;
7 **if** $|\bar{E}_v| > |E_v|$ **then**
8 $E_v \leftarrow \bar{E}_v$;
9 **else**
10 break;

Summary and Guidelines: In summary, Lemma 2.3.2 is the foundation of our results. Based on Lemma 2.3.2, we develop Theorems 2.3.1-2.3.2 to understand the relationship between the feasibility of (2.13) in Lemma 2.3.2 and the magnitudes/directions of power flows. Equipped with Lemma 2.3.2 and Theorems 2.3.1-2.3.2, we develop Theorems 2.4.1-2.4.3 based on verifiable conditions, which lead to the first verification algorithm (VOTE in Alg. 2). Then, we develop Theorem 2.4.4 to provide more verifiable conditions, which supports the second verification algorithm (VOTE-PG in Alg. 3).

The proposed algorithms form a three-step pipeline: FLD $\rightarrow$ VOTE $\rightarrow$ VOTE-PG, where FLD will estimate a set of failed lines ($\hat{F}$), based on which VOTE will identify a subset of lines (E_v) whose estimated states can be verified to be correct, and VOTE-PG will try to expand E_v.

All the proposed algorithms contain a parameter η. From Line 2 of FLD in Alg. 1, it is easy to see that a smaller η will make FLD less likely to miss failed lines. However, a

smaller η will also make a failed line harder to be verified as analyzed in Theorem 2.4.1-2.4.3. Similarly, an operational line is less likely to be detected as failed by FLD but harder to be verified with a larger η. Fortunately, there is no need to tune η. As shown in Fig. 2.4, the operator can run the proposed method with different values of η in parallel. For each value of η, the proposed method will return a set of lines with verified states (which are guaranteed to be consistent with the ground-truth states). Then, the operator can take the union of the sets obtained under different η values to recover more line states.

Remark: If control center knows that the post-attack grid remains connected, both Lemma 2.3.2 and VOTE can be enhanced. The details are given in Appendix 2.8.4.

2.5 Extension to AC Power Flow Model

So far we have assumed the DC power flow approximation as described in Section 2.2. As the real grid behaves according to the AC power flow model, the natural questions are: (i) if we can directly apply the DC-based solution (FLD) under the AC model, and (ii) if we can adapt these algorithms to work better under the AC model.

For the first question, it is easy to see that FLD can be directly applied under the AC model. As for the verification algorithms, we have the following result (see proof in Appendix 2.8.1).

Lemma 2.5.1. *The DC-based line state verification algorithms VOTE can correctly verify the line states under the AC model.*

Despite the applicability of DC-based algorithms, the approximation error in the DC power flow model degrades the performance of failure detection and verification (see Section 2.6.1). Fortunately, we will show that FLD can be easily adapted to suit the AC model. Since only minor modification is needed, we will use "AC-X" to denote the AC-based modification of result "X".

2.5.1 Detection: Adaptation of FLD to AC-FLD

We first show how to adapt the failure detection algorithm FLD. Some notations necessary for presenting the results under the AC model are shown in Table 2.2. Specifically, $D_{f,u,e} = 1$ and $D_{t,v,e} = 1$ if and only if $\exists e = (u, v) \in E$, i.e., $\boldsymbol{D} = \boldsymbol{D}_f - \boldsymbol{D}_t$. Based on a similar discussion as in Section 2.2.3, we assume that the voltage after attack ($\vec{\boldsymbol{v}}'$) has

been recovered through existing mechanisms [39, Lemma 1] or secured PMUs. We refer to Appendix 2.8.3 for details.

Table 2.2. Notations for AC power flow

Notation	Description
$\boldsymbol{p}/\boldsymbol{q} \in \mathbb{C}^{\lvert V\rvert}$	Active/reactive power injection
$\boldsymbol{\Delta}_p/\boldsymbol{\Delta}_q \in \mathbb{C}^{\lvert V\rvert}$	Active/reactive power injection change
$\vec{v}_u = v_u e^{j\cdot\theta_u}$ / l_u	Nodal voltage/current
$\boldsymbol{Y} = \boldsymbol{G} + j\boldsymbol{B}$	Bus admittance matrix
$\boldsymbol{D}_f/\boldsymbol{D}_t \in \{0,1\}^{\lvert V\rvert\times\lvert E\rvert}$	*From/to* end incidence matrix
$\boldsymbol{Y}_f/\boldsymbol{Y}_t \in \mathbb{C}^{\lvert E\rvert\times\lvert V\rvert}$	*From/to* end line admittance matrix

The key is to extend (P1) in (2.7) to the AC model. To this end, we first derive the counterpart of (2.4). Recall that $\boldsymbol{x} \in \{0,1\}^{|E|}$ indicates which lines have failed, i.e., $x_e = 1$ indicates $e \in F$. Recalling that $[\boldsymbol{x}]$ denotes the diagonal matrix with $\boldsymbol{x}$ on the main diagonal and noticing that the post-attack bus admittance matrix is $\boldsymbol{Y}' = \boldsymbol{Y} - \boldsymbol{D}_f[\boldsymbol{x}]\boldsymbol{Y}_f + \boldsymbol{D}_t[\boldsymbol{x}]\boldsymbol{Y}_t$, we can transform the AC power flow equation $\boldsymbol{l}'_H = \boldsymbol{Y}'_{H|G}\vec{\boldsymbol{v}}'$ into

$$\boldsymbol{l}'_H = \boldsymbol{Y}_{H|G}\vec{\boldsymbol{v}}' - \boldsymbol{D}_{f,H|G}[\boldsymbol{x}_H]\boldsymbol{Y}_{f,H|G}\vec{\boldsymbol{v}}' - \boldsymbol{D}_{t,H|G}[\boldsymbol{x}_H]\boldsymbol{Y}_{t,H|G}\vec{\boldsymbol{v}}'. \tag{2.21}$$

Then, by left multiplying the conjugate of both sides of (2.21) by $[\vec{\boldsymbol{v}}'_H]$, we have

$$\boldsymbol{\Delta}_{p,H} = \boldsymbol{p}_H - \mathrm{Re}\left([\vec{\boldsymbol{v}}'_H]\overline{\boldsymbol{Y}_{H|G}\vec{\boldsymbol{v}}'}\right) + \tilde{\boldsymbol{D}}_{p,H}\boldsymbol{x}_H, \tag{2.22a}$$

$$\boldsymbol{\Delta}_{q,H} = \boldsymbol{q}_H - \mathrm{Im}\left([\vec{\boldsymbol{v}}'_H]\overline{\boldsymbol{Y}_{H|G}\vec{\boldsymbol{v}}'}\right) + \tilde{\boldsymbol{D}}_{q,H}\boldsymbol{x}_H, \tag{2.22b}$$

where $\tilde{\boldsymbol{D}}_{p,H} = \mathrm{Re}\left(\tilde{\boldsymbol{D}}_H\right)$, $\tilde{\boldsymbol{D}}_{p,H} = \mathrm{Im}\left(\tilde{\boldsymbol{D}}_H\right)$, and

$$\tilde{\boldsymbol{D}}_H = [\vec{\boldsymbol{v}}'_H]\Big(\overline{\boldsymbol{D}_{f,H|G}[\boldsymbol{Y}_{f,H|G}\vec{\boldsymbol{v}}']} + \overline{\boldsymbol{D}_{t,H|G}[\boldsymbol{Y}_{t,H|G}\vec{\boldsymbol{v}}']}\Big). \tag{2.23}$$

Here we slightly abuse the notation for $\tilde{\boldsymbol{D}}_H$ since it indicates the hypothetical power flow after attack in (2.4) for DC model and (2.23) for AC model, respectively. Let $V_{H,L,I} = \{u \in V_{H,L} : q_{H,u} \le 0\}$. Then, we introduce the row selection matrix $\boldsymbol{\Lambda}_I \in \{0,1\}^{|V_{H,L,I}|\times|V_H|}$ to select entries in $\boldsymbol{q}_H$ corresponding to nodes in $V_{H,L,I}$. Similarly, we introduce $V_{H,L,C} = \{u \in V_{H,L} : q_{H,u} > 0\}$ and the associated $\boldsymbol{\Lambda}_C \in \{0,1\}^{|V_{H,L,C}|\times|V_H|}$. As the counterpart of (2.6b) for reactive power, we have

$$\boldsymbol{0} \ge \boldsymbol{\Lambda}_I\boldsymbol{\Delta}_{q,H} \ge \boldsymbol{\Lambda}_I\boldsymbol{q}_H, \boldsymbol{0} < \boldsymbol{\Lambda}_C\boldsymbol{\Delta}_{q,H} \le \boldsymbol{\Lambda}_C\boldsymbol{q}_H. \tag{2.24}$$

Now, we are ready to give the counterpart of (P1) in (2.8) under the AC power flow model, referred to as AC-(P1), as follows

$$\min_{\boldsymbol{x}_H} \quad \mathbf{1}^T \boldsymbol{x}_H \tag{2.25a}$$

$$\text{s.t.} \quad (2.22), (2.24), (2.6a) - (2.6b), \tag{2.25b}$$

$$0 \le x_{H,e} \le 1, \forall e \in E_H, \tag{2.25c}$$

Thus, FLD can be adapted to the AC model by replacing (P1) in (2.8) by AC-(P1) in (2.25), which will be called AC-FLD in the sequel.

2.5.2 Verification: Adaptation of VOTE(-PG) to AC-VOTE(-PG)

We now show how to adapt the verification algorithms VOTE. Although Lemma 2.5.1 guarantees that VOTE/VOTE-PG can still be used to verify the estimated line states under the AC model, they are developed under the DC model and thus have degraded performance (see Section 2.6.1). To address this issue, we will develop AC-based counterparts of VOTE(-PG) by deriving the counterpart of Lemma 2.3.2 for AC-FLD, which is the foundation of the verification algorithms.

To begin with, we will transform (2.25) into an equivalent LP without equality constraints, as in the transformation of (P1) into (2.11). To achieve this, we notice that any feasible $(\boldsymbol{p}'_H, \boldsymbol{q}'_H, \boldsymbol{x}_H)$ satisfying (2.22) can be represented as (2.26):

$$\begin{bmatrix} \boldsymbol{\Delta}_{p,H} \\ \boldsymbol{\Delta}_{q,H} \\ \boldsymbol{x}_H \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}^*_{p,H} \\ \boldsymbol{\Delta}^*_{q,H} \\ \boldsymbol{x}^*_H \end{bmatrix} + \sum_{e \in E_H} c_e \begin{bmatrix} \tilde{\boldsymbol{d}}_{p,H,e} \\ \tilde{\boldsymbol{d}}_{q,H,e} \\ \boldsymbol{u}_e \end{bmatrix}. \tag{2.26}$$

Then, we modify the definitions in (2.12) as follows: we keep $\boldsymbol{A}_x, \boldsymbol{W}, \boldsymbol{g}_x, \boldsymbol{g}_w$ the same, and redefine $\boldsymbol{A}_D$ and $\boldsymbol{g}_D$ as

$$\begin{aligned} \boldsymbol{A}_D^T := \big[& \boldsymbol{\Lambda}_L^T \tilde{\boldsymbol{D}}_{p,H}^T, -\boldsymbol{\Lambda}_L^T \tilde{\boldsymbol{D}}_{p,H}^T, -\boldsymbol{\Lambda}_S^T \tilde{\boldsymbol{D}}_{p,H}^T, \boldsymbol{\Lambda}_S^T \tilde{\boldsymbol{D}}_{p,H}^T \\ & \boldsymbol{\Lambda}_I^T \tilde{\boldsymbol{D}}_{q,H}^T, -\boldsymbol{\Lambda}_I^T \tilde{\boldsymbol{D}}_{q,H}^T, -\boldsymbol{\Lambda}_C^T \tilde{\boldsymbol{D}}_{q,H}^T, \boldsymbol{\Lambda}_C^T \tilde{\boldsymbol{D}}_{q,H}^T \big], \end{aligned} \tag{2.27a}$$

$$\begin{aligned} \boldsymbol{g}_D^T := \big[& -\boldsymbol{\Lambda}_L^T \boldsymbol{\Delta}_{p,H}^{*T}, -\boldsymbol{\Lambda}_L^T \boldsymbol{p}_H'^{*T}, \boldsymbol{\Lambda}_S^T \boldsymbol{\Delta}_{p,H}^{*T}, \boldsymbol{\Lambda}_S \boldsymbol{p}_H'^{*T} \\ & -\boldsymbol{\Lambda}_I^T \boldsymbol{\Delta}_{q,H}^{*T}, -\boldsymbol{\Lambda}_I^T \boldsymbol{q}_H'^{*T}, \boldsymbol{\Lambda}_C^T \boldsymbol{\Delta}_{q,H}^{*T}, \boldsymbol{\Lambda}_C^T \boldsymbol{q}_H'^{*T} \big]. \end{aligned} \tag{2.27b}$$

Equipped with (2.26) and (2.27), we can obtain the following LP that is equivalent to

(2.25):

$$\min_{\boldsymbol{c}} \quad \mathbf{1}^T \boldsymbol{c} \tag{2.28a}$$

$$\text{s.t.} \quad \boldsymbol{A}_D \boldsymbol{c} \leq \boldsymbol{g}_D, \tag{2.28b}$$

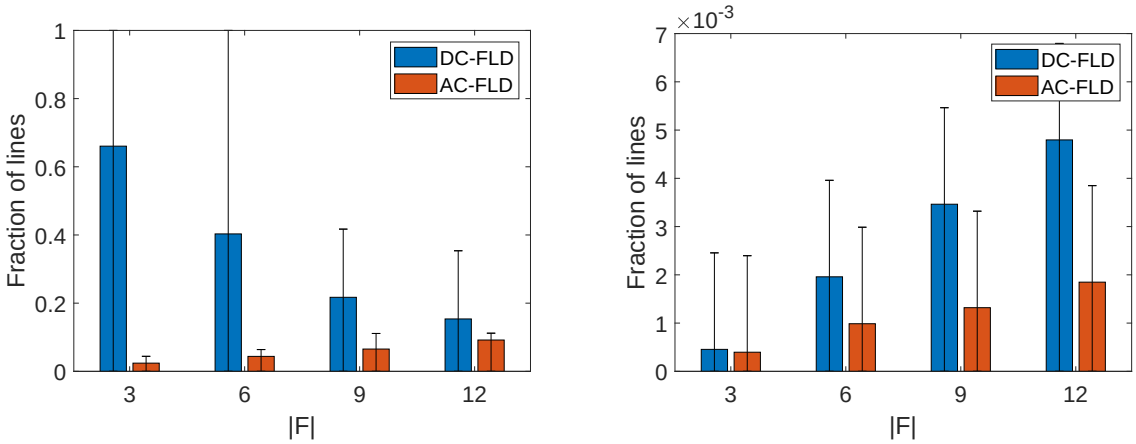
$$\boldsymbol{A}_x \boldsymbol{c} \leq \boldsymbol{g}_x, \tag{2.28c}$$

where (2.28b) corresponds to (2.25b) while (2.28c) corresponds to (2.25c). Since the feasible region of (2.11) can also be written in the form of (2.28b)-(2.28c), AC-Lemma 2.3.2 for AC-FLD has the same form as Lemma 2.3.2 with $\boldsymbol{A}_D$ and $\boldsymbol{g}_D$ redefined as in (2.27). See Appendix 2.8.1 for the proof of AC-Lemma 2.3.2.

Since Theorem 2.4.1-2.4.4 are all proved by contradiction based on Lemma 2.3.2, the corresponding algorithms (VOTE based on Theorem 2.4.1-2.4.3 and VOTE-PG based on Theorem 2.4.4) can be easily adapted to AC-VOTE and AC-VOTE-PG with changed $\boldsymbol{A}_D$ and $\boldsymbol{g}_D$. In Appendix 2.8.5, we provide the adapted theorems and discuss how they are used in AC-VOTE and AC-VOTE-PG.

2.6 Performance Evaluation

We will primarily evaluate our findings on the Polish power grid ("Polish system - winter 1999-2000 peak") [54] with 2383 nodes and 2886 lines (where parallel lines are combined). The ground-truth power grid states are generated according to AC power flow model. Key solutions (FLD, VOTE and VOTE-PG) will also be evaluated on the IEEE 300-bus system extracted from MATPOWER [54] to test their generality. We generate the attacked area H by randomly choosing one node as a starting point and performing a BFS to obtain H with a predetermined $|V_H|$. The generated H consists of buses topologically close to each other, which will intuitively share communication lines in connecting to the control center and can thus be blocked together once a cyber attack jams some of these lines. Note, however, that our solution does not depend on this specific way of forming H. We then randomly choose $|F|$ lines within H to fail. We vary $|V_H|$ and $|F|$ to explore different settings, and for each setting, we generate 70 different H's and 300 different F's per H. In contrast to previous works [19, 24–27] where $|F| \leq 3$, we focus on the scenarios where both $|V_H|$ and $|F|$ are large such that there are likely to be internal nodes in H whose post-attack active power injections cannot be recovered, and there are likely to be island formation in the post-attack grid. In our simulations, we assume that all viable islands have survived the attack (i.e., no frequency collapse),



(a) Miss rate (b) False alarm rate

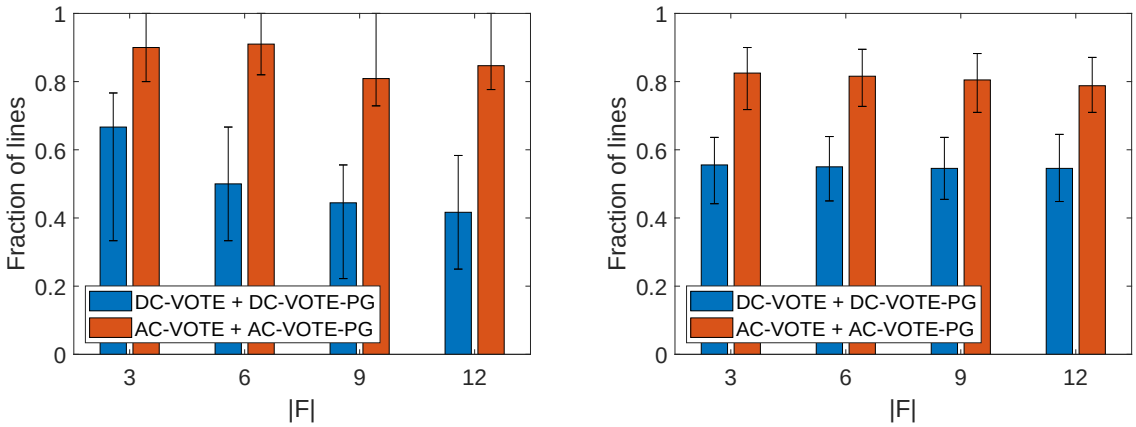
Figure 2.5. Performance of DC-FLD under the AC power flow model in Polish system ($|V_H| = 40$).

but this assumption is not necessary for our algorithms.

We evaluate two types of metrics: (1) how accurate FLD is, and (2) how often the line states estimated by FLD can be verified. Each evaluated metric is shown via the mean and the $25^{\text{th}}/75^{\text{th}}$ percentile (indicated by the error bars) when applicable. The threshold η is set to 0.5.

2.6.1 Performance Loss of DC-based Algorithms

We start by evaluating the performance of the DC-based versions of FLD, VOTE, and VOTE-PG (denoted by DC-*) under the AC power flow model, since they are applicable under the AC model as discussed in Lemma 2.5.1. In Fig. 2.5, we compare the performance of DC-FLD and AC-FLD in terms of miss rate and false alarm rate. In Fig. 2.6, we compare the performance of the combination of DC-VOTE and DC-VOTE-PG with their AC variants. We observe that although the DC-based algorithms are still applicable under the AC power flow model, the approximation error of the DC model leads to performance degradation in both detection and verification. Such observations validate the importance of deriving their AC variants, as shown in Section 2.5. *In the rest of this section, all algorithms (including both proposed and benchmark algorithms) refer to their AC variants developed as in Section 2.5.*



(a) Failed lines (b) Operational lines

Figure 2.6. Performance of DC-VOTE + DC-VOTE-PG under the AC power flow model in Polish system ($|V_H| = 40$).

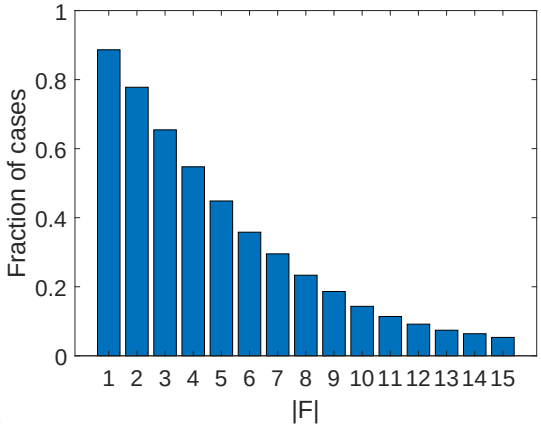
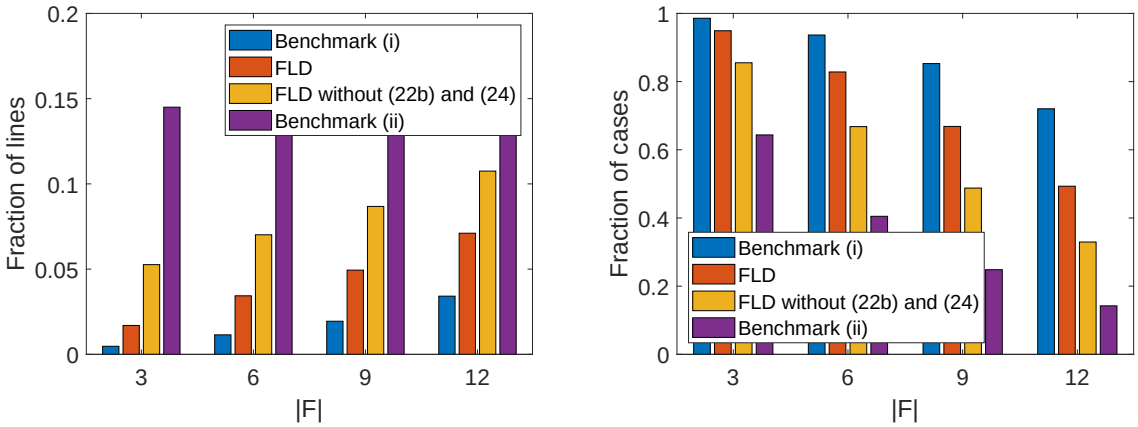


Figure 2.7. Prob. that assuming $\boldsymbol{\Delta} = \mathbf{0}$ leads to a feasible solution in Polish system ($|V_H| = 40$).

2.6.2 Performance of Line State Recovery

First, we observe that for a nontrivial size of H ($|V_H| \geq 20$), there almost always exists $u \in V_H$ for which we cannot recover $\Delta_{H,u}$ by Lemma 2.4.1. We also observe that the solution in [19] (which assumes $\boldsymbol{\Delta} = \mathbf{0}$) is often infeasible, as shown in Fig. 2.7. These observations confirm the necessity of jointly estimating F and $\boldsymbol{\Delta}_H$ during failure localization.

Next, we compare FLD with benchmarks in localizing the failed lines. We consider two benchmarks: (i) the solution extended from [39], i.e., i.e., estimating F by $\text{supp}(\boldsymbol{y})$ for the solution to $\min \|\boldsymbol{y}\|_1$ s.t. (2.22), assuming the knowledge of true $\boldsymbol{\Delta}_{p,H}$ and $\boldsymbol{\Delta}_{q,H}$, and (ii) $\min \|\boldsymbol{y}\|_1$ s.t. $\|\boldsymbol{p}_H - \text{Re}\left([\vec{\boldsymbol{v}}'_H]\overline{\boldsymbol{Y}_{H|G}\vec{\boldsymbol{v}}'}\right) + \tilde{\boldsymbol{D}}_{p,H}\boldsymbol{y}_H\| \leq \|\boldsymbol{p}_H\|$, which is adapted from



(a) Miss rate (b) Probability of no miss

Figure 2.8. Performance comparison on miss rate in Polish system ($|V_H| = 40$).

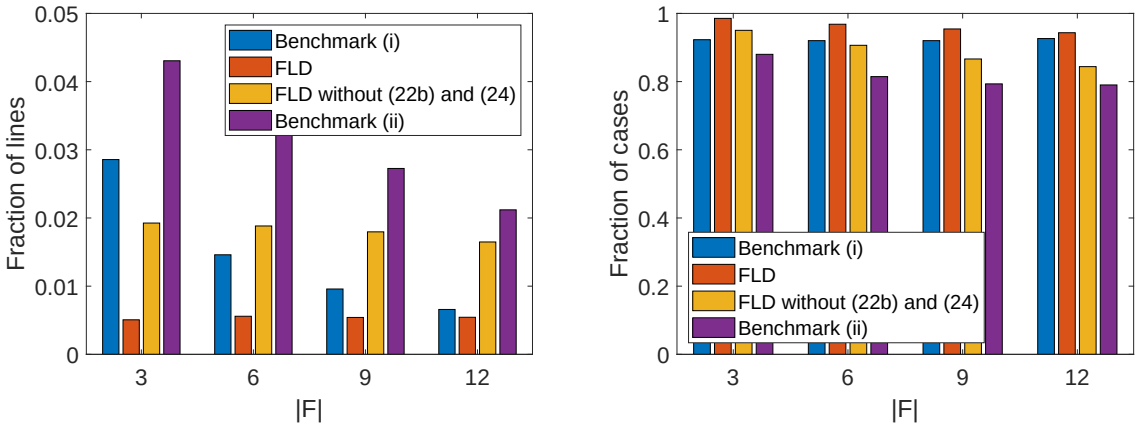
the solutions in [24, 25]. In addition, we consider a variant of AC-FLD that removes the constraints (2.24) and (2.22b) to see the importance of adding constraints on reactive power. Note that benchmark (i) should be treated as a "performance upper bound" as it assumes more knowledge (of $\boldsymbol{\Delta}_H$) than our proposed algorithm.

As shown in Fig. 2.8, benchmark (i) demonstrates the best performance with regard to both the miss rate and the probability of having no miss, while FLD performs much better than benchmark (ii). This confirms the importance of knowing or estimating power injection changes in failure localization. Regarding the false alarm as shown in Fig. 2.9, FLD performs even better than benchmark (i). This is because the decision variable $\boldsymbol{x}$ in benchmark (i) combines the information about both the failed lines and the phase angles $\boldsymbol{\theta}'_H$, and thus does not fully exploit the knowledge of $\boldsymbol{\theta}'_H$. We also notice that adding the constraints (2.24) and (2.22b) can significantly improve detection accuracy by exploiting the knowledge on reactive power injections. Furthermore, from the specific number of false alarms/misses in Fig. 2.10, we see that besides having very few false alarms, FLD also correctly detects most of the failed lines with only a couple of misses for the majority of the time.

2.6.3 Performance of Line State Verification

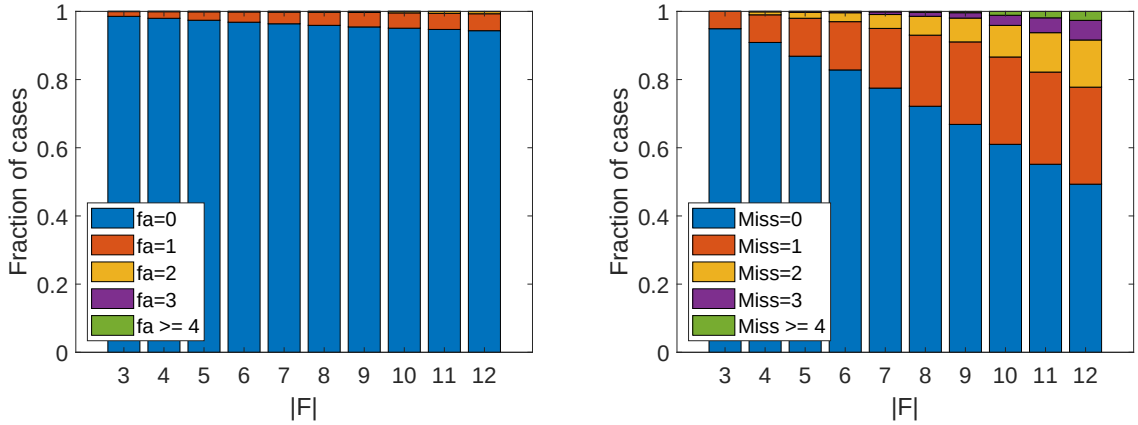
In this subsection, we evaluate the performance of the proposed verification algorithms (VOTE and VOTE-PG) in terms of the fraction of verifiable lines.

We first evaluate the fraction of verifiable lines in E_a (lines in 1-edge cuts) and E_c (lines in 2-edge cuts, i.e., $E_c := \bigcup_{s\in\mathcal{E}_c} s$), as shown in Fig. 2.11. For each generated case (combination of H and F), denote $E_{a,v} := E_a \cap E_v$ and $E_{c,v} := E_c \cap E_v$. Then in



(a) False alarm rate (b) Probability of no false alarm

Figure 2.9. Performance comparison on false alarm rate in Polish system ($|V_H| = 40$).

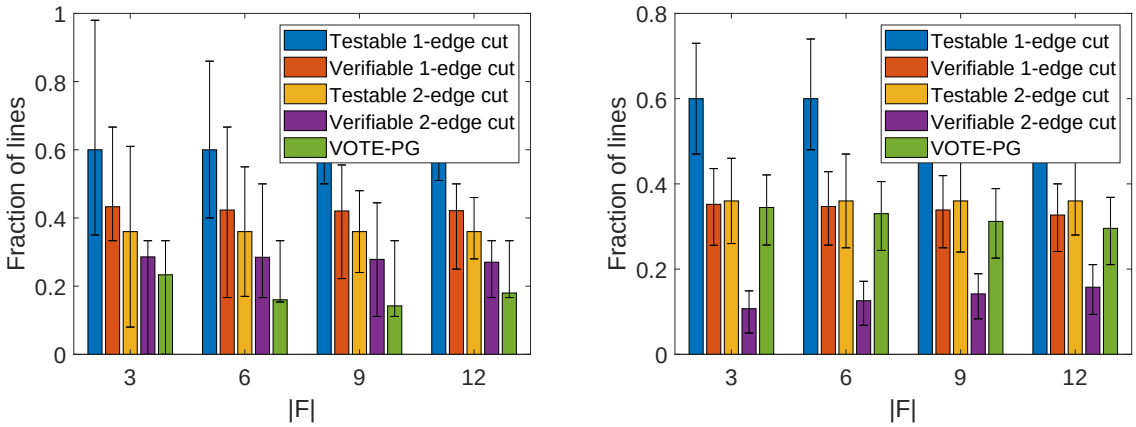


(a) False alarm (b) Miss

Figure 2.10. Number of false alarms/misses of FLD in Polish system ($|V_H| = 40$).

Fig. 2.11(a), we evaluate the fractions of testable and verifiable lines in E_a (E_c) among failed lines, i.e., $\frac{|E_a \cap F|}{|F|}$ ($\frac{|E_c \cap F|}{|F|}$) and $\frac{|E_{a,v} \cap F|}{|F|}$ ($\frac{|E_{c,v} \cap F|}{|F|}$). The evaluation for operational lines is conducted similarly in Fig. 2.11(b). As can be seen, (i) the fractions of testable and verifiable lines both stay almost constant with varying $|F|$, which demonstrates the robustness of VOTE; (ii) among the testable lines ($E_a \cup E_c$), most of the failed lines are verifiable, but only half of the operational lines are verifiable.

Next, we use two metrics to evaluate the value of VOTE-PG. The first metric is the fraction of lines verified by VOTE-PG but not VOTE, as shown in Fig. 2.11 as 'VOTE-PG'. The second is the percentage of cases that VOTE-PG can verify additional lines, given in Table 2.3. We observe that VOTE-PG can usually verify more lines based on the results of VOTE.



(a) Fraction of failed lines (b) Fraction of operational lines

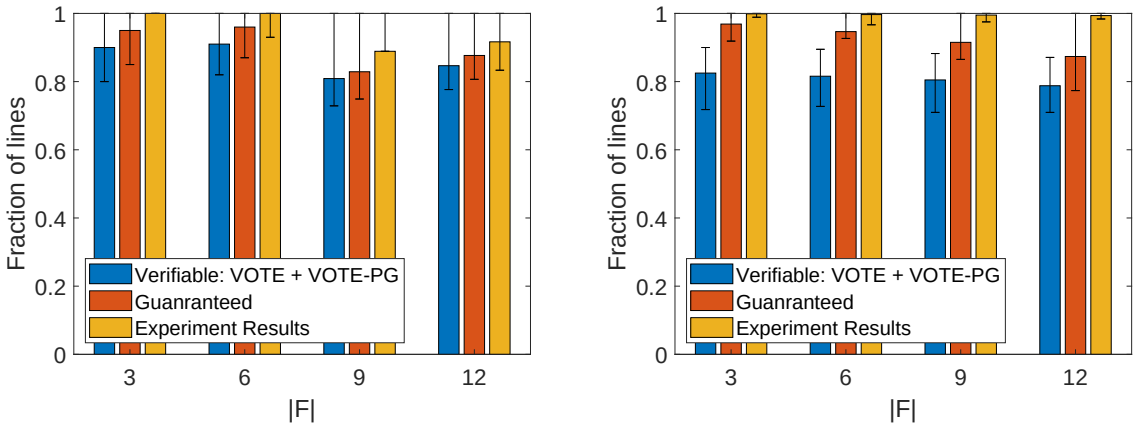
Figure 2.11. Fraction of testable/verifiable lines in Polish system ($|V_H| = 40$).

Table 2.3. Percentage of cases that VOTE-PG verifies additional lines in Polish system

Type of lines	$\|F\| = 3$	$\|F\| = 6$	$\|F\| = 9$	$\|F\| = 12$
Failed lines	24.12%	38.94%	48.72%	57.43%
Operational lines	88.15%	91.45%	92.13%	92.45%
All lines	90.34%	93.61%	95.22%	95.71%

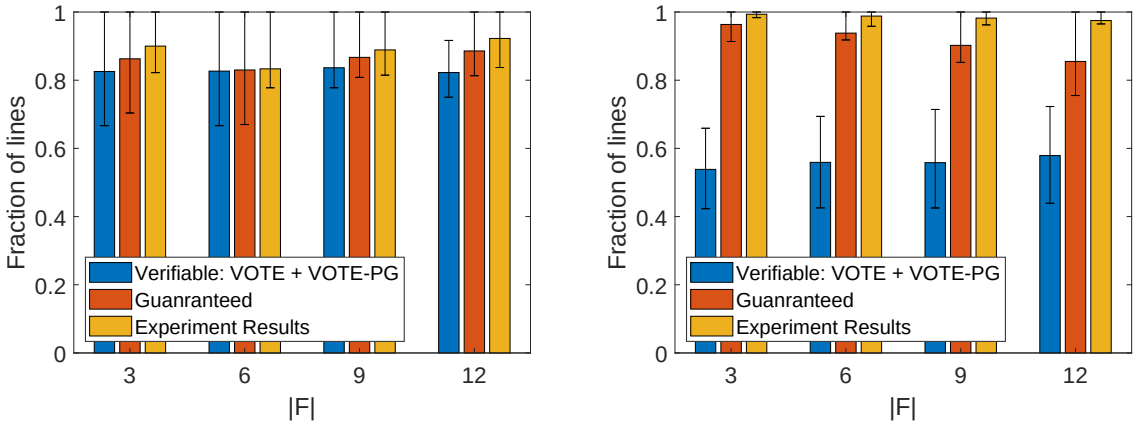
Then, we compare the fraction of verifiable lines ('Verifiable – VOTE + VOTE-PG') to the fraction of lines whose states are guaranteed to be correctly estimated by FLD according to Lemma 2.3.2 ('Guaranteed') and the actual fraction of lines whose states are correctly estimated by FLD ('Experiment Results'), as shown in Fig. 2.12. We see that over 80% of the lines are verifiable. Nevertheless, the fraction of lines whose states are correctly estimated by FLD is even higher: out of all the failed lines, over 80% will be estimated as failed and verified as so, while another 10% will be estimated as failed but not verified; out of all the operational lines, over 80% will be estimated and verified as operational, while the rest will also be estimated as operational but not verified. We have confirmed that the set of verifiable lines is a subset of the set of lines for which FLD is guaranteed to be correct (by Lemma 2.3.2), which is in turn a subset of the set of correctly identified lines.

To validate our key observations, we repeat the experiments in Fig. 2.12 on the IEEE 300-bus system, as shown in Fig. 2.13. Compared with the results from the Polish system, most of the results from the IEEE 300-bus system are qualitatively similar. One notable difference is that although most of the failed lines remain verifiable in the 300-bus system, only half of the operational lines are verifiable. This indicates that most (80-90%) of the unverifiable lines are operational. To understand such a phenomenon, we observe that



(a) Fraction of failed lines (b) Fraction of operational lines

Figure 2.12. Comparison between verifiable lines, theoretically guaranteed lines, and actually correctly identified lines in Polish system ($|V_H| = 40$).



(a) Fraction of failed lines (b) Fraction of operational lines

Figure 2.13. Comparison between verifiable lines, theoretically guaranteed lines, and actually correctly identified lines in IEEE 300-bus system ($|V_H| = 40$).

many operational lines carry small post-attack power flows, which makes the conditions in Theorem 2.4.1-2.4.3 hard to satisfy. On the contrary, the values of hypothetical power flows on failed lines are usually large.

2.6.4 Summary of Observations

1. While the DC-based detection/verification algorithms can be applied in a grid that follows AC power flow equations, their AC-based variants provide better performance.

2. Under the possibility of islanding caused by the physical attack, existing failed line detection algorithms fail with high probability due to the change of power injections (Fig. 2.7), but the proposed FLD can handle the change of power injections and achieve high accuracy.

3. Despite its high accuracy, FLD can still miss failed lines and falsely report failures on operational lines, which will cause problems during recovery.

4. The proposed line state verification algorithms (VOTE and VOTE-PG) can substantially reduce the waste of resources during recovery by providing reliable information on the verifiable line states.

2.7 Conclusion

We investigated the problem of power grid state estimation under cyber-physical attacks that may decompose the grid into islands. Our focus was on recovering the line states within the attack area, due to an observation that existing solutions and their recovery conditions for recovering the phase angles (developed for the case of connected post-attack grid) remain applicable in the case of islanding. To handle the challenge of unknown changes in the power injections within the attack area caused by islanding, under the DC model we proposed an LP-based algorithm to jointly estimate the line states and the power injection changes within the attacked area. We established theoretical conditions under which the line states estimated by the proposed algorithm are guaranteed to be correct, including both more general conditions that depend on the ground-truth line states and less general conditions that only depend on observable information. The latter conditions are further used to develop two polynomial-time algorithms to verify the correctness of the estimated line states. In addition, we extend all results obtained under the DC model to their variants under the AC power flow model. Our evaluations based on the Polish power grid and the IEEE 300-bus system showed that the proposed algorithm is highly accurate in localizing the failed lines, and the correctness of its output can be verified in the majority of cases.

Compared to the previous solutions for line state estimation that label lines with binary states (failed/operational) without guaranteed correctness, our solution labels lines with ternary states (failed/operational/unverifiable), where the states of verifiable lines are identified with guaranteed correctness. This, together with the observation that most of the unverifiable lines are operational, provides valuable information for planning

repair/restoration in the recovery process.

2.8 Appendices

2.8.1 Appendix A: Additional Proofs

Lemma 2.3.1. We will prove the claim by a reduction from the *subset sum problem*, which is known to be NP-hard [55]. Given any set of non-negative integers $\{f_i \geq 0\}_{i=1}^n$ and a target value T, the subset sum problem determines whether there exists $\{x_i \in \{0,1\}\}_{i=1}^n$ such that $\sum_{i=1}^n f_i x_i = T$. For each subset sum instance, we construct the following star-shaped attacked area H: let $H = (V_H, E_H)$ such that V_H is composed of $n+1$ nodes, where node u_0 is the hub with $p_{u_0} = 0$ and $\theta'_{u_0} = 0$, and node u_i ($i \in [n]$ for $[n] := \{1, \ldots, n\}$) is incident to only one link $e_i = (u_0, u_i)$, with $p_{u_i} = -f_i$, $\theta'_{u_i} = -f_i$, and $r_{e_i} = 1$. In addition, u_0 is connected to $v \in V_{\bar{H}}$, with $\theta'_v = \sum_{i=1}^n f_i - T$, through link $e_0 = (u_0, v)$ with $r_{e_0} = 1$.

By substituting (2.4) and $p_{u_0} = 0$, (2.6b) for node u_0 becomes $\tilde{\boldsymbol{D}}_{H,u_0} \boldsymbol{x}_H = \boldsymbol{B}_{u_0|G} \boldsymbol{\theta}'$, where $\tilde{\boldsymbol{D}}_{H,u_0}$ is the row of $\tilde{\boldsymbol{D}}_H$ corresponding to node u_0. Since $(\tilde{D}_{H,u_0})_i = \frac{\theta'_{u_0} - \theta'_{u_i}}{r_{e_i}}$, it is easy to check that $\tilde{\boldsymbol{D}}_{H,u_0} \boldsymbol{x}_H = \sum_{i=1}^n f_i x_i$. Moreover, $\boldsymbol{B}_{u_0|G} \boldsymbol{\theta}' = \sum_{i=1}^n f_i + \frac{\theta'_{u_0} - \theta'_v}{r_{e_0}} = T$. Since u_i ($i \in [n]$) is connected to only one link $e_i = (u_0, u_i)$, we have that $\tilde{\boldsymbol{D}}_{H,u_i} \boldsymbol{x}_H = -f_i x_i$ and $\boldsymbol{B}_{u_i|G} \boldsymbol{\theta}' = -f_i$. Thus, (2.6b) for u_i becomes $-f_i \leq -f_i x_i \leq 0$, which is satisfied whatever value x_i takes. Therefore, a subset sum instance returns true if and only if the instance of (P0) constructed as above is feasible, which completes the proof. □

Lemma 2.3.2. We prove the lemma in two steps. First, note that $\boldsymbol{c}^* = \boldsymbol{0}$ corresponding to the ground-truth F is feasible for (2.11). If $\hat{F}$ is returned by Alg. 1 with $e \in Q_m$, there must exist a corresponding optimal solution $\hat{\boldsymbol{c}}$ to (2.11) with $\hat{c}_e \leq \eta - 1$ and $\boldsymbol{1}^T \hat{\boldsymbol{c}} \leq \boldsymbol{1}^T \boldsymbol{c}^* = 0$. Together with the feasibility constraints in (2.11), $\hat{\boldsymbol{c}}$ must satisfy

$$[\boldsymbol{A}_D^T, \boldsymbol{A}_x^T, \boldsymbol{W}^T, \boldsymbol{1}]^T \hat{\boldsymbol{c}} \leq [\boldsymbol{g}_D^T, \boldsymbol{g}_x^T, \boldsymbol{g}_w^T, 0]^T, \tag{2.29}$$

where $\boldsymbol{W}$ and $\boldsymbol{g}_w$ are defined such that $e \in Q_m$. To prove $e \notin Q_m$, we only need to show the infeasibility of (2.29), which can be proved if there is no solution to (2.29) when $\boldsymbol{W}$ and $\boldsymbol{g}_w$ are defined for $Q_m = \{e\}, Q_f = \emptyset$. This is because a linear system must be infeasible if there is no solution to a subset of its inequalities. According to Gale's theorem of alternative [56], there is no solution to (2.29) if and only if there exists solutions $\boldsymbol{z} \geq \boldsymbol{0}$ to (2.13), which completes the proof. □

Theorem 2.3.1. We will prove by showing that there is a solution to (2.13) for $Q_f = \emptyset$ and $Q_m = \{l\}$ where $l \in E_U$. We prove this by directly constructing a solution $\boldsymbol{z}$ for (2.13) as follows: $\forall u \in U$, if $\tilde{D}_{U,l} < 0$, set $z_{D,u} = 1$; otherwise, set $z_{D,-u} = 1$. Set $z_{w,m,l} = |\tilde{D}_{U,l}|$, $z_{x-,e'} = |\tilde{D}_{U,e'}|$ for $e' \in E_U \setminus F$, $z_{x+,e} = |\tilde{D}_{U,e}|$ for $e \in E_U \cap F \setminus \{l\}$, and other entries of $\boldsymbol{z}$ to 0. Note that $(x^*)_{e'} = 0, \forall e' \in E_U \setminus F$, and $(1-x^*)_e = 0, \forall e \in E_U \cap F$. Then, we will demonstrate why (2.13) is satisfied under this assignment of $\boldsymbol{z}$. First, (2.14) for link l is expanded as $-|\tilde{D}_{U,l}| + z_{w,m,e} = 0$, and (2.14) for $e \in F \setminus \{l\}$ is expanded as $-|\sum_{u\in U} \tilde{D}_{u,e}| + z_{x+,e} = -|\tilde{D}_{U,e}| + z_{x+,e} = 0$ due to condition 1). Second, since $S_U = \emptyset$, for all $e' \in E_U \setminus F$, the corresponding row in (2.13a) is expanded into $|\tilde{D}_{U,e'}| - z_{x-,e'} = 0$. Other rows of (2.13a) holds trivially since they only involve the zero-entries in the constructed $\boldsymbol{z}$. Thus, (2.13a) holds under this assignment. As for (2.13b), its l.h.s can be expanded as $f_{U,g} + (\eta - 1)|\tilde{D}_{U,l}| < 0$ due to condition 3). According to Lemma 2.3.2, $l \in E_U \cap F$ will not be missed, which completes the proof. □

Theorem 2.3.2. Similar to the proof of Theorem 2.3.1, we will prove by showing that there is a solution to (2.13) for $Q_f = \{l\}$ and $Q_m = \emptyset$ where $l \in E_U$. We construct the following $\boldsymbol{z}$: $\forall u \in U$, if $\tilde{D}_{U,l} < 0$, set $z_{D,u} = 1$; otherwise, set $z_{D,-u} = 1$. Set $z_{w,f,l} = |\tilde{D}_{U,l}|$, $z_{x-,e'} = |\tilde{D}_{U,e'}|$ for $e' \in E_U \setminus (F \cup \{l\})$, $z_{x+,e} = |\tilde{D}_{U,e}|$ for $e \in E_U \cap F$, and other entries of $\boldsymbol{z}$ to 0. Then, it is easy to check that (2.13a) is satisfied. As for (2.13b), considering that $\boldsymbol{g}_x^T \boldsymbol{z}_x = \sum_{e' \in E_U \setminus F} x^*_{e'} + \sum_{e \in E_U \cap F}(1 - x^*_e) = 0$ since $(x^*)_{e'} = 0$, $\forall e' \in E_U \setminus F$ and $(1 - x^*)_e = 0$, $\forall e \in E_U \cap F$, the l.h.s of (2.13b) can be expanded as

$$\boldsymbol{g}_D^T \boldsymbol{z}_D + \boldsymbol{g}_x^T \boldsymbol{z}_x - \eta z_{w,f,l} = f_{U,g} - \eta |\tilde{D}_{u,l}| < 0, \tag{2.30}$$

where $f_{U,g} = \sum_{u\in U} g_{D,u}$ if $\tilde{D}_{U,l} > 0$ and $f_{U,g} = \sum_{u\in U} g_{D,-u}$ if $\tilde{D}_{U,l} < 0$, and the last inequality holds due to condition 3). Thus, according to Lemma 2.3.2, $l \notin \hat{F}$, which completes the proof. □

Lemma 2.4.1. As failures can only occur within E_H, nodes in $N(v; \bar{H})$ must be in the same island as v after the attack. Under the proportional load shedding policy, we know that (i) if $\exists u \in N(v; \bar{H})$ of the same type as v, then we can recover the post-attack active power at v by $p'_v = p_v p'_u / p_u$ and thus recover Δ_v; (ii) if $\exists u \in N(v; \bar{H})$ of a different type from v (e.g., u is a generator bus but v is a load bus) and $\Delta_u \neq 0$, then Δ_v must be zero. This proves the claim. □

Theorem 2.4.1. We first prove the case that $\tilde{D}_{U,e_1} \tilde{D}_{U,e_2} < 0$. Given $e_1, e_2 \in E_H \setminus \hat{F}$ where $\hat{F}$ is returned by Alg. 1, there are 3 possible forms of mistakes when the ground truth

failed link set F is unknown, and we will prove the impossibility for each of them. If $e_1 \in F, e_2 \in E_H \setminus F$, Theorem 2.3.1 guarantees that $e_1 \notin Q_m$ due to condition 1), which introduces contradiction. Similarly, $e_2 \in F, e_1 \in E_H \setminus F$ is also impossible. If $e_1, e_2 \in Q_m$, assume without loss of generality that $\eta < 1 - \frac{\hat{f}_{U,g}+|\tilde{D}_{U,e_1}|}{|\tilde{D}_{U,e_2}|}$. Then, we construct the following $\boldsymbol{z}$: $\forall u \in U$, $z_{D,u} = 1$ if $\tilde{D}_{U,e_2} < 0$ or $z_{D,-u} = 1$ if $\tilde{D}_{U,e_2} > 0$, $z_{w,m,e_2} = |\tilde{D}_{U,e_2}|$, $z_{x-,e_1} = |\tilde{D}_{U,e_1}|$, and other entries of $\boldsymbol{z}$ are 0. Then, (2.13a) holds for sure and (2.13b) holds since it can be expanded as $\hat{f}_{U,g} + (\eta - 1)|\tilde{D}_{U,e_2}| + |\tilde{D}_{U,e_1}| < 0$ due to condition 2). According to Lemma 2.3.2, it is impossible to have $e_1, e_2 \in Q_m$, which verifies that $e_1, e_2 \in E_H \setminus F$.

Next, for $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} > 0$, we show how to verify e_1. If $e_1 \in Q_m$, regardless of the true state of e_2, we construct the following $\boldsymbol{z}$ for Lemma 2.3.2: $\forall u \in U$, $z_{D,u} = 1$ if $\tilde{D}_{U,e_1} < 0$ or $z_{D,-u} = 1$ if $\tilde{D}_{U,e_1} > 0$, $z_{w,m,e_1} = |\tilde{D}_{U,e_1}|$, $z_{x+,e_2} = |\tilde{D}_{U,e_2}|$, and other entries of $\boldsymbol{z}$ are 0. Then (2.13) holds due to condition 1), which contradicts the assumption that $e_1 \in Q_m$. The verification condition for e_2 can be derived similarly. □

Theorem 2.4.2. We first prove the impossibility of each possible mistake if $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} > 0$. First, we rule out the possibility that $e_1 \in Q_f$, $e_2 \in E_H \setminus F$ according to Theorem 2.3.2 and condition 1). Similarly, according to Theorem 2.3.1 and condition 1), $e_1 \in F$ while $e_2 \in Q_m$ is also impossible. Next, we prove the impossibility of $e_1 \in Q_f, e_2 \in Q_m$ by constructing a solution $\boldsymbol{z}$ to (2.13). Specifically, if $\eta > \frac{\hat{f}_{U,g}+|\tilde{D}_{U,e_2}|}{|\tilde{D}_{U,e_1}|}$, then $\forall u \in U$, we set $z_{D,u} = 1$ if $\tilde{D}_{U,e_1} > 0$ or $z_{D,-u} = 1$ if $\tilde{D}_{U,e_1} < 0$, $z_{w,f,e_1} = |\tilde{D}_{U,e_1}|$, $z_{x-,e_2} = |\tilde{D}_{U,e_2}|$, and other entries of $\boldsymbol{z}$ as 0. If $\eta < 1 - \frac{\hat{f}_{U,g}+|\tilde{D}_{U,e_1}|}{|\tilde{D}_{U,e_2}|}$, then $\forall u \in U$, we set $z_{D,u} = 1$ if $\tilde{D}_{U,e_2} < 0$ or $z_{D,-u} = 1$ if $\tilde{D}_{U,e_2} > 0$, $z_{w,m,e_2} = |\tilde{D}_{U,e_2}|$, $z_{x+,e_1} = |\tilde{D}_{U,e_1}|$, and other entries of $\boldsymbol{z}$ as 0. It is easy to check the satisfaction of (2.13) under both constructions above, which rules out the possibility of $e_1 \in Q_f, e_2 \in Q_m$ according to Lemma 2.3.2 and $e_1 \in F, e_2 \in E_H \setminus F$ is thus guaranteed.

Next, we prove the verification condition for $e_1 \notin Q_f$ if $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} < 0$. We prove by constructing a solution $\boldsymbol{z}$ as follows regardless of the states of e_2: $\forall u \in U$, if $\tilde{D}_{U,e_1} < 0$, we set $z_{D,-u} = 1$; otherwise, we set $z_{D,u} = 1$. Then, we set $z_{w,f,e_1} = |\tilde{D}_{U,e_1}|$, $z_{x+,e_2} = |\tilde{D}_{U,e_2}|$, and other entries of $\boldsymbol{z}$ as 0. Then, (2.13a) holds for sure and (2.13b) holds since it can be expanded as $\hat{f}_{U,g} - \eta|\tilde{D}_{U,e_1}| + |\tilde{D}_{U,e_2}| < 0$ due to condition 1), which rules out the possibility of $e_1 \in Q_f$ according to Lemma 2.3.2 and thus verifies that $e_1 \in F$. The verification condition for $e_2 \notin Q_m$ can be proved similarly. □

Theorem 2.4.3. We only prove the verification condition for $e_1 \in F$ since the condition for e_2 can be proved similarly. We prove by contradiction that constructs a solution to

(2.13) if $e_1 \in Q_f$. Specifically, with condition 1), we can always construct a $\boldsymbol{z}$ for (2.13) as follows regardless of the states of e_2: $\forall u \in U$, $z_{D,u} = 1$ if $\tilde{D}_{U,e_1} > 0$ or $z_{D,-u} = 1$ if $\tilde{D}_{U,e_1} < 0$ and $z_{w,f,e_1} = |\tilde{D}_{U,e_1}|$. In addition, if $\tilde{D}_{U,e_1}\tilde{D}_{U,e_2} > 0$, we set $z_{x-,e_2} = |\tilde{D}_{U,e_2}|$; otherwise, we set $z_{x+,e_2} = |\tilde{D}_{U,e_2}|$. Finally, other entries of $\boldsymbol{z}$ are set as 0. It is easy to check the satisfaction of (2.13a), and (2.13b) holds since it can be expanded as $[\boldsymbol{g}_D^T, \boldsymbol{g}_x^T, \boldsymbol{g}_w^T, \boldsymbol{0}]\boldsymbol{z} \leq \hat{f}_{U,g} + |\tilde{D}_{U,e_2}| - \eta|\tilde{D}_{U,e_1}| < 0$, where the last inequality holds due to condition 1). Thus, we must have $e_1 \notin Q_f$ according to Lemma 2.3.2. □

Theorem 2.4.4. We only prove for the case that $l \in \hat{F}$ since the case that $e \in E_H \setminus \hat{F}$ is similar. First note that if $\exists \boldsymbol{z}_0 \geq \boldsymbol{0}$ that satisfies (2.13) for $\boldsymbol{W}$ constructed according to $Q_f = \{l\}$ and $Q_m = \emptyset$, then for any $\boldsymbol{W}$ corresponding to Q_f that contains l, we can always construct a non-negative solution to (2.13) based on $\boldsymbol{z}_0$ by setting $z_{w,f,e'} = 0, \forall e' \in Q_f \setminus \{l\}$. Thus, according to Lemma 2.3.2, l can be verified as $l \in F$ if $\exists \boldsymbol{z} \geq \boldsymbol{0}$ for (2.13) where $\boldsymbol{W}$ is constructed for $Q_f = \{l\}$ and $Q_m = \emptyset$, since otherwise l must have been estimated to be operational. Thus, we only need to prove that any solution to (2.20) is a solution to (2.13) when $Q_f = \{l\}$ and $Q_m = \emptyset$. To this end, let $\bar{\boldsymbol{z}} \geq \boldsymbol{0}$ be a feasible solution to (2.20). First, (2.13a) holds since it is the same as (2.20a) in this case. As for (2.13b), we have

$$[\boldsymbol{g}_D^T, \boldsymbol{g}_x^T, g_w, \boldsymbol{0}]\bar{\boldsymbol{z}} \leq [\hat{\boldsymbol{g}}_D^T, \hat{\boldsymbol{g}}_x^T, g_w, \boldsymbol{0}]\bar{\boldsymbol{z}} < 0, \tag{2.31}$$

where the first inequality holds since $\boldsymbol{0} \leq [\boldsymbol{g}_D^T, \boldsymbol{g}_x^T] \leq [\hat{\boldsymbol{g}}_D^T, \hat{\boldsymbol{g}}_x^T]$ (element-wise inequality), while the second inequality holds since $\bar{\boldsymbol{z}}$ satisfies (2.20). Therefore, $\bar{\boldsymbol{z}}$ is also a feasible solution to (2.13), which verifies that $l \in F$. □

Lemma 2.5.1. It is easy to see that Alg. 1 without modification is applicable under the AC model.

To prove the applicability of Alg. 2-3, we only need to prove that DC-based Lemma 2.3.2 holds under the AC model since Theorem 2.4.1-2.4.4 are proved by contradiction based on Lemma 2.3.2. It is worth noting that (2.4) no longer holds for the ground-truth $\boldsymbol{\Delta}_p$, $\boldsymbol{x}$ and $\boldsymbol{\theta}'$ due to the difference between DC model and AC model. However, suppose $\boldsymbol{\theta}'_{\text{DC}}$ is the recovered phase angles under the DC model, then there exists a corresponding $\boldsymbol{\Delta}^*_{\text{DC}}$ that is compatible with (2.4) under $\boldsymbol{x}^*$. Then, any solution of (P1) will be in the form of

$$\begin{bmatrix} \boldsymbol{\Delta}_H \\ \boldsymbol{x}_H \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Delta}^*_{\text{DC}} \\ \boldsymbol{x}^*_H \end{bmatrix} + \sum_{e \in E_H} c_e \begin{bmatrix} \tilde{\boldsymbol{d}}_e \\ \boldsymbol{u}_e \end{bmatrix}, \tag{2.32}$$

which is similar to (2.9) with $\boldsymbol{\Delta}^*$ changed into $\boldsymbol{\Delta}^*_{\text{DC}}$. That is to say, (2.11) is still an

equivalent LP of (P1) in (2.8) by replacing $\boldsymbol{g}_D$ as

$$\boldsymbol{g}_D^T = [-(\boldsymbol{\Delta}^*_{\mathrm{DC},L})^T, (-\boldsymbol{p}'_{\mathrm{DC},L})^T, (\boldsymbol{\Delta}^*_{\mathrm{DC},S})^T, (\boldsymbol{p}'_{\mathrm{DC},S})^T]. \tag{2.33}$$

Thus, with $\boldsymbol{g}_D$ changed into (2.33), DC-based Lemma 2.3.2 holds under the AC model, which completes the proof. □

AC-Lemma 2.3.2. Similar to the proof of Lemma 2.3.2, we have $\boldsymbol{c}^* = \mathbf{0}$ that corresponds to the ground-truth F and is feasible for (2.28). If $\hat{F}$ is returned by AC-Alg. 1 with $e \in Q_m$, there must exist a corresponding optimal solution $\hat{\boldsymbol{c}}$ to (2.25) with $\hat{c}_e \leq \eta - 1$ and $\mathbf{1}^T\hat{\boldsymbol{c}} \leq \mathbf{1}^T\boldsymbol{c}^* = 0$. Together with the feasibility constraints (2.28b)-(2.28c), $\hat{\boldsymbol{c}}$ must satisfy

$$[\boldsymbol{A}_D^T, \boldsymbol{A}_x^T, \boldsymbol{W}^T, \mathbf{1}]^T\hat{\boldsymbol{c}} \leq [\boldsymbol{g}_D^T, \boldsymbol{g}_x^T, \boldsymbol{g}_w^T, 0]^T, \tag{2.34}$$

where $\boldsymbol{g}_D$ and $\boldsymbol{A}_D$ are defined in (2.27). It can be seen that (2.34) has exactly the same form of (2.29). Then, following the same proving procedure in Lemma 2.3.2, we can have AC-Lemma 2.3.2. □

2.8.2 Appendix B: Proportional Loadshedding/generation Reduction

We consider adjusting load/generation in the case of islanding [51, 52], under which either the load or the generation (but not both) will be reduced upon the formation of an island. Moreover, if nodes u and v are in the same island and of the same type (both load or generator), then $p'_u/p_u = p'_v/p_v$.

We take the island that initially has supply > demand as an example to explain the policy. Typically, this reduction is performed through governor action, whose droop coefficient (say, R_i for the ith generator) determines the ratio in which generation is decreased. The total reduction in generation is equal to

$$\text{supply} - \text{demand} = \Delta f \sum_{i \in \mathcal{I}} \frac{1}{R_i},$$

where $\mathcal{I}$ is the set of generators in this island, and Δf is the change in frequency in the island after generation reduction, i.e., $p_u - p'_u = \frac{\Delta f}{R_u}$. Typically, $\frac{1}{R_i}$ is chosen proportional to the machine rating of the generator – the higher the machine rating, the larger the value of $\frac{1}{R_i}$. By assuming the proportional load shedding/generation reduction, we consider the generators to be fully loaded before the disturbance.. This implies that

$\frac{1}{R_u} = Kp_u$ for some constant K, and thus $\frac{p'_u}{p_u} = \frac{p'_v}{p_v} = 1 - K\Delta f$ for any two generators u, v in this island.

In the case of demand more than supply in an island, the frequency nadir during the inertial phase becomes low enough to activate underfrequency relays, and the load shedding action that follows leads to balance of supply and demand. Since generators are assumed fully loaded, we do not consider increase of generation.

2.8.3 Appendix C: Recovery of Phase Angles and Voltages

Under the assumption that G remains connected after the attack and thus $\boldsymbol{\Delta} = \mathbf{0}$, [19] showed that the post-attack phase angles $\boldsymbol{\theta}'_H$ can be recovered if the submatrix $\boldsymbol{B}_{\bar{H}|H}$ of the admittance matrix has a full column rank. Below, we will show that the same condition holds without this limiting assumption.

Specifically, we have the following lemma that extends [19, Lemma 1] to the case of arbitrary $\boldsymbol{\Delta}$ ("supp": indices of non-zero entries in the input vector).

Lemma 2.8.1. *supp*$(\boldsymbol{B}(\boldsymbol{\theta} - \boldsymbol{\theta}') - \boldsymbol{\Delta}) \subseteq V_H$.

Proof. For a link (s,t), define a column vector $\boldsymbol{x}_{st} \in \{-1, 0, 1\}^{|V|}$, which has 1 in s-th element, -1 in t-th element, and 0 elsewhere. The failure of links in F changes the admittance matrix by[6]

$$\boldsymbol{B}' = \boldsymbol{B} + \sum_{(s,t)\in F} B_{st}\boldsymbol{x}_{st}\boldsymbol{x}_{st}^T, \tag{2.35}$$

where B_{st} is the (s,t)-th element in $\boldsymbol{B}$. Before the attack, we have $\boldsymbol{B\theta} = \boldsymbol{p}$. After the attack, we have $\boldsymbol{B}'\boldsymbol{\theta}' = \boldsymbol{p}' = \boldsymbol{p} - \Delta$. Therefore, the following holds:

$$\boldsymbol{B\theta} - \boldsymbol{B}'\boldsymbol{\theta}' = \boldsymbol{\Delta} \tag{2.36}$$

$$\Rightarrow \boldsymbol{B}(\boldsymbol{\theta} - \boldsymbol{\theta}') - \boldsymbol{\Delta} = \sum_{(s,t)\in F} B_{st}\boldsymbol{x}_{st}\boldsymbol{x}_{st}^T\boldsymbol{\theta}' \tag{2.37}$$

$$\Rightarrow \text{supp}(\boldsymbol{B}(\boldsymbol{\theta} - \boldsymbol{\theta}') - \boldsymbol{\Delta}) \subseteq \bigcup_{(s,t)\in F} \{s,t\} \subseteq V_H, \tag{2.38}$$

where (2.37) is obtained by plugging in (2.35) into (2.36). □

Using Lemma 2.8.1, we prove that the recovery condition in [19, Theorem 1] remains sufficient even if the post-attack grid may be disconnected.

[6]There was a typo in the proof of [19, Lemma 1], which claimed that $\boldsymbol{B}' = \boldsymbol{B} - \sum_{(s,t)\in F} b_{st}\boldsymbol{x}_{st}\boldsymbol{x}_{st}^T$.

Theorem 2.8.1. *The phase angles $\boldsymbol{\theta}'_H$ within the attacked area can be recovered correctly if $\boldsymbol{B}_{\bar{H}|H}$ has a full column rank.*

Proof. By Lemma 2.8.1, we see that $\boldsymbol{B}_{\bar{H}|G}(\boldsymbol{\theta} - \boldsymbol{\theta}') - \boldsymbol{\Delta}_{\bar{H}} = \mathbf{0}$. Writing this equation in more detail shows that

$$\boldsymbol{B}_{\bar{H}|H}(\boldsymbol{\theta}_H - \boldsymbol{\theta}'_H) + \boldsymbol{B}_{\bar{H}|\bar{H}}(\boldsymbol{\theta}_{\bar{H}} - \boldsymbol{\theta}'_{\bar{H}}) - \boldsymbol{\Delta}_{\bar{H}} = \mathbf{0} \tag{2.39}$$

$$\Rightarrow \boldsymbol{B}_{\bar{H}|H}\boldsymbol{\theta}'_H = \boldsymbol{B}_{\bar{H}|H}\boldsymbol{\theta}_H + \boldsymbol{B}_{\bar{H}|\bar{H}}(\boldsymbol{\theta}_{\bar{H}} - \boldsymbol{\theta}'_{\bar{H}}) - \boldsymbol{\Delta}_{\bar{H}}. \tag{2.40}$$

Since both $\boldsymbol{B}_{\bar{H}|H}$ and the righthand side of (2.40) are known to the control center, we can uniquely recover $\boldsymbol{\theta}'_H$ if $\boldsymbol{B}_{\bar{H}|H}$ has a full column rank. □

The phase angles $\boldsymbol{\theta}'_H$ can be recovered not only through Theorem 2.8.1, but also through the secured PMUs, whose measurements and communications to the control center are much harder to attack due to the security mechanisms in modern WAMPAC network design [57]. It is widely accepted that secured PMUs can be used to defend against cyber attacks [48–50].

In the following, we provide guidelines for choosing nodes on which to install secured PMUs.

Corollary 2.8.1.1. *Let $E_{M,\bar{H}|H} \subseteq E_{\bar{H}|H}$ denote a matching (a set of links without common endpoints) covering nodes $V_{\bar{H}_M} \subseteq V_{\bar{H}}$ and $V_{H_M} \subseteq V_H$. Then $\boldsymbol{\theta}'_H$ can be recovered almost surely [7] if there is a secured PMU measuring the (voltage) phase angle at each $u \in V_H \setminus V_{H_M}$.*

Proof. Since the PMUs at $u \in V_H \setminus V_{H_M}$ will directly report θ'_u, the only unknown term left in (2.40) is $\boldsymbol{\theta}'_{H_M}$. Next, we will show that $\boldsymbol{\theta}'_{H_M}$ can be uniquely determined by (2.40) with the aid of $\theta'_u, u \in V_H \setminus V_{H_M}$ measured by secured sensors. To achieve this, we re-write the left-hand-side (l.h.s) of (2.40) as

$$\boldsymbol{B}_{\bar{H}|H}\boldsymbol{\theta}'_H = \boldsymbol{B}_{\bar{H}|H_M}\boldsymbol{\theta}'_{H_M} + \sum_{u \in V_H \setminus V_{H_M}} \boldsymbol{B}_{\bar{H}|u} \cdot \theta'_u, \tag{2.41}$$

where the second term on the right-hand-side is known due to PMUs. By plugging (2.41) back into (2.40), we obtain an equation in $\boldsymbol{\theta}'_{H_M}$ that can uniquely determine $\boldsymbol{\theta}'_{H_M}$ if $\boldsymbol{B}_{\bar{H}|H_M}$ has a full column rank. This condition holds almost surely according

[7] In probability theory, an event happens almost surely if it happens with probability one. In other words, among all possible combinations of $\{r_e\}_{e \in E}$, the set of reactance values for $E_{\bar{H}|H}$ resulting in $\boldsymbol{\theta}'_H$ uncoverable is a measure zero set in the real space.

to [19, Corollary 2] since the subgraph corresponding to $\boldsymbol{B}_{\bar{H}|H_M}$ contains the matching $E_{M,\bar{H}|H}$. □

Discussion: Under the North American SynchroPhasor Initiative (NASPI) [58], the number of PMUs is steadily growing [59], and installing PMUs is becoming part of routine transmission system upgrades and new construction [60]. Some utilities have achieved full observability in their networks, e.g., Dominion Power has piloted the PMU-based linear state estimator [61, 62]. These trends motivate us to consider failure localization based on phase angles.

Although fully equipped PMUs can measure both voltage phasors at buses and current phasors at their incident lines, each phasor to be measured requires extra instrumentation. To this end, we will show that measuring voltage phasors alone is almost good enough in that: under normal conditions, voltage phasors can be used to compute line currents; after attacks, the voltage phase angles can be used to estimate link states (and hence currents) with high accuracy (see Sections 2.3). Measuring only voltage phasors by PMUs has also been assumed in prior works [48].

Previous discussion is based on DC power flow model. In [39, Lemma 1], the conditions for recovering phase angles [19, Lemma 1] are extended to the AC power flow model to recover complex-valued voltages ($\vec{\boldsymbol{v}}'_H$) after attack. Next, we will show that the conditions in [39, Lemma 1] still hold even if the post-attack grid may be disconnected, as in our extension of [19, Lemma 1] to Theorem 2.8.1.

Theorem 2.8.2. *The voltages $\vec{\boldsymbol{v}}'_H$ within the attacked area can be recovered almost surely if $\exists E_{M,\bar{H}|H} \subseteq E_{\bar{H}|H}$ with $V_{H_M} = V_H$.*

Proof. Due to the attack model discussed in Section 2.2.2, we have $\boldsymbol{l}'_{\bar{H}} = \boldsymbol{Y}'_{\bar{H}|G}\vec{\boldsymbol{v}}' = \boldsymbol{Y}_{\bar{H}|G}\vec{\boldsymbol{v}}'$. Then, by denoting $\overline{x}$ as the conjugate of x and $[\boldsymbol{x}]$ as the diagonal operation of $\boldsymbol{x}$, we have

$$[\vec{\boldsymbol{v}}'_{\bar{H}}]\overline{\boldsymbol{Y}'_{\bar{H}|G}\vec{\boldsymbol{v}}'} = [\vec{\boldsymbol{v}}'_{\bar{H}}]\overline{\boldsymbol{l}'_{\bar{H}}} = \boldsymbol{p}'_{\bar{H}} + j\boldsymbol{q}'_{\bar{H}}, \tag{2.42}$$

which leads to

$$\boldsymbol{\Xi}_{\bar{H}|H}\begin{bmatrix} \mathrm{Re}(\vec{\boldsymbol{v}}'_H) \\ \mathrm{Im}(\vec{\boldsymbol{v}}'_H) \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}'_{\bar{H}} - \mathrm{Re}([\vec{\boldsymbol{v}}'_{\bar{H}}]\overline{\boldsymbol{Y}'_{\bar{H}|\bar{H}}\vec{\boldsymbol{v}}'_{\bar{H}}}) \\ \boldsymbol{q}'_{\bar{H}} - \mathrm{Im}([\vec{\boldsymbol{v}}'_{\bar{H}}]\overline{\boldsymbol{Y}'_{\bar{H}|\bar{H}}\vec{\boldsymbol{v}}'_{\bar{H}}}) \end{bmatrix} \tag{2.43}$$

$$\boldsymbol{\Xi}_{\bar{H}|H} = \begin{bmatrix} \boldsymbol{G}_{\bar{H}|H} & -\boldsymbol{B}_{\bar{H}|H} \\ \boldsymbol{B}_{\bar{H}|H} & \boldsymbol{G}_{\bar{H}|H} \end{bmatrix}. \tag{2.44}$$

It is easy to see that we can uniquely recover both $\mathrm{Re}(\vec{\boldsymbol{v}}'_H)$ and $\mathrm{Im}(\vec{\boldsymbol{v}}'_H)$ if $\boldsymbol{\Xi}_{\bar{H}|H}$ has full column rank. According to [39, Lemma 1] and [19, Corollary 2], $\boldsymbol{\Xi}_{\bar{H}|H}$ has full column rank almost surely if there is a matching between the nodes V_H and $V_{\bar{H}}$ that covers the nodes V_H. □

It is easy to see that Corollary 2.8.1.1 holds under the AC power flow model by comparing (2.43) and (2.40).

2.8.4 Appendix D: Special Case: Known Post-Attack Power Injections

In this section, we extend our results to the special case that the control center can either know that the grid after attack remains connected or fully recover the power injections. We first extend [19, Lemma 2] as follows.

Lemma 2.8.2. *There exists a vector $\boldsymbol{x} \in \mathbb{R}^{|E_H|}$ that satisfies supp$(\boldsymbol{x}) = F$, and*

$$\boldsymbol{D}_H \boldsymbol{x} = \boldsymbol{B}_{H|G}(\boldsymbol{\theta} - \boldsymbol{\theta}') - \boldsymbol{\Delta}_H. \tag{2.45}$$

Proof. Note that by definition, $\boldsymbol{x}_{st}$ defined in the proof of Lemma 2.8.1 is the same as the column corresponding to link (s,t) in $\boldsymbol{D}$. Define a vector $\boldsymbol{y} \in \mathbb{R}^{|E|}$ by

$$y_e = \begin{cases} B_{st}(\theta'_s - \theta'_t) & \text{if } e = (s,t) \in F, \\ 0 & \text{o.w.} \end{cases} \tag{2.46}$$

Then it is easy to see that $\sum_{(s,t)\in F} B_{st}\boldsymbol{x}_{st}\boldsymbol{x}_{st}^T\boldsymbol{\theta}' = \boldsymbol{D}\boldsymbol{y}$. By (2.37), we have $\boldsymbol{B}(\boldsymbol{\theta}-\boldsymbol{\theta}')-\boldsymbol{\Delta} = \boldsymbol{D}\boldsymbol{y}$. Considering only the equations corresponding to V_H yields

$$\boldsymbol{B}_{H|G}(\boldsymbol{\theta} - \boldsymbol{\theta}') - \boldsymbol{\Delta}_H = \boldsymbol{D}_H \boldsymbol{y}_H, \tag{2.47}$$

where we have used the fact that $\boldsymbol{y}_{\bar{H}} = \boldsymbol{0}$. Thus $\boldsymbol{x} = \boldsymbol{y}_H$ satisfies the conditions in the lemma. □

Based on this result, [19, Theorem 2] can be easily extended as follow:

Theorem 2.8.3. *The failed links F within the attacked area can be localized correctly if:*

1. *H is acyclic (i.e., a tree or a set of trees), in which case (2.45) has a unique solution $\boldsymbol{x}$ for which supp$(\boldsymbol{x}) = F$, or*

2. *H is a planar graph satisfying (i) for any cycle C in H, $|C \cap F| < |C \setminus F|$, and (ii) F^* is H^*-separable[8], in which case the optimization* $\min \|\boldsymbol{x}\|_1$ *s.t. (2.45) has a unique solution $\boldsymbol{x}$ for which supp$(\boldsymbol{x}) = F$.*

Proof. Condition (1) is implied by [19, Lemma 3], which proved that $\boldsymbol{D}_H$ has a full column rank if and only if H is acyclic. This combined with Lemma 2.8.2 shows that if H is acyclic, then (2.45) only has one solution, and hence the support of this solution must be F.

Condition (2) is implied by the proof of [19, Theorem 2], which showed that if H satisfies this condition, then any $\boldsymbol{x}$ for (2.45) satisfies $\|\boldsymbol{x}\|_1 \geq \|\boldsymbol{x}^*\|_1$, where $\boldsymbol{x}^*$ is a vector satisfying the conditions in Lemma 2.8.2. Moreover, it showed that $\|\boldsymbol{x}\|_1 = \|\boldsymbol{x}^*\|_1$ only if $\boldsymbol{x} = \boldsymbol{x}^*$. Thus, $\boldsymbol{x}^*$, whose support equals F, can be computed by minimizing $\|\boldsymbol{x}\|_1$ s.t. (2.45). □

Next, based on Theorem 2.3.1-2.3.2, we give an specific condition of H under which F can be correctly recovered even if $\boldsymbol{\Delta}^*$ is unknown.

Corollary 2.8.3.1. *If the grid stays connected after failure, H is acyclic, and H contains either no load bus or no generator bus, then Alg. 1 is guaranteed to detect F correctly, i.e., $\hat{F} = F$.*

Proof. We only prove the case that H contains no generator bus since the other case can be proved similarly. We first prove that any failed link $l \in F$ will not be missed ($l \in \hat{F}$). Under Assumption 1, link l must have one endpoint (say u) such that $\tilde{D}_{u,l} < 0$. Next, we will build a hyper-node U such that the induced subgraph is a tree rooted at node u. Specifically, such hyper-node can be constructed by breadth-first search (BFS) starting from node u. In the first iteration of BFS, we start with $U = \{u\}$ and add a neighbor v_i of u into U if $e = (u, v_i) \in F$ with $\tilde{D}_{u,l}\tilde{D}_{u,e} < 0$ or $e = (u, v_i) \in E_U \setminus F$ with $\tilde{D}_{u,l}\tilde{D}_{u,e} > 0$. Then, we repeatedly add node v into U if $\exists e = (s, v) \in E_U \cap F$ such that $\tilde{D}_{U,l}\tilde{D}_{U,e} < 0$ or $\exists e = (s, v) \in E_U \setminus F$ such that $\tilde{D}_{U,l}\tilde{D}_{U,e} > 0$. This procedure will terminate since H is acyclic, and the constructed U will satisfy condition 1) and condition 2) of Theorem 2.3.1. Since all nodes $u \in U$ are load buses, $\tilde{D}_{U,l} < 0$, and the grid stays connected after failure, we have $f_{U,g} = -\sum_{u \in U} \Delta_u = 0$, which satisfies condition 3) of Theorem 2.3.1. Thus, we have $F \subseteq \hat{F}$.

Next, we show that any operational link $e \in E_H \setminus F$ will not be falsely detected by

[8]Here H^* is the dual graph of H, and F^* is the set of edges in H^* such that each edge in F^* connects a pair of vertices that correspond to adjacent faces in H separated by a failed link.

Alg. 1 ($e \notin \hat{F}$). Under Assumption 1, link e must have have one endpoint (say u) such that $\tilde{D}_{u,e} > 0$. The hyper-node U can be constructed as follows: start with $U = \{u\}$, add node v into U if $\exists e' = (s, v) \in E_U \cap F$ or $\exists e' = (s, v) \in E_U \setminus F$ such that $\tilde{D}_{U,e}\tilde{D}_{U,e'} < 0$. The resulting hyper-node must satisfy condition 1) and condition 2) of Theorem 2.3.2. Again, we have $f_{U,g} = -\sum_{u\in U} \Delta_u = 0$, which leads to satisfaction of condition 3) in Theorem 2.3.2. Therefore, we have $\hat{F} \subseteq F$. □

Now we demonstrate how to modify Alg. 1-3 if the grid after attack is known to stay connected. In this case, Alg. 1 is modified by replacing constraints (2.6a) and (2.6b) with $\boldsymbol{\Delta}_H = \mathbf{0}$ (implied by the assumption of the connected post-attack grid). Next, we demonstrate how Alg. 2-3 will change in this case. To this end, we study the effect of $\boldsymbol{\Delta}_H = \mathbf{0}$ on Lemma 2.3.2. Noting that according to (2.9), any pair of $(\boldsymbol{\Delta}_H, \boldsymbol{x}_H)$ satisfying (2.4) can be represented by $\boldsymbol{c} \in \mathbb{R}^{|E_H|}$ as

$$\boldsymbol{\Delta}_H = \boldsymbol{\Delta}_H^* + \tilde{\boldsymbol{D}}_H \boldsymbol{c}, \quad \boldsymbol{x}_H = \boldsymbol{x}_H^* + \boldsymbol{I}_{|E_H|}\boldsymbol{c}. \tag{2.48}$$

Thus, we have $\tilde{\boldsymbol{D}}_H\boldsymbol{c} = \mathbf{0}$ due to $\boldsymbol{\Delta}_H = \boldsymbol{\Delta}_H^* = \mathbf{0}$, which is equivalent to requiring $\tilde{\boldsymbol{D}}_H\boldsymbol{c} \leq \mathbf{0}$ and $-\tilde{\boldsymbol{D}}_H\boldsymbol{c} \leq \mathbf{0}$. Accordingly, $\boldsymbol{A}_D$ and $\boldsymbol{g}_D$ in (2.13), which is used to model (2.6a) and (2.6b) in Lemma 2.3.2, now become $\boldsymbol{A}_D^T := [\tilde{\boldsymbol{D}}_H^T, -\tilde{\boldsymbol{D}}_H^T], \boldsymbol{g}_D := \mathbf{0}$. The direct implication of $\boldsymbol{g}_D = \mathbf{0}$ is that $f_{U,g} = \sum_{u\in U} f_{u,g} = 0, \forall U \subseteq V_H$. That is to say, Theorems 2.4.1-2.4.3 still hold for the modified Alg. 1 except that $\hat{f}_{U,g} = \mathbf{0}$, which implies the following result:

Corollary 2.8.3.2. *If it is known that the post-attack grid $G' = (V, E \setminus F)$ is connected, then the state of any link that forms a 1-edge cut of H will be identified correctly by a variation of Alg. 1 that replaces the constraints* (2.6a) *and* (2.6b) *by $\boldsymbol{\Delta}_H = \mathbf{0}$.*

Proof. As in the proof of Corollary 2.4.0.1, for any link $e = (u_1, u_2) \in \hat{F}$ forming a cut of H, we can verify that $e \in F$ if $\min\{f_{U_1,g}, f_{U_2,g}\} - \eta|\tilde{D}_{U_1,e}| < 0$ (otherwise, e must have been estimated as operational by Theorem 2.3.2). Since $f_{U_i,g} = 0$ $(i = 1, 2)$ if the grid remains connected after the attack and $|\tilde{D}_{U_1,e}| > 0$ by Assumption 3, $e \in F$ can always be verified. Similar argument applies to any link $l \in E_H \setminus \hat{F}$. □

By Corollary 2.8.3.2, the verification of the link states in E_a can be skipped if the post-attack grid is known to stay connected.

Finally, we experimentally study the benefits of knowing the connectivity, as shown in Fig. 2.14 and Table 2.4. Specifically, 'X-agnostic' denotes the performance of 'X' without knowing the connectivity, while 'X-known' denotes the counterpart that adopts the modification in this section. The meaning of 'X' is the same as that in Fig. 2.12. In

Table 2.4. Percentage of cases of connected post-attack Polish system ($|V_H| = 40$)

$\|F\| = 3$	$\|F\| = 6$	$\|F\| = 9$	$\|F\| = 12$
57.12%	26.33%	11.87%	5.04%

Table 2.5. Percentage of cases of connected post-attack IEEE 300-bus system ($|V_H| = 20$)

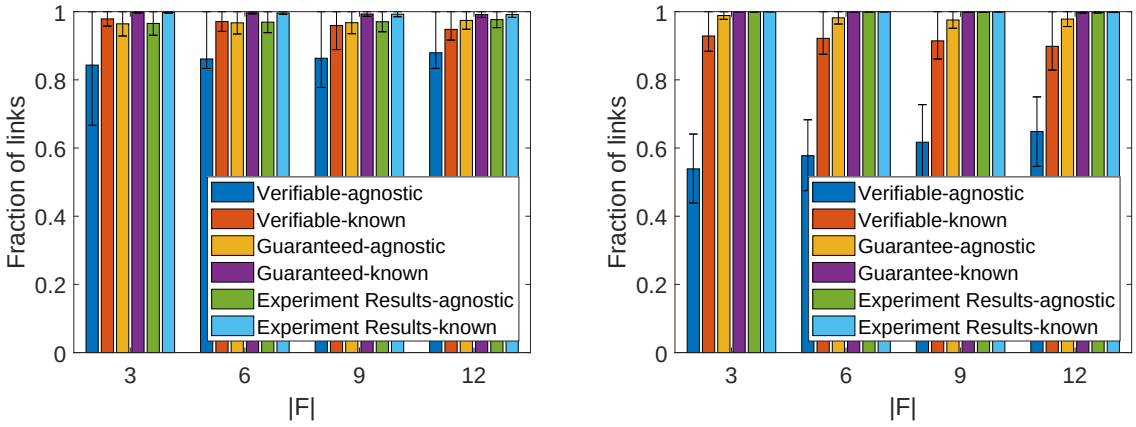
$\|F\| = 2$	$\|F\| = 4$	$\|F\| = 6$	$\|F\| = 8$
73.73%	51.10%	32.89%	18.54%

Table 2.4, we evaluate the percentage of randomly generated cases (H and F) in which the post-attack grid G' remains connected. We observe that (i) the knowledge of connectivity can help verify more than 10% additional failed links and 30% additional operational links; (ii) when $|F|$ is small (e.g., $|F| \leq 3$), G' remains connected in the majority of the cases. These results indicate the value of the knowledge of connectivity. In Fig. 2.15 and Table 2.5, we evaluate the same metrics on IEEE 300-bus system, the results of which are similar as that in Polish grid.

2.8.5 Appendix E: Adaptation of Verification Algorithms to AC Power Flow Model

In this section, we will show how to obtain AC-Theorem 2.4.1–2.4.4 and the associated AC-VOTE and AC-VOTE-PG.

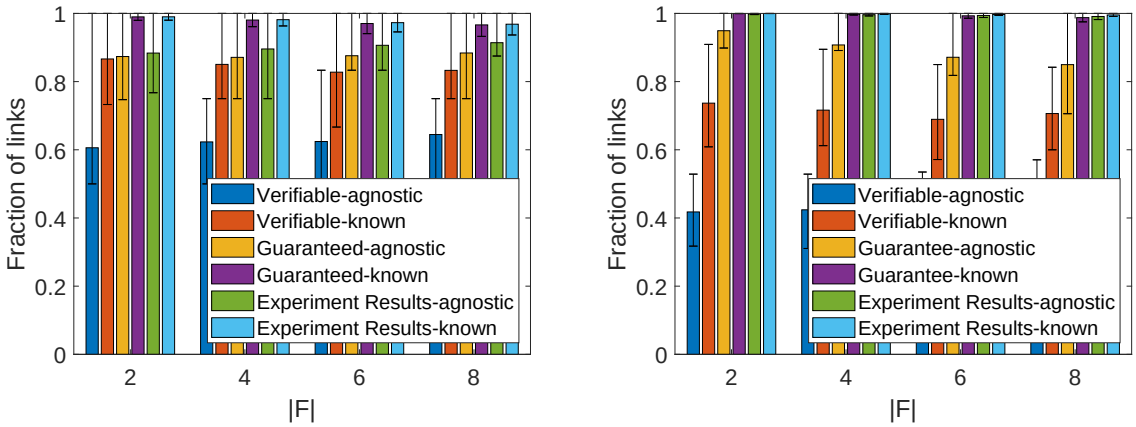
Recall that the AC variants of $\boldsymbol{g}_D$ and $\boldsymbol{A}_D$ are given in (2.27). In this section, $\boldsymbol{g}_D$, $\boldsymbol{A}_D$, and the associated $f_{U,g}$ refer to the values given in (2.27). For the ease of presentation,



(a) Fraction of failed links.

(b) Fraction of operational links.

Figure 2.14. Performance comparison for connected post-attack Polish system ($|V_H| = 40$).



(a) Fraction of failed links.

(b) Fraction of operational links.

Figure 2.15. Performance comparison for connected post-attack IEEE 300-bus system ($|V_H| = 20$).

we first give the complete statement of AC-Lemma 2.3.2.

Lemma 2.8.3 (AC-Lemma 2.3.2)**.** *A line $e \in F$ cannot be missed by FLD if for $Q_m = \{e\}$ and $Q_f = \emptyset$, there is a solution $\boldsymbol{z} \geq \mathbf{0}$ to*

$$[\boldsymbol{A}_D^T, \boldsymbol{A}_x^T, \boldsymbol{W}^T, \mathbf{1}]\boldsymbol{z} = \mathbf{0}, \tag{2.49a}$$

$$[\boldsymbol{g}_D^T, \boldsymbol{g}_x^T, \boldsymbol{g}_w^T, \mathbf{0}]\boldsymbol{z} < 0. \tag{2.49b}$$

Similarly, a line $e' \in E_o$ cannot be falsely detected as failed by FLD if there exists a solution $\boldsymbol{z} \geq \mathbf{0}$ to (2.13) *where $\boldsymbol{W}$ is constructed according to $Q_f = \{e'\}$ and $Q_m = \emptyset$.*

2.8.5.1 AC-VOTE

Recall from (2.23) that the counterpart of (2.5) under the AC model is

$$\tilde{\boldsymbol{D}} = [\vec{\boldsymbol{v}}']\Big(\overline{\boldsymbol{D}_f[\boldsymbol{Y}_f\vec{\boldsymbol{v}}']} + \overline{\boldsymbol{D}_t[\boldsymbol{Y}_t\vec{\boldsymbol{v}}']}\Big), \tag{2.50}$$

which indicates the "power flows" after attack. Recall from (2.22b) that $\tilde{\boldsymbol{D}}_{p,H} = \mathrm{Re}\left(\tilde{\boldsymbol{D}}_H\right)$ and $\tilde{\boldsymbol{D}}_{q,H} = \mathrm{Im}\left(\tilde{\boldsymbol{D}}_H\right)$. Then, we define $\tilde{D}_{p,U,e}$ and $\tilde{D}_{q,U,e}$ following (2.16), where $\tilde{D}$ is replaced by $\tilde{D}_p$ and $\tilde{D}_q$. Then, AC-Theorem 2.4.1-2.4.3 are given as follows:

Theorem 2.8.4 (AC-Theorem 2.4.1)**.** *Consider a hyper-node U with $E_U = \{e_1, e_2\}$ and $e_1, e_2 \in E_H \setminus \hat{F}$. If $\tilde{D}_{p,U,e_1}\tilde{D}_{p,U,e_2} < 0$, then e_1, e_2 are guaranteed to both belong to $E_H \setminus F$ if*

1. $\hat{f}_{U,g} + (\eta - 1)\min\{|\tilde{D}_{p,U,e_1}|, |\tilde{D}_{p,U,e_2}|\} < 0$, *and*

2. $\eta < 1 - \min\{\frac{\hat{f}_{U,g}+|\tilde{D}_{p,U,e_1}|}{|\tilde{D}_{p,U,e_2}|}, \frac{\hat{f}_{U,g}+|\tilde{D}_{p,U,e_2}|}{|\tilde{D}_{p,U,e_1}|}\}$.

If $\tilde{D}_{p,U,e_1}\tilde{D}_{p,U,e_2} > 0$, *then we can verify:*

1. $e_1 \in E_H \setminus F$ *if* $(1-\eta)|\tilde{D}_{p,U,e_1}| > \hat{f}_{U,g} + |\tilde{D}_{p,U,e_2}|$,

2. $e_2 \in E_H \setminus F$ *if* $(1-\eta)|\tilde{D}_{p,U,e_2}| > \hat{f}_{U,g} + |\tilde{D}_{p,U,e_1}|$.

Theorem 2.8.5 (AC-Theorem 2.4.2)**.** *Consider a hyper-node* U *with* $E_U = \{e_1, e_2\}$ *and* $e_1 \in \hat{F}, e_2 \in E_H \setminus \hat{F}$. *If* $\tilde{D}_{p,U,e_1}\tilde{D}_{p,U,e_2} > 0$, *then the states of* e_1, e_2 *are guaranteed to be correctly identified if*

1. $\hat{f}_{U,g} - \eta|\tilde{D}_{p,U,e_1}| < 0$, $\hat{f}_{U,g} + (\eta - 1)|\tilde{D}_{p,U,e_2}| < 0$, *and*

2. *either* $\eta > \frac{\hat{f}_{U,g}+|\tilde{D}_{p,U,e_2}|}{|\tilde{D}_{p,U,e_1}|}$ *or* $\eta < 1 - \frac{\hat{f}_{U,g}+|\tilde{D}_{p,U,e_1}|}{|\tilde{D}_{p,U,e_2}|}$.

If $\tilde{D}_{p,U,e_1}\tilde{D}_{p,U,e_2} < 0$, *then we can verify:*

1. $e_1 \in F$ *if* $\eta|\tilde{D}_{p,U,e_1}| > \hat{f}_{U,g} + |\tilde{D}_{p,U,e_2}|$,

2. $e_2 \in E_H \setminus F$ *if* $(1-\eta)|\tilde{D}_{p,U,e_2}| > \hat{f}_{U,g} + |\tilde{D}_{p,U,e_1}|$.

Theorem 2.8.6 (AC-Theorem 2.4.3)**.** *Consider a hyper-node* U *with* $E_U = \{e_1, e_2\}$ *and* $e_1, e_2 \in \hat{F}$. *Then, we can verify:*

1. $e_1 \in F$ *if* $\eta|\tilde{D}_{p,U,e_1}| > \hat{f}_{U,g} + |\tilde{D}_{p,U,e_2}|$,

2. $e_2 \in F$ *if* $\eta|\tilde{D}_{p,U,e_2}| > \hat{f}_{U,g} + |\tilde{D}_{p,U,e_1}|$.

We sketch the proofs below. Recall that the key of our proof for Theorem 2.4.1-2.4.3 is to find a solution for (2.13a). That is to say, the key to prove their AC variants is to find solution for (2.49). To achieve this, we first rewrite (2.49) as follows

$$[\boldsymbol{A}_{p,D}^T, \boldsymbol{A}_{q,D}^T, \boldsymbol{A}_x^T, \boldsymbol{W}^T, \mathbf{1}]\boldsymbol{z} = \mathbf{0}, \tag{2.51a}$$

$$[\boldsymbol{g}_{p,D}^T, \boldsymbol{g}_{q,D}^T, \boldsymbol{g}_x^T, \boldsymbol{g}_w^T, \mathbf{0}]\boldsymbol{z} < 0. \tag{2.51b}$$

where $\boldsymbol{A}_{p,D}$ ($\boldsymbol{A}_{q,D}$) denotes the submatrix of $\boldsymbol{A}_D$ that involves $\tilde{\boldsymbol{D}}_{p,H}$ ($\tilde{\boldsymbol{D}}_{q,H}$), and $\boldsymbol{g}_{p,D}$ ($\tilde{\boldsymbol{D}}_{q,H}$) denotes the subvector of $\boldsymbol{g}_D$ that involves active (reactive) power injections. Next, we consider a subsystem of (2.51) given below:

$$[\boldsymbol{A}_{p,D}^T, \boldsymbol{A}_x^T, \boldsymbol{W}^T, \mathbf{1}]\boldsymbol{z} = \mathbf{0}, \tag{2.52a}$$

$$[\boldsymbol{g}_{p,D}^T, \boldsymbol{g}_x^T, \boldsymbol{g}_w^T, \boldsymbol{0}]\boldsymbol{z} < 0. \tag{2.52b}$$

Suppose there exists a solution $[\boldsymbol{z}_p, \boldsymbol{z}_A, \boldsymbol{z}_x, \boldsymbol{z}_w, \boldsymbol{z}_*]$ that is feasible to (2.52), then it is easy to see that $[\boldsymbol{z}_p, \boldsymbol{0}, \boldsymbol{z}_A, \boldsymbol{z}_x, \boldsymbol{z}_w, \boldsymbol{z}_*]$ must also be feasible to (2.51). Notice that (2.52) is the same as (2.13a) with $\tilde{\boldsymbol{D}}$ replaced as $\tilde{\boldsymbol{D}}_p$. Thus, following the proof of Theorem 2.4.1-2.4.3, we have AC-Theorem 2.4.1-2.4.3.

Considering the similarity between Theorem 2.4.1-2.4.3 and their AC variants, **AC-VOTE** takes the same form as **VOTE** with $\tilde{\boldsymbol{D}}$ replaced as $\tilde{\boldsymbol{D}}_p$.

2.8.5.2 AC-VOTE-PG

It is worth noting that the only difference in $\boldsymbol{g}_D$ between (2.27b) and (2.12) is the additional part corresponding to the reactive power $\boldsymbol{q}_H$ and $\boldsymbol{q}_H'^*$ in (2.27). Thus, as shown in (2.53), we denote $g_{p,D,L,u} := -\Delta_{p,H,u}^*$ and $g_{p,D,L,-u} := -p_{H,u}'^*$, $g_{p,D,S,u} := p_{H,u}'^*$ and $g_{p,D,S,-u} := \Delta_{p,H,u}^*$. Similarly, we denote $g_{q,D,I,u} := -\Delta_{q,H,u}^*$ and $g_{q,D,I,-u} := -q_{H,u}'^*$, $g_{q,D,C,u} := q_{H,u}'^*$ and $g_{q,D,C,-u} := \Delta_{q,H,u}^*$.

$$\begin{array}{c} \\ g_{p,D,L,u} \\ g_{p,D,L,-u} \\ g_{p,D,S,-u} \\ g_{p,D,S,u} \\ g_{q,D,I,u} \\ g_{q,D,I,-u} \\ g_{q,D,C,-u} \\ g_{q,D,C,u} \end{array} \begin{array}{c} \begin{array}{cc} \boldsymbol{g}_D & \boldsymbol{A}_D \end{array} \\ \begin{bmatrix} -\boldsymbol{\Lambda}_L\boldsymbol{\Delta}_{p,H}^* & \boldsymbol{\Lambda}_L\tilde{\boldsymbol{D}}_{p,H} \\ -\boldsymbol{\Lambda}_L\boldsymbol{p}_H'^* & -\boldsymbol{\Lambda}_L\tilde{\boldsymbol{D}}_{p,H} \\ \boldsymbol{\Lambda}_S\boldsymbol{\Delta}_{p,H}^* & -\boldsymbol{\Lambda}_S\tilde{\boldsymbol{D}}_{p,H} \\ \boldsymbol{\Lambda}_S\boldsymbol{p}_H'^* & \boldsymbol{\Lambda}_S\tilde{\boldsymbol{D}}_{p,H} \\ -\boldsymbol{\Lambda}_I\boldsymbol{\Delta}_{q,H}^* & \boldsymbol{\Lambda}_I\tilde{\boldsymbol{D}}_{q,H} \\ -\boldsymbol{\Lambda}_I\boldsymbol{q}_H'^* & -\boldsymbol{\Lambda}_I\tilde{\boldsymbol{D}}_{q,H} \\ \boldsymbol{\Lambda}_C\boldsymbol{\Delta}_{q,H}^* & -\boldsymbol{\Lambda}_C\tilde{\boldsymbol{D}}_{q,H}^T \\ \boldsymbol{\Lambda}_C\boldsymbol{q}_H'^* & \boldsymbol{\Lambda}_C\tilde{\boldsymbol{D}}_{q,H} \end{bmatrix} \end{array}. \tag{2.53}$$

Then, by following the proof of Theorem 2.4.4, we have

Theorem 2.8.7 (AC-Theorem 2.4.4)**.** *Given a set E_v of lines with known states, we define $\hat{\boldsymbol{g}}_x \in \mathbb{R}^{2|E_H|}$ exactly the same as that in Theorem 2.4.4. We define $\hat{g}_{p,D,L,u}$ as follows:*

$$\hat{g}_{p,D,L,u} = \begin{cases} g_{p,D,L,u} \text{ if } u \in U_B \\ |p_u| \text{ if } u \notin U_B \end{cases} \tag{2.54}$$

and define $\hat{g}_{p,D,L,-u}$, $\hat{g}_{p,D,S,u}$ *and* $\hat{g}_{p,D,S,-u}$ *similarly. Then, we define* $\hat{g}_{q,D,I,u}$ *as follows:*

$$\hat{g}_{q,D,I,u} = \begin{cases} g_{q,D,I,u} \text{ if } u \in U_B \\ |q_u| \text{ if } u \notin U_B \end{cases} \tag{2.55}$$

and define $\hat{g}_{p,D,I,-u}$, $\hat{g}_{q,D,C,u}$ *and* $\hat{g}_{q,D,C,-u}$ *similarly. Then, a line* $l \in \hat{F}$ *is verified to have failed if there exists a solution* $\boldsymbol{z} \geq \mathbf{0}$ *to*

$$[\boldsymbol{A}_D^T, \boldsymbol{A}_x^T, \boldsymbol{w}^T, \mathbf{1}]\boldsymbol{z} = \mathbf{0}, \tag{2.56a}$$

$$[\hat{\boldsymbol{g}}_D^T, \hat{\boldsymbol{g}}_x^T, g_w, \mathbf{0}]\boldsymbol{z} < 0, \tag{2.56b}$$

where $\boldsymbol{w} \in \{0,1\}^{|E_H|}$ *is defined to be* $\boldsymbol{W}_f$ *with* $Q_f = \{l\}$, *and* $g_w := -\eta$. *Similarly, a line* $e \in E_H \setminus \hat{F}$ *is verified to be operational if* $\exists \boldsymbol{z} \geq \mathbf{0}$ *that satisfies* (2.20), *where* $\boldsymbol{w} \in \{0,1\}^{|E_H|}$ *is defined to be* $\boldsymbol{W}_m$ *with* $Q_m = \{e\}$, *and* $g_w := \eta - 1$.

Thus, **AC-VOTE-PG** takes the same form as **VOTE-PG**, where solving (2.20) in Line 4 becomes solving (2.56).

Chapter 3
Preventing Outages under Coordinated Cyber-Physical Attack with Secured PMUs

3.1 Introduction

Coordinated cyber-physical attacks (CCPA) [33] have gained a great deal of attention due to the stealthiness of such attacks and the potential for severe damage on to the smart grid. The power of CCPA is that its physical component damages the grid while its cyber component masks such damage from the control center (CC) to prolong outages and potentially enable cascades. For instance, in the Ukrainian power grid attack [23], attackers remotely switched off substations (damaging the physical system) while disrupting the control through telephonic floods and KillDisk server wiping (damaging the cyber system).

Defenses against CCPA can be broadly categorized into *detection* and *prevention*. Attack detection mechanisms aim at detecting attacks that are otherwise undetectable using traditional bad data detection (BDD) by exploiting knowledge unknown to the attacker [63]. However, the knowledge gap between the attacker and the defender may disappear due to more advanced attacks, and relying on detection alone risks severe consequences in case of misses. Therefore, in this work, we focus on preventing attacks using secured sensors.

We consider a powerful attacker with full knowledge of the pre-attack state of the grid and the locations of secured PMUs. The attacker launches an optimized CCPA where the physical attack disconnects a limited number of lines and the cyber attack falsifies the breaker status and the measurements from unsecured sensors to mask the physical attack

while misleading security constrained economic dispatch (SCED) at the CC. Such attacks can result in severe cascading failures. For example, under the setting in Section 3.5, CCPA in absence of secured PMUs can cause initial overload-induced tripping at 2, 1, and 2 lines in IEEE 30-bus, 57-bus, and 118-bus systems, respectively. Moreover, the re-distribution of power flows on the initially tripped lines may cause cascading outages. Take IEEE 118-bus system as an example. There is an attack that physically disconnects line 144 and manipulates the measurements to cause overload-induced tripping at line 109. These initial outages will trigger a cascade that eventually results in outages at 82 lines [1]. This observation highlights the importance of defending against such attacks.

While attack prevention traditionally aims at eliminating undetectable attacks by deploying secured PMUs to achieve full observability [67], this approach can require a large number of PMUs. With budget constraint, the operator may want to gradually deploy PMUs with a lower defending goal as the trade-off. To address this issue, we lower the goal to *preventing undetectable attacks from causing outages*. Specifically, we want to deploy the minimum number of secured PMUs such that the attacker will not be able to cause overload-induced line tripping due to overcurrent protection devices. The key novelty of our approach is that we allow undetectable attacks to exist but prevent them from causing any outages, hence potentially requiring fewer secured PMUs. For instance, we can prevent overload-induced tripping using 71% fewer secured PMUs compared to the requirement of full observability in IEEE 118-bus system.

3.1.1 Related Work

Attacks: False data injection (FDI) is widely adopted to launch cyber attacks in CCPA to bypass the traditional BDD [33]. A typical form of FDI is load redistribution attack [68], which together with physical attacks [17, 33, 69] that alter grid topology, aims to mislead SCED by injecting false data for economic loss or severe physical consequences such as sequential outages [17]. Bi-level optimization is widely adopted for analyzing the impact of CCPA on state deviation [70] or line flow changes [71–74]. In this work, we extend them into a stronger attacker that jointly optimizes the location of physical attacks and the attacking target.

Defenses: To eliminate the existence of FDI with minimal cost, different strategies

[1]This simplified example is based on the DC power flow model for illustrative purposes. Under the quasi-steady-state modeling assumption, it is assumed that the power grid can always reach a steady state. In practice, factors such as voltage collapse during the transient phase [64], preventive control measures [65], and other considerations [66] may lead to a different number of tripped lines than initially anticipated from such line tripping in this example.

have been studied, such as directly protecting meters [48, 74–78] or deploying secured PMUs [50, 67, 79]. Due to the connection between observability of the grid and FDI [80], solutions on achieving full observability through PMUs [81, 82] can also be leveraged to defend against FDI. Unlike the aforementioned works, our work only aims to prevent attacks from causing outages, which can significantly reduce the required number of secured PMUs.

Tri-level optimization is widely used for modeling interactions among the defender, the attacker and the operator in smart grid. To name a few, a tri-level model is proposed in [83] to find the optimal set of lines to protect from physical attacks to minimize load shedding. In [84, 85], budget-constrained equipment protection are studied. In [86], the network component hardening problem is studied in distribution networks. The work closest to ours is [74], which formulates a tri-level optimization to defend against CCPAs by securing sensor measurements, with a different objective of minimizing the number of overloaded lines under a budget constraint. Besides the different objective, [74] also differs from our work in that: (i) their physical attack is limited to a single line and not optimized, and accordingly, their defender simply minimizes the sum of overloaded lines across all attack instances (each disconnecting a single line); (ii) their defender is allowed to select individual meters to protect. In contrast, we consider physical attacks that can disconnect multiple lines at optimized locations, and our defense is via deploying secured PMUs that offer protection at the granularity of one-hop neighborhoods. These differences make our problem much more challenging, while enabling our solution to defend against stronger attacks.

3.1.2 Summary of Contributions

We summarize our contributions as follows:

1. Instead of eliminating the existence of FDI, we investigate the optimal secured PMU Placement for Outage Prevention (PPOP) problem to defend against CCPA, where we formulate a strong attacker that jointly optimizes physical attack locations and target lines. The proposed approach can potentially require fewer PMUs than approaches that eliminate FDI.

2. We propose an alternating optimization algorithm to solve PPOP by generating additional constraints from each infeasible PMU placement. Specifically, we demonstrate how to generate "No-Good" constraints and "Attack-Denial" constraints to solve PPOP optimally.

3. We develop a heuristic algorithm for PPOP to produce a possibly suboptimal solution. The complexity of the proposed heuristic is polynomial in the grid size, which makes it scalable to large networks.

4. We systematically evaluate the proposed solution on IEEE 30-bus, IEEE 57-bus, IEEE 118-bus, and IEEE 300-bus systems. The results demonstrate that the proposed solution can help to save the secured PMUs, which sheds light on the practical deployment of secured sensors to defend against CCPA.

Roadmap: We formulate our PPOP problem in Section 3.2 and demonstrate both exact algorithms and inexact heuristic to solve PPOP in Section 3.3. We evaluate the performance of PPOP in Section 3.5 and conclude the paper in Section 3.6. For the ease of reading, we put some technical details and proofs in the Section 3.7.

3.2 Problem formulation

Notations: For a matrix $\boldsymbol{A}$, we denote by $\boldsymbol{a}_i$ its i-th column and $\boldsymbol{A}_k$ its k-th row. We slightly abuse the notation $|\cdot|$ in that $|A|$ indicates the cardinality if A is a set and the element-wise absolute value if A is a vector or matrix. Logical expression $\leftrightarrow$ indicates the "if and only if" logic, while $\rightarrow$ denotes the "if then" logic. When the operators $\geq, \leq, =$ are applied to two vectors, they indicate element-wise operations. Let $\boldsymbol{a} \in \mathbb{R}^{n_a}, \boldsymbol{b} \in \mathbb{R}^{n_b}$ be two vectors, then $\boldsymbol{a} \oplus \boldsymbol{b} \in \mathbb{R}^{n_a+n_b}$ indicates the vertical concatenation of $\boldsymbol{a}$ and $\boldsymbol{b}$. Let $\lceil \boldsymbol{a} \rceil$ denote the element-wise ceiling. If $n_a = n_b = n$, then $\boldsymbol{a} \odot \boldsymbol{b} := (a_i b_i)_{i=1}^n$ denotes the Hadamard product, i.e., the element-wise product. We use $\boldsymbol{\Lambda}_{(\cdot)} \in \{0,1\}^{m \times n}$ with one nonzero element in each row to select entries from a vector such that $\boldsymbol{\Lambda}_{(\cdot)}\boldsymbol{x}$ is a subvector of $\boldsymbol{x}$.

3.2.1 Power Grid Modeling

We model the power grid as a connected undirected graph $G = (V, E)$, where E denotes the set of lines (lines) and V the set of nodes (buses). Under the DC power flow approximation, which is widely adopted for studying security issue on grid [17, 33, 67–74], each line $e = (s, t)$ is characterized by reactance $r_e = r_{st} = r_{ts}$. The grid topology can be

represented by the *admittance matrix* $\boldsymbol{B} := (B_{uv})_{u,v\in V} \in \mathbb{R}^{|V|\times|V|}$, defined as

$$B_{uv} = \begin{cases} 0 & \text{if } u \neq v, (u,v) \notin E, \\ -1/r_{uv} & \text{if } u \neq v, (u,v) \in E, \\ -\sum_{w\in V\setminus\{u\}} B_{uw} & \text{if } u = v. \end{cases} \tag{3.1}$$

Besides $\boldsymbol{B}$, the grid topology can also be described by *incidence matrix* $\boldsymbol{D} \in \{-1,0,1\}^{|V|\times|E|}$, which is defined as follows:

$$D_{ij} = \begin{cases} 1 & \text{if line } e_j \text{ comes out of node } v_i, \\ -1 & \text{if line } e_j \text{ goes into node } v_i, \\ 0 & \text{otherwise,} \end{cases} \tag{3.2}$$

where the orientation of each line is assigned arbitrarily. By defining $\boldsymbol{\Gamma} \in \mathbb{R}^{|E|\times|E|}$ as a diagonal matrix with diagonal entries $\Gamma_e = \frac{1}{r_e}$ $(e \in E)$, we have $\boldsymbol{B} = \boldsymbol{D}\boldsymbol{\Gamma}\boldsymbol{D}^T$ and $\boldsymbol{f} = \boldsymbol{\Gamma}\boldsymbol{D}^T\boldsymbol{\theta} \in \mathbb{R}^{|E|}$ where $\boldsymbol{f}$ denotes the line flows. By defining network states as phase angles $\boldsymbol{\theta} := (\theta_u)_{u\in V}$ and active powers as $\boldsymbol{p} = (p_u)_{u\in V}$, the relationship between $\boldsymbol{p}, \boldsymbol{\theta}$ and $\boldsymbol{f}$ is given as

$$\boldsymbol{p} = \boldsymbol{B}\boldsymbol{\theta} = \boldsymbol{D}\boldsymbol{f}, \tag{3.3}$$

The CC will periodically conduct state estimation, whose results will be used for SCED to re-plan the power generation [17, 68]. Formally, let $\boldsymbol{z} = [\boldsymbol{z}_N^T, \boldsymbol{z}_L^T]^T \in \mathbb{R}^m$ denote the unsecured meter measurements, where $\boldsymbol{z}_N \in \mathbb{R}^{m_N}$ denotes the power injection measurements over (a subset of) nodes and $\boldsymbol{z}_L \in \mathbb{R}^{m_L}$ denotes the power flow measurements over (a subset of) lines. Let $\boldsymbol{\Lambda}_N$ and $\boldsymbol{\Lambda}_p$ be two row selection matrices such that $\boldsymbol{z}_N = \boldsymbol{\Lambda}_N\boldsymbol{z} = \boldsymbol{\Lambda}_p\boldsymbol{p}$. Similarly, we define row selection matrices $\boldsymbol{\Lambda}_L$ and $\boldsymbol{\Lambda}_f$ such that $\boldsymbol{z}_L = \boldsymbol{\Lambda}_L\boldsymbol{z} = \boldsymbol{\Lambda}_f\boldsymbol{f}$. Then, we have

$$\boldsymbol{z} = \boldsymbol{H}\boldsymbol{\theta} + \boldsymbol{\epsilon} \text{ for } \boldsymbol{H} := \begin{bmatrix} \boldsymbol{\Lambda}_p\boldsymbol{B} \\ \boldsymbol{\Lambda}_f\boldsymbol{\Gamma}\boldsymbol{D}^T \end{bmatrix}, \tag{3.4}$$

where $\boldsymbol{H}$ is the measurement matrix based on the meter locations and the reported breaker status, and $\boldsymbol{\epsilon}$ is the measurement noise. In the rest of the paper, we assume that the measurements satisfy the conditions of [87, Theorem 5] such that $\boldsymbol{H}$ has full column rank to support unique recovery of $\boldsymbol{\theta}$ from (3.4) (before attack). If $\bar{\boldsymbol{\theta}}$ is the estimated phase angle from $\boldsymbol{z}$ and $\boldsymbol{H}$, then BDD will raise alarm if $\|\boldsymbol{z} - \boldsymbol{H}\bar{\boldsymbol{\theta}}\|$ is greater than a

predefined threshold.

Given $\boldsymbol{p}_0 := \boldsymbol{B}\bar{\boldsymbol{\theta}}$, the CC will conduct SCED to calculate new generation to meet the demand with minimal cost. Specifically, let $\boldsymbol{\Lambda}_g \in \{0,1\}^{|V_g|\times|V|}$, $\boldsymbol{\Lambda}_d \in \{0,1\}^{|V_d|\times|V|}$ be row selection matrices for generator/load buses in $\boldsymbol{p}$, where V_d and V_g denote the sets of load buses and generator buses, respectively. Denote $\hat{\boldsymbol{\theta}}$ as the decision variable where $\boldsymbol{B}\hat{\boldsymbol{\theta}}$ represents the new power injection after SCED, and $\boldsymbol{\phi} \in \mathbb{R}^{|V_g|}$ as the cost vector for power generation. Then, SCED can be formulated as follows [17]:

$$\psi_s(\boldsymbol{p}_0, \boldsymbol{D}) = \arg\min_{\hat{\boldsymbol{\theta}}} \quad \boldsymbol{\phi}^T(\boldsymbol{\Lambda}_g \boldsymbol{B}\hat{\boldsymbol{\theta}}) \tag{3.5a}$$

$$\text{s.t.} \quad \boldsymbol{\Lambda}_d \boldsymbol{B}\hat{\boldsymbol{\theta}} = \boldsymbol{\Lambda}_d \boldsymbol{p}_0, \tag{3.5b}$$

$$\boldsymbol{\Gamma}\boldsymbol{D}^T\hat{\boldsymbol{\theta}} \in [-\boldsymbol{f}_{max}, \boldsymbol{f}_{max}], \tag{3.5c}$$

$$\boldsymbol{\Lambda}_g \boldsymbol{B}\hat{\boldsymbol{\theta}} \in [\boldsymbol{p}_{g,min}, \boldsymbol{p}_{g,max}], \tag{3.5d}$$

where $\boldsymbol{f}_{max} \in \mathbb{R}^{|E|}$ indicates the normal line flow limits, $\boldsymbol{p}_{g,min}$ and $\boldsymbol{p}_{g,max}$ denote lower/upper bounds on generation, and (3.5b) indicates that demands on all load buses are satisfied.

3.2.2 Modeling Coordinated Cyber-Physical Attack (CCPA)

In this section, we formulate the attack model according to a load redistribution attack [68] that aims at causing the maximum outages, so that a defense against this attack can prevent outage under any attack under the same constraints. In the sequel, "ground truth" means the estimated value based on unmanipulated measurements, which may contain noise.

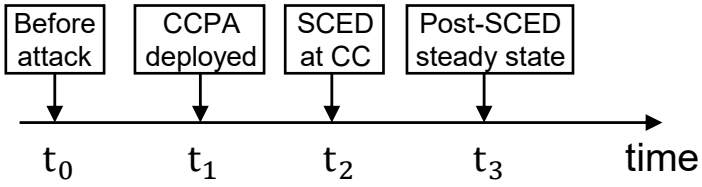


Figure 3.1. Timeline of an instance of CCPA

For ease of presentation, we summarize the timeline of the entire attack process, as shown in Fig 3.1. Specifically,

- At t_0, the attacker estimates $\boldsymbol{\theta}_0$ and $\boldsymbol{p}_0 := \tilde{\boldsymbol{B}}\boldsymbol{\theta}_0$ by eavesdropping on $\boldsymbol{z}_0$ and $\tilde{\boldsymbol{H}}$.
- At t_1, CCPA is deployed to change the ground-truth from $\boldsymbol{z}_0, \tilde{\boldsymbol{H}}, \boldsymbol{\theta}_0$ to $\boldsymbol{z}_1, \boldsymbol{H}$ and $\boldsymbol{\theta}_1$, respectively.

- At t_2, the CC receives falsified information, i.e., $\tilde{\boldsymbol{H}}$ and $\tilde{\boldsymbol{z}}_2$, which leads to $\tilde{\boldsymbol{\theta}}_2$. Then the CC will deploy a new dispatch of power generation as $\tilde{\boldsymbol{p}}_3 := \tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3$, where $\tilde{\boldsymbol{\theta}}_3$ denotes the associated predicted phase angles.

- At t_3, the new dispatch takes effect and reaches steady state, with the true phase angles $\boldsymbol{\theta}_3$ and power flows $\boldsymbol{f}_3$.

Key notations at different time instances are summarized in Table 3.1, where "—" means that the information is not available to the CC at the given time instance.

Table 3.1. Notations v.s. Timeline

time	t_0	t_1	t_2	t_3
True measurement matrix	$\tilde{\boldsymbol{H}}$	$\boldsymbol{H}$	$\boldsymbol{H}$	$\boldsymbol{H}$
Measurement matrix at CC	—	—	$\tilde{\boldsymbol{H}}$	$\tilde{\boldsymbol{H}}$
True phase angle	$\boldsymbol{\theta}_0$	$\boldsymbol{\theta}_1$	$\boldsymbol{\theta}_2 = \boldsymbol{\theta}_1$	$\boldsymbol{\theta}_3$
Phase angle at CC	—	—	$\tilde{\boldsymbol{\theta}}_2$	$\tilde{\boldsymbol{\theta}}_3$
True measurement	$\boldsymbol{z}_0$	$\boldsymbol{z}_1$	$\boldsymbol{z}_2 = \boldsymbol{z}_1$	$\boldsymbol{z}_3$
measurement at CC	—	—	$\tilde{\boldsymbol{z}}_2$	—

First, we model the influence of attacks on SCED. We define $\boldsymbol{a}_c \in \mathbb{R}^m$ to be the *cyber-attack vector*, which changes the measurements received by the CC to $\tilde{\boldsymbol{z}}_2 = \boldsymbol{z}_2 + \boldsymbol{a}_c$, and $\boldsymbol{a}_p \in \{0,1\}^{|E|}$ the *physical-attack vector*, where $a_{p,e} = 1$ indicates that line e is disconnected by the physical attack. As the physical attack changes the topology, we use $\tilde{G}$ to denote the pre-attack topology and G the post-attack topology. Accordingly, $\tilde{\boldsymbol{B}}, \tilde{\boldsymbol{D}}, \tilde{\boldsymbol{H}}$ denote the pre-attack admittance, incidence, and measurement matrices, and $\boldsymbol{B}, \boldsymbol{D}, \boldsymbol{H}$ their (true) post-attack counterparts, related by

$$\boldsymbol{B} = \tilde{\boldsymbol{B}} - \tilde{\boldsymbol{D}}\boldsymbol{\Gamma}\mathrm{diag}(\boldsymbol{a}_p)\tilde{\boldsymbol{D}}^T, \quad \boldsymbol{D} = \tilde{\boldsymbol{D}} - \tilde{\boldsymbol{D}}\mathrm{diag}(\boldsymbol{a}_p), \tag{3.6}$$

and $\boldsymbol{H} = \tilde{\boldsymbol{H}} - [(\boldsymbol{\Lambda}_p\tilde{\boldsymbol{D}}\boldsymbol{\Gamma}\mathrm{diag}(\boldsymbol{a}_p)\tilde{\boldsymbol{D}}^T)^T, (\boldsymbol{\Lambda}_f\tilde{\boldsymbol{D}}\mathrm{diag}(\boldsymbol{a}_p))^T]^T$. Falsified measurements in $\tilde{\boldsymbol{z}}_2$ and breaker status will mislead CC to an incorrect state estimation and thus falsified SCED decisions. Hence, overload-induced line tripping can happen at t_3.

To bypass BDD, the attacker has to manipulate breaker status information to mask the physical attack, misleading the CC to believe that the measurement matrix is $\tilde{\boldsymbol{H}}$ instead of $\boldsymbol{H}$. Also, measurements have to be modified into $\tilde{\boldsymbol{z}}_2$ such that BDD with $\tilde{\boldsymbol{z}}_2$ and $\tilde{\boldsymbol{H}}$ as input will not raise any alarm. Below, we will derive constraints on $\boldsymbol{a}_p$ and $\boldsymbol{a}_c$ such that the modified data can pass BDD under the assumption that the pre-attack

data can pass BDD as assumed in FDI [33]. Considering that $\tilde{\boldsymbol{z}}_2 = \boldsymbol{z}_2 + \boldsymbol{a}_c$, $\boldsymbol{a}_c$ should be constructed such that

$$\begin{aligned}\|\tilde{\boldsymbol{z}}_2 - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{\theta}}_2\| &= \|\boldsymbol{z}_0 - \tilde{\boldsymbol{H}}\boldsymbol{\theta}_0 + \boldsymbol{z}_2 + \boldsymbol{a}_c - \boldsymbol{z}_0 + \tilde{\boldsymbol{H}}\boldsymbol{\theta}_0 - \tilde{\boldsymbol{H}}\tilde{\boldsymbol{\theta}}_2\| \\ &= \|\boldsymbol{z}_0 - \tilde{\boldsymbol{H}}\boldsymbol{\theta}_0\|, \quad \text{(pre-attack residual)} \end{aligned} \tag{3.7}$$

which leads to the following construction of $\boldsymbol{a}_c$:

$$\boldsymbol{a}_c = \boldsymbol{z}_0 - \boldsymbol{z}_2 + \tilde{\boldsymbol{H}}(\tilde{\boldsymbol{\theta}}_2 - \boldsymbol{\theta}_0) \tag{3.8}$$

$$= \tilde{\boldsymbol{H}}\boldsymbol{\theta}_0 + \epsilon_0 - (\boldsymbol{H}\boldsymbol{\theta}_2 + \epsilon_0) + \tilde{\boldsymbol{H}}(\tilde{\boldsymbol{\theta}}_2 - \boldsymbol{\theta}_0) \tag{3.9}$$

$$= \begin{bmatrix} \boldsymbol{\Lambda}_p\tilde{\boldsymbol{B}} \\ \boldsymbol{\Lambda}_f\boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T \end{bmatrix}\tilde{\boldsymbol{\theta}}_2 - \begin{bmatrix} \boldsymbol{\Lambda}_p\boldsymbol{B} \\ \boldsymbol{\Lambda}_f\boldsymbol{\Gamma}\boldsymbol{D}^T \end{bmatrix}\boldsymbol{\theta}_2. \tag{3.10}$$

Besides (3.8), there may be additional constraints on $\boldsymbol{a}_c$ to avoid causing suspicion. Specifically, following [68], we assume that all the power injections at generator buses are measured and not subject to attacks, i.e.,

$$\boldsymbol{\Lambda}_g\tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2 = \boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_2 = \boldsymbol{\Lambda}_g\boldsymbol{B}\boldsymbol{\theta}_2 = \boldsymbol{\Lambda}_g\boldsymbol{D}\boldsymbol{f}_2 = \boldsymbol{\Lambda}_g\boldsymbol{p}_0, \tag{3.11}$$

recalling that $\boldsymbol{\Lambda}_g$ is the row selection matrix corresponding to generator buses. Moreover, by representing the maximum normal load fluctuation through $\alpha \geq 0$, the magnitude of falsification at load buses needs to be constrained due to load forecasting [17, 68], which can be modeled by [2]

$$-\alpha|\boldsymbol{p}_0| \leq \tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_2 - \boldsymbol{p}_0 \leq \alpha|\boldsymbol{p}_0|. \tag{3.12}$$

Following the convention in [68, 72], the attack is constrained by a predefined constant ξ_p denoting the maximum number of attacked lines and another constant ξ_c denoting the maximum number of manipulated measurements, i.e.,

$$\|\boldsymbol{a}_p\|_0 \leq \xi_p, \quad \|\boldsymbol{a}_c\|_0 \leq \xi_c. \tag{3.13}$$

In addition, we constrain $\boldsymbol{a}_p$ so that the graph after physical attack remains connected,

[2] In contrast to [88] that only imposes the magnitude constraint on measured buses, constraint (3.12) is imposed on all buses (although subsumed by (3.11) for generator buses). This is because under the assumption of full-rank measurement matrix (Section 3.2.1), the CC can recover all the phase angles and hence the power injections at all the buses, and thus the attacker needs to avoid causing too much deviation in the power injections at all the buses.

which is needed for stealth of the attack according to [17, 70]. Specifically, defining $\boldsymbol{f}_{con} \in \mathbb{R}^{|E|}$ as a pseudo flow and u_0 as the reference node, we can guarantee network connectivity at t_2 by ensuring

$$\tilde{\boldsymbol{D}}_u \boldsymbol{f}_{con} = \begin{cases} |V| - 1, & \text{if } u = u_0, \\ -1, & \text{if } u \in V \setminus \{u_0\}, \end{cases} \tag{3.14a}$$

$$-|V| \cdot (1 - a_{p,e}) \leq f_{con,e} \leq |V| \cdot (1 - a_{p,e}). \tag{3.14b}$$

With lines oriented as in $\tilde{\boldsymbol{D}}$, (3.14a) (flow conservation constraint) and (3.14b) (line capacity constraint) ensure the existence of a unit pseudo flow from u_0 to every other node in the post-attack grid and hence the connectivity of the post-attack grid, where $f_{con,e} > 0$ if the flow on e is in the same direction of the line and $f_{con,e} < 0$ otherwise.

As shown in [17], attacks that mislead SCED can cause initial outage at t_3, which can cause cascading outages at other lines since significantly overloaded lines will be automatically tripped by protective devices and the associated power flow will be re-distributed. Specifically, let $\boldsymbol{f}_{max} \in \mathbb{R}^{|E|}$ be the power flow limits imposed in SCED and $\boldsymbol{\gamma} := (\gamma_e)_{e \in E}$ be the threshold for overload-induced tripping, i.e., e will be tripped by protection devices (i.e., having an outage) if

$$|f_e| > \gamma_e f_{max,e}. \tag{3.15}$$

3.2.3 Modeling the Protection Effect of Secured PMUs

Phasor Measurement Units (PMUs) exhibit enhanced resilience against false data injection attacks [50], attributed to their advanced features. Firstly, PMU measurements are marked with high-precision timestamps, synchronized via GPS, complicating the execution of covert attacks. Secondly, PMUs sample the grid's state at a rate of 60 to 120 frames per second [89], a frequency significantly higher than that of traditional Remote Terminal Units (RTUs). This high sampling rate generates time series data that characterizes the system's dynamics, thereby facilitating the use of advanced detection algorithms [90].

Let $\boldsymbol{\beta} \in \{0, 1\}^{|V|}$ be the indicator vector for PMU placement such that $\beta_u = 1$ if and only if a secured PMU is installed at node u. We define $\Omega(\boldsymbol{\beta}) := \{u | \beta_u > 0\}$ and the inverse process $\boldsymbol{\beta}(\Omega) : \beta_u = 1$ if $u \in \Omega$ and $\beta_u = 0$ otherwise. Let $\mathcal{N}_u$ be the node set containing neighbors of node u (including u) and E_u be the line set composed of lines incident on u. According to [50], by measuring both voltage and current phasor, a PMU on node u can guarantee the correctness of phase angles in $\mathcal{N}_u$ and protect lines in

E_u from both cyber and physical attacks. Formally, we define $\boldsymbol{x}_N \in \{0,1\}^{|V|}$ such that $(x_{N,u} = 1) \leftrightarrow (\exists v \in \mathcal{N}_u$ such that $\beta_v = 1)$, which can be modeled as

$$\boldsymbol{\Delta}^{-1}\underline{\boldsymbol{A}}\boldsymbol{\beta} \leq \boldsymbol{x}_N \leq \boldsymbol{\Delta}^{-1}\underline{\boldsymbol{A}}\boldsymbol{\beta} + \frac{\|\boldsymbol{\Delta}\|_\infty - 1}{\|\boldsymbol{\Delta}\|_\infty}, \tag{3.16}$$

where $\boldsymbol{\Delta} \in \mathbb{Z}^{|V|\times|V|}$ is a diagonal matrix with $\Delta_{uu} = |\mathcal{N}_u|$, while $\underline{\boldsymbol{A}} := \boldsymbol{A} + \boldsymbol{I}$ is the adjacency matrix of the grid with added self-loops at all nodes. Similarly, we define ζ to be any constant within $[0.5, 1)$ and $\boldsymbol{x}_L \in \{0,1\}^{|E|}$ satisfying $(x_{L,e} = 1) \leftrightarrow (\exists v$ with $e \in E_v$ and $\beta_v = 1)$, which can be linearlized as

$$0.5|\boldsymbol{D}|^T\boldsymbol{\beta} \leq \boldsymbol{x}_L \leq 0.5|\boldsymbol{D}|^T\boldsymbol{\beta} + \zeta. \tag{3.17}$$

We assume that the PMU locations are known to the attacker, thus the cyber attack is constrained as follows:

$$x_{N,u} = 1 \rightarrow \tilde{\boldsymbol{\theta}}_{2,u} = \boldsymbol{\theta}_{2,u}, \forall u \in V, \tag{3.18a}$$

$$x_{L,e} = 1 \rightarrow a_{p,e} = 0, \quad \forall e \in E. \tag{3.18b}$$

Note that (3.16)-(3.18) implicitly protect the power flow measurements on lines incident to a PMU. To see this, suppose that $e = (s,t)$ and $\beta_s = 1$. Then we must have $x_{N,s} = x_{N,t} = x_{L,e} = 1$ due to (3.16)-(3.17). By (3.18), it is guaranteed that $\tilde{z}_{2,e} := (\tilde{\theta}_{2,s} - \tilde{\theta}_{2,t})/r_{st} = (\theta_{2,s} - \theta_{2,t})/r_{st} =: z_{2,e}$.

3.2.4 Optimal PMU Placement Problem

Our main problem, named *PMU Placement for Outage Prevention (PPOP)*, aims at placing the minimum number of secured PMUs so that no undetectable CCPA can cause overload-induced tripping. To achieve this, we model the problem as a tri-level optimization problem (an overview of PPOP is given in Fig. 4 in Appendix 3.7.1).

The *middle-level* optimization is the attacker's problem, which aims to maximize the number of overloaded lines without being detected. Instead of using $\boldsymbol{a}_c$ as decision variable, we propose to formulate over $\tilde{\boldsymbol{f}}_i, \boldsymbol{f}_i$ and $\tilde{\boldsymbol{\theta}}_i, \boldsymbol{\theta}_i$ where $i \in \{2,3\}$. In the rest of the paper, we will apply big-M modeling technique that introduces sufficiently large constants denoted as $M_{(\cdot)}$ for linearization. The calculation of $M_{(\cdot)}$ is given in Appendix 3.7.2.

Specifically, the constraints on $\boldsymbol{\theta}_2$ and $\boldsymbol{f}_2$ are:

$$-M_{2,a,e}\left(\mathbf{1}-\boldsymbol{a}_p\right) \leq \boldsymbol{f}_2 \leq M_{2,a,e}\left(\mathbf{1}-\boldsymbol{a}_p\right), \tag{3.19a}$$

$$\tilde{\boldsymbol{D}}\boldsymbol{f}_2 = \boldsymbol{p}_0, \tag{3.19b}$$

$$-M_{2,f}\boldsymbol{a}_p \leq \boldsymbol{\Gamma}\boldsymbol{D}\boldsymbol{\theta}_2 - \boldsymbol{f}_2 \leq M_{2,f}\boldsymbol{a}_p. \tag{3.19c}$$

The constraints (3.19a) and (3.19b) guarantee the consistency between $\boldsymbol{f}_2$ and $\boldsymbol{p}_0$ given $\boldsymbol{a}_p$, where $a_{p,e}=1$ will force $f_{2,e}=0$. The role of (3.19c) is to force the consistency between $\boldsymbol{f}_2$ and $\boldsymbol{\theta}_2$ for all e with $a_{p,e}=0$, which is necessary for the uniqueness of $\boldsymbol{f}_2$. Similarly, we can transform (3.7)-(3.13) into constraints over $\tilde{\boldsymbol{f}}_2$, $\tilde{\boldsymbol{\theta}}_2$, and $\boldsymbol{a}_p$, which are

$$-\boldsymbol{f}_{\max} \leq \tilde{\boldsymbol{f}}_2 \leq \boldsymbol{f}_{\max}, \tag{3.20a}$$

$$\boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T\tilde{\boldsymbol{\theta}}_2 - \tilde{\boldsymbol{f}}_2 = 0, \tag{3.20b}$$

$$\tilde{\theta}_{2,u} - \theta_{2,u} \leq M_{2,\theta}\cdot(1-\boldsymbol{x}_{N,u}), \tag{3.20c}$$

$$\tilde{\theta}_{2,u} - \theta_{2,u} \geq -M_{2,\theta}\cdot(1-\boldsymbol{x}_{N,u}), \tag{3.20d}$$

$$-\alpha|\boldsymbol{p}_0| \leq \tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2 - \boldsymbol{p}_0 \leq \alpha|\boldsymbol{p}_0|, \tag{3.20e}$$

$$\boldsymbol{\Lambda}_g\tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2 = \boldsymbol{\Lambda}_g\boldsymbol{p}_0, \tag{3.20f}$$

$$\|\boldsymbol{\Lambda}_f\left(\tilde{\boldsymbol{f}}_2 - \boldsymbol{f}_2\right)\|_0 + \|\boldsymbol{\Lambda}_p\left(\tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2 - \boldsymbol{p}_0\right)\|_0 \leq \xi_c, \tag{3.20g}$$

$$\|\boldsymbol{a}_p\|_0 \leq \xi_p, \tag{3.20h}$$

where (3.20a)-(3.20b) guarantee the validity of $\tilde{\boldsymbol{f}}_2$ as in (3.19a)-(3.19c), (3.20c)-(3.20d) linearize (3.18a) ($M_{2,\theta}$ defined in Appendix 3.7.2), while (3.20e), (3.20f), and (3.20g)–(3.20h) correspond to (3.12), (3.11), and (3.13), respectively. It is worth noting that there exists an $\boldsymbol{a}_c$ in the form of (3.10) for any $\tilde{\boldsymbol{f}}_2$ and $\tilde{\boldsymbol{\theta}}_2$ satisfying (3.20) due to the relationship between $\tilde{\boldsymbol{f}}_2$, $\tilde{\boldsymbol{\theta}}_2$ and $\boldsymbol{a}_c$ shown in (3.10) and (3.20b). Moreover, the constraints on $\boldsymbol{\theta}_3, \tilde{\boldsymbol{\theta}}_3$, and $\boldsymbol{f}_3$ are

$$\boldsymbol{p}_{g,min} \leq \boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3 \leq \boldsymbol{p}_{g,max} \tag{3.21a}$$

$$-\boldsymbol{f}_{\max} \leq \boldsymbol{\Gamma}\boldsymbol{D}^T\tilde{\boldsymbol{\theta}}_3 \leq \boldsymbol{f}_{\max}, \tag{3.21b}$$

$$\boldsymbol{\Lambda}_d\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3 = \boldsymbol{\Lambda}_d\tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2 \tag{3.21c}$$

$$-M_{3,a}(\mathbf{1}-\boldsymbol{a}_p) \leq \boldsymbol{f}_3 \leq M_{3,a}(\mathbf{1}-\boldsymbol{a}_p), \tag{3.21d}$$

$$\boldsymbol{\Lambda}_d\tilde{\boldsymbol{D}}\boldsymbol{f}_3 = \boldsymbol{\Lambda}_d\boldsymbol{p}_0, \quad \boldsymbol{\Lambda}_g\tilde{\boldsymbol{D}}\boldsymbol{f}_3 = \boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3, \tag{3.21e}$$

$$-M_{3,f}\boldsymbol{a}_p \leq \boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T\boldsymbol{\theta}_3 - \boldsymbol{f}_3 \leq M_{3,f}\boldsymbol{a}_p, \tag{3.21f}$$

where (3.21a)-(3.21c) describe the feasible region of $\tilde{\boldsymbol{\theta}}_3$ under false data injection, and (3.21d)–(3.21f) are used to enforce the power flow equation (3.3) at t_3, where $\boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3$ is the post-SCED generation predicted by the attacker. While a straightforward formulation of the power flow equation should be

$$\boldsymbol{\Gamma}\boldsymbol{D}^T\boldsymbol{\theta}_3 = \boldsymbol{f}_3,\ \boldsymbol{\Lambda}_d\boldsymbol{D}\boldsymbol{f}_3 = \boldsymbol{\Lambda}_d\boldsymbol{p}_0,\ \boldsymbol{\Lambda}_g\boldsymbol{D}\boldsymbol{f}_3 = \boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3, \tag{3.22}$$

such a formulation will introduce bilinear terms $\boldsymbol{D}^T\boldsymbol{\theta}_3$ and $\boldsymbol{D}\boldsymbol{f}_3$, as the post-attack incidence matrix $\boldsymbol{D}$ is a function of the physical-attack vector $\boldsymbol{a}_p$ that is also a decision variable for the attacker. To avoid the bilinear terms, we use (3.21d) to force $f_{3,e} = 0$ when $a_{p,e} = 1$ (line e is disconnected), and (3.21f) to force $\Gamma_e\tilde{\boldsymbol{d}}_e^T\boldsymbol{\theta}_3 = \Gamma_e\boldsymbol{d}_e^T\boldsymbol{\theta}_3 = f_{3,e}$ when $a_{p,e} = 0$. Moreover, under (3.21d), we observe that $\boldsymbol{D}\boldsymbol{f}_3 = \sum_{e\in E}\boldsymbol{d}_e f_{3,e} = \sum_{e\in E}\tilde{\boldsymbol{d}}_e f_{3,e} = \tilde{\boldsymbol{D}}\boldsymbol{f}_3$, as $\boldsymbol{d}_e = \tilde{\boldsymbol{d}}_e$ if $a_{p,e} = 0$ and $\boldsymbol{d}_e f_{3,e} = \tilde{\boldsymbol{d}}_e f_{3,e} = \boldsymbol{0}$ if $a_{p,e} = 1$, which explains (3.21e).

Thus, the attacker's problem, which defines the optimal attack strategy, can be formulated as:

$$\begin{aligned}
\psi_a(\boldsymbol{\beta}) := \max\quad & \|\boldsymbol{\pi}\|_0 && \text{(3.23a)}\\
\text{s.t.}\quad & (3.14), (3.16) - (3.21), && \text{(3.23b)}\\
& \theta_{2,u_0} = \theta_{3,u_0} = \tilde{\theta}_{2,u_0} = \tilde{\theta}_{3,u_0} = 0, && \text{(3.23c)}\\
& \tilde{\boldsymbol{\theta}}_3 = \psi_s(\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_2, \tilde{\boldsymbol{D}}), && \text{(3.23d)}\\
& \frac{|f_{3,e}|}{f_{\max,e}} > \gamma_e \leftrightarrow \pi_e = 1, \forall e \in E, && \text{(3.23e)}
\end{aligned}$$

where $\boldsymbol{y}_c := \tilde{\boldsymbol{\theta}}_2 \oplus \tilde{\boldsymbol{\theta}}_3 \oplus \boldsymbol{\theta}_2 \oplus \boldsymbol{\theta}_3 \oplus \boldsymbol{f}_2 \oplus \boldsymbol{f}_3 \oplus \tilde{\boldsymbol{f}}_2 \oplus \boldsymbol{f}_{con}$ and $\boldsymbol{y}_b := \boldsymbol{\pi} \oplus \boldsymbol{a}_p \oplus \boldsymbol{x}_N \oplus \boldsymbol{x}_L$ are continuous and binary decision variables, respectively. Here, $\pi_e = 1$ if and only if line e is overloaded to be tripped, which is ensured by (3.23e). Thus, the objective is to maximize the number of overload-induced tripped lines due to the attack-induced load redistribution. The constraints (3.23c) fixes the phase angle at the reference node, denoted as node u_0. The constraint (3.23d) incorporates the *lower-level* optimization of SCED (3.5) by specifying the post-SCED generation, determined by $\tilde{\boldsymbol{\theta}}_3$.

We formulate the *upper-level* PMU placement problem as

$$\begin{aligned}
\min\quad & \|\boldsymbol{\beta}\|_0 && \text{(3.24a)}\\
\text{s.t.}\quad & \psi_a(\boldsymbol{\beta}) = 0 && \text{(3.24b)}
\end{aligned}$$

where the decision variable is $\boldsymbol{\beta} \in \{0,\ 1\}^{|V|}$, and $\psi_a(\boldsymbol{x})$ defined in (3.23) denotes the

maximum number of lines that will be tripped according to (3.15) at t_3. In the sequel, we call $(\boldsymbol{a}_p, \boldsymbol{a}_c, e)$ an *attack tuple*, which is called "successful" under PMU placement $\boldsymbol{\beta}$ if there exists a feasible solution to (3.23) with physical attack $\boldsymbol{a}_p$ and cyber attack $\boldsymbol{a}_c$ such that $\pi_e = 1$. Moreover, we call $(\boldsymbol{a}_p, e)$ a successful *attack pair* under $\boldsymbol{\beta}$ if it can form a successful attack tuple under $\boldsymbol{\beta}$.

Remark 1: While the above formulation treats the load profile $\boldsymbol{p}_0$ as a constant, it can be easily extended to handle the fluctuations in loads. This can be modeled by treating $\boldsymbol{p}_0$ as a decision variable in the attacker's optimization, constrained by the expected range of fluctuation, e.g., $\boldsymbol{p}_0 \in [\underline{\kappa}\boldsymbol{p}^{(0)}, \overline{\kappa}\boldsymbol{p}^{(0)}]$, or the union of ranges around multiple operating points:

$$\boldsymbol{p}_0 \in \bigcup_{i=1}^{i_0} \{\underline{\kappa}_i \boldsymbol{p}^{(i)} \leq \boldsymbol{p} \leq \overline{\kappa}_i \boldsymbol{p}^{(i)}\}. \tag{3.25}$$

This enlarges the solution space for the attacker, which changes the meaning of $\psi_a(\boldsymbol{\beta})$ to the *maximum number of tripped lines under the worst load profile and the worst attack under this load profile*. Clearly, a PMU placement that avoids overload-induced tripping in this worst scenario can avoid overload-induced tripping in any scenario encountered during operation, as long as the load profile stays within the predicted range.

Remark 2: In practice, PMUs are often deployed in stages. Thus, it may be desirable that a temporary PMU placement designed to prevent outages can be augmented into an optimal PMU placement $\boldsymbol{\beta}^{opt}$ in the long run (e.g., a minimum placement that provides full observability). This can be modeled by adding a constraint in (3.24) that requires $\boldsymbol{\beta} \leq \boldsymbol{\beta}^{opt}$.

3.3 Solving PPOP

The PPOP problem (3.23)-(3.24) is a tri-level non-linear mixed integer problem, which is notoriously hard [91]. In this section, we first formally prove that the problem is NP-hard, and then demonstrate how to transform it into a *bi-level mixed-integer linear programming (MILP)* problem. Next, we propose an alternating optimization framework based on constraint generation to solve the problem optimally. Finally, to accelerate the computation, we develop a polynomial-time heuristic.

3.3.1 Hardness and Conversion to Bi-Level MILP

Although multi-level non-linear mixed integer programming is generally hard, PPOP is only a special case and hence needs to be analyzed separately. Nevertheless, we show that PPOP is NP-hard (See proof in Section. 3.7.5).

Theorem 3.3.1. *The PPOP problem* (3.24) *is NP-hard.*

The attacker's problem (3.23) can be linearized into a MILP (see details in Section 3.7.1), which implies that PPOP can be converted into a bi-level MILP.

Instead of solving the PPOP problem (3.24), we will focus on a variant by considering a stronger attacker that relaxes the constraints in (3.23). Specifically, we will relax (3.23d) by omitting the optimality requirements in (3.5), i.e., (3.23d) is replaced by (3.5b)-(3.5d). Such replacement results in a relaxed attacker's problem $\underline{\psi}_a(\boldsymbol{\beta})$, whose feasible region is described by (3.23b), (3.23c), (3.23e) and (3.5b)-(3.5d). The relationship between $\underline{\psi}_a(\boldsymbol{\beta})$ and $\psi_a(\boldsymbol{\beta})$ is summarized as follows:

Lemma 3.3.1. *Every feasible solution to* $\psi_a(\boldsymbol{\beta})$ *is also feasible to* $\underline{\psi}_a(\boldsymbol{\beta})$.

Proof. In terms of the feasible region, the only difference between $\psi_a(\boldsymbol{\beta})$ and $\underline{\psi}_a(\boldsymbol{\beta})$ is the constraint on $\tilde{\boldsymbol{\theta}}_3$. It is easy to see that $\underline{\psi}_a(\boldsymbol{\beta})$ has less constraints on $\tilde{\boldsymbol{\theta}}_3$ than $\psi_a(\boldsymbol{\beta})$, which completes the proof. □

Then, we denote PPOP-r as the PMU placement problem that replace (3.24b) as

$$\underline{\psi}_a(\boldsymbol{\beta}) = 0. \tag{3.26}$$

The relationship between PPOP and PPOP-r is summarized as follows:

Theorem 3.3.2. *Every feasible PMU placement (i.e.,* $\boldsymbol{\beta}$*) to PPOP-r is also feasible to PPOP.*

Proof. The direct implication of Lemma. 3.3.1 is that if there exists $(\boldsymbol{y}_c, \boldsymbol{y}_b)$ such that $\psi_a(\boldsymbol{\beta}) \geq 1$, we must have $\underline{\psi}_a(\boldsymbol{\beta})$ with $(\boldsymbol{y}_c, \boldsymbol{y}_b)$. In other words, if $\underline{\psi}_a(\boldsymbol{\beta}) = 0$ for the given $\boldsymbol{\beta}$, we must have $\psi_a(\boldsymbol{\beta}) = 0$, which completes the proof. □

In the sequel, we will replace PPOP as PPOP-r and use these two terms interchangeably. It is reasonable to consider PPOP-r instead of PPOP since the cost in SCED (i.e., ϕ) may change over time. Therefore, a PMU placement should be be robust under all possible costs in SCED.

3.3.2 An Alternating Optimization Framework

Algorithm 4: Alternating Optimization

1 **Initialization:** $k = 1$, $\hat{\boldsymbol{\beta}}^{(k)} = \mathbf{0}$;
2 **while** *True* **do**
3 Solve (3.23) under $\hat{\boldsymbol{\beta}}^{(k)}$ to obtain $\psi_a(\hat{\boldsymbol{\beta}}^{(k)})$;
4 **if** $\psi_a(\hat{\boldsymbol{\beta}}^{(k)}) > 0$ **then**
5 Add constraints to (3.24);
6 $k \leftarrow k + 1$, obtain $\hat{\boldsymbol{\beta}}^{(k)}$ by solving (3.24), with (3.24b) replaced by the generated constraints
7 **else** break ;
8 Return $\hat{\boldsymbol{\beta}}^{(k)}$, indicators of the selected PMU placement;

As a bi-level MILP, PPOP is still difficult to solve due to the integer variables in (3.23) and (3.24). Since one of the fundamental challenges in solving bi-level MILPs is the lack of explicit description of the upper-level optimization's feasible region, we propose an alternating optimization framework shown in Alg. 4 to solve PPOP by gradually approximating the feasible region of the upper-level optimization through constraint generation. In Sections 3.3.3–3.3.4, we will give two concrete constraint generation methods for Line 5 of Alg. 4 based on the results of (3.23).

In the sequel, we assume that solving (3.23) returns a successful attack tuple $(\boldsymbol{a}_p^{(k)}, \boldsymbol{a}_c^{(k)}, e^{(k)})$ if $\psi_a(\hat{\boldsymbol{\beta}}^{(k)}) > 0$.

3.3.3 Alternating Optimization with No-Good Constraints (AONG)

In this section, we give the first specific algorithm under the framework of Alg. 4, in which the added constraints in Line 5 are motivated by the following observation:

Lemma 3.3.2. *Given $\hat{\boldsymbol{\beta}}$ and $\Omega(\hat{\boldsymbol{\beta}}) := \{u \in V : \hat{\beta}_u > 0\}$, if there exists a successful attack tuple $(\boldsymbol{a}_p, \boldsymbol{a}_c, e)$, then for all $\boldsymbol{\beta}$ with $\Omega(\boldsymbol{\beta}) \subseteq \Omega(\hat{\boldsymbol{\beta}})$, there exists a successful attack tuple.*

Proof. For any $\boldsymbol{\beta}$ with $\Omega(\boldsymbol{\beta}) \subseteq \Omega(\hat{\boldsymbol{\beta}})$, $(\boldsymbol{a}_p, \boldsymbol{a}_c, e)$ remains a successful attack tuple. □

The above observation indicates that at least one PMU must be placed in $\Omega(\hat{\boldsymbol{\beta}})^c := V \setminus \Omega(\hat{\boldsymbol{\beta}})$. Therefore, the optimal $\boldsymbol{\beta}$ can be obtained in an iterative manner: during each iteration, we use the PMU placement $\hat{\boldsymbol{\beta}}$ from the previous iteration (initially, $\hat{\boldsymbol{\beta}} = \mathbf{0}$) to solve (3.23) for $\psi_a(\hat{\boldsymbol{\beta}})$. If $\psi_a(\hat{\boldsymbol{\beta}}) = 0$, $\hat{\boldsymbol{\beta}}$ is the final solution; otherwise, we will add the

following "No-Good" constraint: $\sum_{i:\hat{\beta}_i=0} \beta_i \geq 1$ to (3.24) for the next iteration to rule out the infeasible solution $\hat{\boldsymbol{\beta}}$.

However, the above procedure will converge very slowly as $|\Omega(\hat{\boldsymbol{\beta}})^c|$ is usually large. To speed up convergence, we augment each discovered infeasible solution $\hat{\boldsymbol{\beta}}$ into a maximal infeasible solution $\hat{\boldsymbol{\beta}}'$ to narrow down candidate solutions. This can be achieved by solving the following problem:

$$\max \quad \|\hat{\boldsymbol{\beta}}'\|_0 \tag{3.27a}$$

$$\text{s.t.} \quad \psi_a(\hat{\boldsymbol{\beta}}') \geq 1, \tag{3.27b}$$

$$\hat{\beta}'_u = 1, \forall u \in V \text{ with } \hat{\beta}_u = 1, \tag{3.27c}$$

which has the same decision variables as (3.23) and the additional $\hat{\boldsymbol{\beta}}'$. Algorithm AONG adds the following "No-Good" constraint in Line 5 of Alg. 4:

$$\sum_{i:\hat{\beta}'_i=0} \beta_i \geq 1. \tag{3.28}$$

AONG solves PPOP optimally, as proved in Appendix 3.7.5.

Theorem 3.3.3. *AONG converges in finite time to an optimal solution to* (3.24).

Given the MILP formulation of (3.23) in Appendix 3.7.1, it is easy to write (3.27) as a MILP and solve it by existing MILP solvers. It is worth noting that solving (3.27) suboptimally does not affect the optimality of AONG. Thus, we can also apply heuristic algorithms (e.g., LP relaxation with rounding).

3.3.4 Alternating Optimization with Double Constraints (AODC)

Building on AONG, we develop an additional constraint as a complement of (3.28) to accelerate convergence, in the special case where $\xi_c = \infty$ and $\psi_s(\boldsymbol{p}, \boldsymbol{D})$ returns the set of $\boldsymbol{\theta}$'s satisfying (3.5b)-(3.5d), i.e., it returns the feasible region of SCED rather than a single solution. Such a special case is worth consideration because (i) $\xi_c = \infty$ represents the strongest cyber attack, and (ii) relaxing the optimality requirement in (3.23d) means that the attacker is allowed to pick a solution for SCED within its feasible region, both making the attack stronger and hence the resulting PMU placement more robust in preventing outages.

Below we will first introduce the new constraints, called "Attack-Denial" constraints, and then give the AODC algorithm, in which both "No-Good" constraints and "Attack-

Denial" constraints are added in Line 5 of Alg. 4. The new constraints are motivated by the following observations about AONG:

1. *Cold start.* The efficiency of (3.28) can be characterized by the number of infeasible $\boldsymbol{\beta}$'s that are cut out. Let $\{\hat{\boldsymbol{\beta}}^{(k)}\}_{k=1}^{K}$ be the PMU placements obtained in the first K iterations of Alg. 4 and $\{\hat{\boldsymbol{\beta}}'^{(k)}\}_{k=1}^{K}$ the corresponding augmented placements obtained from (3.27). Then, the number of feasible $\boldsymbol{\beta}$'s for the next iteration is at least

$$\left(2^{|\bigcap_{k=1}^{K}\Omega(\hat{\boldsymbol{\beta}}'^{(k)})^c|}-1\right)\cdot 2^{|V|-|\bigcap_{k=1}^{K}\Omega(\hat{\boldsymbol{\beta}}'^{(k)})^c|} \tag{3.29}$$

   if $\bigcap_{k=1}^{K}\Omega(\hat{\boldsymbol{\beta}}'^{(k)})^c \neq \emptyset$, as placing at least one PMU in $\bigcap_{k=1}^{K}\Omega(\hat{\boldsymbol{\beta}}'^{(k)})^c$ will satisfy (3.28) for every placement in $\{\hat{\boldsymbol{\beta}}'^{(k)}\}_{k=1}^{K}$. This implies that the number of $\boldsymbol{\beta}$'s that are cut out is at most $2^{|V|-|\bigcap_{k=1}^{K}\Omega(\hat{\boldsymbol{\beta}}'^{(k)})^c|}$. Therefore, the first K "No-Good" constraints (3.28) added in Alg. 4 will be inefficient if $|\bigcap_{k=1}^{K}\Omega(\hat{\boldsymbol{\beta}}'^{(k)})^c|$ is large. We observe that $|\bigcap_{k=1}^{K}\Omega(\hat{\boldsymbol{\beta}}'^{(k)})^c|$ is large at the beginning of Alg. 4 and decreases quickly as $\|\hat{\boldsymbol{\beta}}^{(k)}\|_0$ increases.

2. *Repeated successful attacks.* Another cause of inefficiency is that for many PMU placements enumerated by AONG, there exist successful attacks based on the same attack pair $(\boldsymbol{a}_p, e)$, indicating that new constraints are needed to better defend against identified attacks.

These observations motivate us to generate constraints that can invalidate the identified attack pairs.

The above observations motivate the following idea of "Attack-Denial" constraints: *given a successful attack pair* $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ *under* $\boldsymbol{\beta}^{(k)}$*, the added constraints should guarantee that any PMU placement satisfying the constraints can prevent attacks that fail lines according to* $\boldsymbol{a}_p^{(k)}$ *from causing overload-induced tripping at line* $e^{(k)}$*.* We focus on $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ instead of $(\boldsymbol{a}_p^{(k)}, \boldsymbol{a}_c^{(k)}, e^{(k)})$ due to the following observations:

1. The number of $(\boldsymbol{a}_p^{(k)}, \boldsymbol{a}_c^{(k)}, e^{(k)})$'s is infinite since $\boldsymbol{a}_c^{(k)}$ is continuous, but the number of $(\boldsymbol{a}_p^{(k)}, e^{(k)})$'s is finite.

2. Given $\boldsymbol{x}_N$ and $(\boldsymbol{a}_p^{(k)}, e^{(k)})$, (3.23b)-(3.23e) reduce to a linear system with only the continuous variables contained in $\boldsymbol{y}_c$ under the assumptions that $\xi_c = \infty$ and $\psi_s(\boldsymbol{p}, \boldsymbol{D})$ returns the set of $\boldsymbol{\theta}$'s satisfying (3.5b)-(3.5d). The linear system can be

summarized as

$$\boldsymbol{F}_1^{(k)}\boldsymbol{y}_c = \boldsymbol{s}_1^{(k)}, \tag{3.30a}$$

$$\boldsymbol{F}_2^{(k)}\boldsymbol{y}_c \leq \boldsymbol{s}_2^{(k)} + \boldsymbol{F}_3\boldsymbol{x}_N, \tag{3.30b}$$

where $\boldsymbol{F}_1^{(k)}$, $\boldsymbol{F}_2^{(k)}$, $\boldsymbol{F}_3$, $\boldsymbol{s}_1^{(k)}$, $\boldsymbol{s}_2^{(k)}$ are constant matrices/vectors defined in Appendix 3.7.3. An attack pair $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ can form a successful attack if and only if (3.30) has a feasible solution.

The above assumptions (i.e., $\xi_c = \infty$ and $\psi_s(\boldsymbol{p}, \boldsymbol{D})$ returns all the $\boldsymbol{\theta}$'s satisfying (3.5b)-(3.5d)) are needed because: (i) $\xi_c = \infty$ implies that we no longer need the binary variables used to linearize (3.20g) (i.e., $\boldsymbol{w}_f$ and $\boldsymbol{w}_p$ in (40) in Appendix 3.7.1); (ii) when the lower-level optimization returns the feasible region of (3.5), (3.23d) can be replaced by (3.5b)-(3.5d) without introducing binary variables required for transforming (3.5) into its KKT conditions [68].

Our key observation is that a PMU placement $\boldsymbol{\beta}$ can defend against an attack pair $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ by either preventing the physical attack $\boldsymbol{a}_p^{(k)}$ or making (3.30) infeasible. The former can be achieved by adding constraint $\sum_{l:a_{p,l}^{(k)}=1} x_{L,l} \geq 1$ (i.e., at least one attacked line must be incident to a PMU). The latter holds according to Gale's theorem of alternative [56] if and only if there exists $\boldsymbol{q}_1^{(k)}$ and $\boldsymbol{q}_2^{(k)} \geq \boldsymbol{0}$ satisfying

$$(\boldsymbol{F}_1^{(k)})^T\boldsymbol{q}_1^{(k)} + (\boldsymbol{F}_2^{(k)})^T\boldsymbol{q}_2^{(k)} = \boldsymbol{0}, \tag{3.31a}$$

$$(\boldsymbol{s}_1^{(k)})^T\boldsymbol{q}_1^{(k)} + (\boldsymbol{s}_2^{(k)} + \boldsymbol{F}_3\boldsymbol{x}_N)^T\boldsymbol{q}_2^{(k)} < 0, \tag{3.31b}$$

where $\boldsymbol{q}_1^{(k)} \in \mathbb{R}^{m_1}$ and $\boldsymbol{q}_2^{(k)} \in \mathbb{R}^{m_2}$ can be treated as the dual variables for (3.30a) and (3.30b), respectively.

Based on the above observation, the "Attack-Denial" constraints for defending against $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ are:

$$(\boldsymbol{F}_1^{(k)})^T\boldsymbol{q}_1^{(k)} + (\boldsymbol{F}_2^{(k)})^T\boldsymbol{q}_2^{(k)} = \boldsymbol{0}, \tag{3.32a}$$

$$(\boldsymbol{s}_1^{(k)})^T\boldsymbol{q}_1^{(k)} + (\boldsymbol{s}_2^{(k)} + \boldsymbol{F}_3\boldsymbol{x}_N)^T\boldsymbol{q}_2^{(k)} \leq w_{a,k} - 1, \tag{3.32b}$$

$$\sum_{l:a_{p,l}^{(k)}=1} x_{L,l} \geq w_{a,k}, \tag{3.32c}$$

$$\boldsymbol{q}_2^{(k)} \geq \boldsymbol{0}, w_{a,k} \in \{0,1\}, \tag{3.32d}$$

where $\boldsymbol{q}_1^{(k)}$, $\boldsymbol{q}_2^{(k)}$, and $w_{a,k}$ are newly introduced variables. Note that (3.31b) and (3.32b)

are equivalent when $w_k = 0$ since we can scale $\boldsymbol{q}_1^{(k)}$ and $\boldsymbol{q}_2^{(k)}$ to satisfy (3.32b) if (3.31b) holds. The binary variable $w_{a,k}$ indicates which approach to use for defending against $(\boldsymbol{a}_p^{(k)}, e^{(k)})$. When $w_{a,k} = 0$, (3.32c) holds trivially, in which case $\boldsymbol{\beta}$ defends against $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ by satisfying (3.31), i.e., preventing the cyber attack from causing overload-induced tripping at line $e^{(k)}$. When $w_{a,k} = 1$, $\boldsymbol{q}_1^{(k)} = \boldsymbol{0}$ and $\boldsymbol{q}_2^{(k)} = \boldsymbol{0}$ will satisfy the constraints (3.32a)-(3.32b), in which case $\boldsymbol{\beta}$ defends against $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ by preventing the physical attack $\boldsymbol{a}_p^{(k)}$.

Now, we are ready to present the AODC algorithm, where $\hat{\boldsymbol{\beta}}^{(K+1)}$ in Line 6 of Alg. 4 is obtained by solving:

$$\min \quad \|\boldsymbol{\beta}\|_0 \tag{3.33a}$$

$$\text{s.t.} \quad (3.16) - (3.17), (3.32) \text{ for } k = 1, \cdots, K, \tag{3.33b}$$

$$\sum_{i:\hat{\beta}_i'^{(k)}=0} \beta_i \geq 1, k = 1, \cdots, K, \tag{3.33c}$$

$$\boldsymbol{\beta} \in \{0,\ 1\}^{|V|}, \tag{3.33d}$$

where the decision variables are $\boldsymbol{\beta}$, $\boldsymbol{x}_N$, $\boldsymbol{x}_L$, $\boldsymbol{q}_1^{(k)}$, $\boldsymbol{q}_2^{(k)}$, and $w_{a,k}$ for $k = 1, \cdots, K$.

To convert (3.33) to a MILP, we linearize $(\boldsymbol{F}_3\boldsymbol{x}_N)^T\boldsymbol{q}_2^{(k)}$ using McCormick's relaxation. Concretely, note that

$$(\boldsymbol{F}_3\boldsymbol{x}_N)^T\boldsymbol{q}_2^{(k)} = \sum_{u\in V} \boldsymbol{x}_{N,u} \left(\sum_{i=1}^{m_2} F_{3,i,u} q_{2,i}^{(k)} \right), \forall k. \tag{3.34}$$

Assuming that $\sum_i F_{3,i,u} q_{2,i}^{(k)} \in [\underline{M}_F, \overline{M}_F]$, we introduce a continuous auxiliary variable y_u and the following constraints:

$$\underline{M}_F x_{N,u} \leq y_u \leq \overline{M}_F x_{N,u}, \tag{3.35a}$$

$$y_u \leq \left(\sum_{i=1}^{m_2} F_{3,i,u} q_{2,i}^{(k)} \right) + \underline{M}_F x_{N,u} - \underline{M}_F, \tag{3.35b}$$

$$y_u \geq \left(\sum_{i=1}^{m_2} F_{3,i,u} q_{2,i}^{(k)} \right) + \overline{M}_F x_{N,u} - \overline{M}_F. \tag{3.35c}$$

Note that $y_u = \sum_{i=1}^{m_2} F_{3,i,u} q_{2,i}^{(k)}$ if $x_{N,u} = 1$ and $y_u = 0$ otherwise, i.e., $y_u = \boldsymbol{x}_{N,u} \left(\sum_{i=1}^{m_2} F_{3,i,u} q_{2,i}^{(k)} \right)$. Then, $(\boldsymbol{F}_3\boldsymbol{x}_N)^T\boldsymbol{q}_2^{(k)}$ in (3.32b) can be replaced by $\sum_{u\in V} y_u$ subject to (3.35).

AODC guarantees an optimal solution at convergence in the considered special case (see proof in Appendix 3.7.5).

Theorem 3.3.4. *If $\xi_c = \infty$ and $\psi_s(\boldsymbol{p}, \boldsymbol{D})$ returns the feasible region of* (3.5)*, then AODC will converge in finite time to an optimal solution to* (3.24)*.*

Although in the worst case AODC may still enumerate all the attack pairs, which can be exponential in $|E|$, we have observed that in practice it usually converges after identifying a relatively small set of "typical attack pairs", as shown in Table 3.7.

3.3.5 Efficient Heuristics

Although Alg. 4 is guaranteed to find the optimal solution, the computational complexity can grow exponentially with the network size due to the requirement of solving MILPs in each iteration, which motivates us to develop polynomial-time heuristics. A scenario of particular interest is when ξ_p is small, i.e., $\xi_p = \mathcal{O}(1)$. In this case, the total number of attack pairs is polynomial in $|E|$, and thus the number of iterations in AODC and the complexity of computing a new attack pair in each iteration are both polynomial in $|E|$. Our focus in this case is thus on solving (3.33) approximately in polynomial time.

Relaxation: One idea is to directly relax the MILP version of (3.33) into an LP. However, simple LP relaxation will not work:

1. The LP relaxation will invalidate the McCormick relaxation (3.35) for the bilinear term $(\boldsymbol{F}_3\boldsymbol{x}_N)^T\boldsymbol{q}_2^{(k)}$.

2. The feasible region is significantly extended by the LP relaxation due to the adopted big-M modeling technique.

3. Given a continuous solution $\tilde{\boldsymbol{\beta}}$ obtained from the LP relaxation, it is non-trivial to determine which subset of $\Omega(\tilde{\boldsymbol{\beta}})$, if any, can achieve our defense goal.

We have developed a polynomial-time heuristic that can find a better PMU placement. The core of our heuristic is a different "LP relaxation" of (3.33). Recall that the main challenge in directly relaxing the MILP version of (3.33) is the invalidation of (3.35) for linearizing $(\boldsymbol{F}_3\boldsymbol{x}_N)^T\boldsymbol{q}_2^{(k)}$. To overcome this issue, we make the following observation (see proof in Appendix 3.7.5):

Lemma 3.3.3. *Define $\boldsymbol{\Lambda}_{x,p}, \boldsymbol{\Lambda}_{x,n} \in \{0,1\}^{|V|\times m_2}$ such that $(\boldsymbol{\Lambda}_{x,p}\boldsymbol{q}_2)_u$ is the dual variable for* (3.20c) *and $(\boldsymbol{\Lambda}_{x,n}\boldsymbol{q}_2)_u$ is the dual variable for* (3.20d)*. Suppose that the linear system*

$$(\boldsymbol{F}_1^{(k)})^T\boldsymbol{q}_1^{(k)} + (\boldsymbol{F}_2^{(k)})^T\boldsymbol{q}_2^{(k)} = \boldsymbol{0}, \tag{3.36a}$$

$$(\boldsymbol{s}_1^{(k)})^T \boldsymbol{q}_1^{(k)} + (\boldsymbol{s}_2^{(k)} + \boldsymbol{F}_3)^T \boldsymbol{q}_2^{(k)} \leq -1, \tag{3.36b}$$

$$(\boldsymbol{\Lambda}_{x,p} + \boldsymbol{\Lambda}_{x,n})\boldsymbol{q}_2 \leq M_q \underline{\boldsymbol{A}}\boldsymbol{\beta}, \tag{3.36c}$$

$$\boldsymbol{q}_2^{(k)} \geq \mathbf{0},\ \mathbf{1} \geq \boldsymbol{\beta} \geq \mathbf{0} \tag{3.36d}$$

for attack pair $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ *is feasible under* $\boldsymbol{\beta} = \check{\boldsymbol{\beta}}$*, where* M_q *is a large constant (defined in Appendix 3.7.2). Then,* $\boldsymbol{\beta} = \lceil \check{\boldsymbol{\beta}} \rceil$ *satisfies* (3.16)–(3.17) *and* (3.32) *with* $w_{a,k} = 0$ *for the attack pair* $(\boldsymbol{a}_p^{(k)}, e^{(k)})$.

Lemma 3.3.3 suggests that given an attack pair $(\boldsymbol{a}_p^{(k)}, e^{(k)})$, we can relax the mixed integer "Attach-Denial" constraints (3.32) into the linear constraints (3.36) and round up the fractional solution to obtain a valid PMU placement, which is guaranteed to prevent the given attack pair from forming successful attack tuples. According to Gale's theorem of alternative, $\left((\boldsymbol{\Lambda}_{x,p} + \boldsymbol{\Lambda}_{x,n})\boldsymbol{q}_2^{(k)}\right)_u > 0$ only if at least one of (3.20c) and (3.20d) is *effective* for making (3.30) infeasible[3]. Since (3.20c)-(3.20d) is effective if and only if $x_{N,u} = 1$ (under the constraint of $x_{N,u} \in \{0, 1\}$), we use $(\boldsymbol{\Lambda}_{x,p} + \boldsymbol{\Lambda}_{x,n})\boldsymbol{q}_2^{(k)}$ as a proxy of $\boldsymbol{x}_N$ in Lemma 3.3.3.

Lemma 3.3.3 motivates us to formulate the following LP based on a given set $\mathcal{C}$ of infeasible PMU placements and a given set $\{(\boldsymbol{a}_p^{(k)}, e^{(k)})\}_{k=1}^K$ of attack pairs:

$$\min \quad \sum_{u \in V} \beta_u \tag{3.37a}$$

$$\text{s.t.} \quad (3.36) \text{ for } k = 1, \cdots, K, \tag{3.37b}$$

$$\sum_{i:\hat{\beta}_i = 0} \beta_i \geq 1, \forall \hat{\boldsymbol{\beta}} \in \mathcal{C}, \tag{3.37c}$$

where (3.37b) models relaxed "Attack-Denial" constraints and (3.37c) models relaxed "No-Good" constraints. In this sense, (3.37) is a "LP relaxation" of (3.33). However, instead of directly computing a PMU placement from (3.37) which still faces some of the issues for simple LP relaxation, our idea is to use the result of (3.37) to identify important nodes for PMU placement to defend against the given attack pairs in the case of $w_{a,k} = 0$ in (3.32). We will account for the case of $w_{a,k} = 1$ separately in the proposed algorithm to avoid scaling and numerical issues.

Algorithm: The details of the proposed heuristic is given in Alg. 5, which relies on the function *UpdateCandidate(·)* shown in Alg. 6. The logic behind the heuristic is similar to that in AODC, i.e., iteratively updating PMU placements based on newly found attack

[3]We say that an inequality in (3.30) is *effective* for making (3.30) infeasible if removing it will change the feasibility of (3.30).

Algorithm 5: 3-phase Secured PMU Placement

/* Phase-1: find a set $\mathcal{A}_0$ of attack pairs */

Initialization: $k = 1$, $\hat{\boldsymbol{\beta}}^{(k)} = \mathbf{0}$, $\mathcal{A}_0 = \emptyset$, $\mathcal{C} = \emptyset$;

while $\psi_a(\hat{\boldsymbol{\beta}}^{(k)}) > 0$ **do**

  $\mathcal{A}_0 \leftarrow \mathcal{A}_0 \cup \{(\boldsymbol{a}_p^{(k)}, e^{(k)})\}$, where $(\boldsymbol{a}_p^{(k)}, e^{(k)})$ is obtained by solving (3.23) under $\hat{\boldsymbol{\beta}}^{(k)}$;

  $\mathcal{C} \leftarrow \mathcal{C} \cup \{\hat{\boldsymbol{\beta}}^{(k)}\}$, $k \leftarrow k + 1$;

  obtain $\check{\boldsymbol{\beta}}^{(k)}$ by solving (3.37) over $\mathcal{C}$ and $\mathcal{A}_0$;

  Rounding: $\hat{\boldsymbol{\beta}}^{(k)} \leftarrow \lceil \check{\boldsymbol{\beta}}^{(k)} \rceil$;

/* Phase-2: find candidate placements $\{\Omega_i\}_{i=1}^{K_c}$ to defend against $\mathcal{A}_0$ */

Set $\Omega_i := \{u_i\}, i = 1, \cdots, K_c$, where $\{u_i\}_{i=1}^{K_c}$ are the indices of the largest K_c elements of $\check{\boldsymbol{\beta}}^{(k)}$ that is obtained in the last iteration of phase-1;

$\{\Omega_i\}_{i=1}^{K_c}, \mathcal{C} \leftarrow \text{UpdateCandidate}\left(\{\Omega_i\}_{i=1}^{K_c}, \mathcal{A}_0, \mathcal{C}\right)$;

/* Phase-3: augment $\{\Omega_i\}_{i=1}^{K_c}$ to find a placement Ω with $\psi_a(\boldsymbol{\beta}(\Omega)) = 0$ */

while *True* **do**

  $\mathcal{A} \leftarrow \emptyset$;

  for $i \leftarrow 1$ **to** K_c **do**

    if $\psi_a(\boldsymbol{\beta}(\Omega_i)) > 0$ **then** Generate $(\boldsymbol{a}_p^{(i)}, e^{(i)})$ and $\mathcal{A} \leftarrow \mathcal{A} \cup (\boldsymbol{a}_p^{(i)}, e^{(i)})$;

    else Return $\Omega^* = \arg\min_{\Omega_j : \psi_a(\boldsymbol{\beta}(\Omega_j)) = 0} |\Omega_j|$ if $|\Omega^*| \leq 1 + \min_{\Omega_j : \psi_a(\boldsymbol{\beta}(\Omega_j)) > 0} |\Omega_j|$;

  $\{\Omega_i\}_{i=1}^{K_c}, \mathcal{C} \leftarrow \text{UpdateCandidate}\left(\{\Omega_i\}_{i=1}^{K_c}, \mathcal{A}, \mathcal{C}\right)$;

pairs. The questions are: (i) how to generate initial placements, (ii) how to find attack pairs that can cause outages under given placements, and (iii) how to update the given placements to defend against the newly found attack pairs, all in polynomial time. Since this algorithm is designed for the case of $\xi_p = \mathcal{O}(1)$, under which question (ii) is easily solvable, our focus will be on questions (i) and (iii).

We answer question (i) in two phases. Specifically, in *phase-1*, we iteratively find a set of attack pairs $\mathcal{A}_0$ such that solving (3.37) over $\mathcal{A}_0$ leads to a fractional solution $\check{\boldsymbol{\beta}}$ with $\psi_a\left(\lceil \check{\boldsymbol{\beta}} \rceil\right) = 0$. Then in *phase-2*, we search for a set of candidate PMU placements $\{\Omega_i\}_{i=1}^{K_c}$ to defend against $\mathcal{A}_0$ in the hope that $|\Omega_i| < |\Omega(\lceil \check{\boldsymbol{\beta}} \rceil)|$. The motivation for maintaining $K_c > 1$ candidates is to avoid the situation where the computed placement is effective in defending against the given attacks but ineffective for other attacks.

We answer (iii) in Alg. 6, which iteratively augments a given set of candidate placements $\{\Omega_i\}_{i=1}^{K_c}$ to defend against a given set $\mathcal{A}$ of attack pairs. For each candidate

placement not effective against all the attack pairs in $\mathcal{A}$, Alg. 6 will generate K_L and K_A new candidate placements in Line 7 and Lines 8-9, respectively. Then, Line 10 will select the K_c placements most effective in defending against the attack pairs in $\mathcal{A}$ from the pool of $K_c \cdot (K_A + K_L)$ candidate placements. We now characterize the complexity of Alg. 5 (see proof in Appendix 3.7.5).

Theorem 3.3.5. *If $\xi_p = \mathcal{O}(1)$, then the complexity of Alg. 5 is polynomial in $|V|$, $|E|$, and K_c.*

Algorithm 6: UpdateCandidate$\Big(\{\Omega_i\}_{i=1}^{K_c}, \mathcal{A}, \mathcal{C}\Big)$

1 **Initialization:** $\mathcal{A}_i = \mathcal{A}, i = 1, \cdots, K_c$;
2 **while** $\exists i$ *such that* $\mathcal{A}_i \neq \emptyset$ **do**
3 $\mathcal{Q} \leftarrow \emptyset$;
4 **for** $i \leftarrow 1$ **to** K_c **do**
5 **if** $\mathcal{A}_i = \emptyset$ **then** $\mathcal{Q} \leftarrow \mathcal{Q} \bigcup \{\Omega_i\}$ and continue;
6 **else** $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{\beta}(\Omega_i)\}$;
7 $\mathcal{Q} \leftarrow \mathcal{Q} \bigcup (\Omega_i \cup \{v_j\})$ for $j = 1, \cdots, K_L$, where v_j can prevent the j-th most physical attacks in $\mathcal{A}_i$;
8 Solve (3.37) over $\mathcal{A}$, $\mathcal{C}$, and the constraints $\beta_u = 1, \forall u \in \Omega_i$, which results in $\check{\boldsymbol{\beta}}$;
9 $\mathcal{Q} \leftarrow \mathcal{Q} \bigcup (\Omega_i \cup \{u_j\})$ for $j = 1, \cdots, K_A$, where u_j is the index of the j-th largest element in $\{\check{\beta}_u\}_{u \in V \setminus \Omega_i}$;
10 Update $\{\Omega_i\}_{i=1}^{K_c}$ as the K_c elements in $\mathcal{Q}$ that can defend against the most attack pairs in $\mathcal{A}$;
11 $\mathcal{A}_i \leftarrow \{(\boldsymbol{a}_p, e) \in \mathcal{A} | \Omega_i$ cannot defend against $(\boldsymbol{a}_p, e)\}$, $\forall i = 1, \ldots, K_c$;
12 Return $\{\Omega_i\}_{i=1}^{K_c}$ and $\mathcal{C}$;

3.4 Extension to AC Power Flow Model

So far we have assumed the DC power flow approximation for the power grid given in Section 3.2.1. It remains to validate the resulting PMU placement under the AC power flow model that describes the grid state more accurately. To this end, we will address the following questions: given a PMU placement $\Omega_{\text{DC}} \subseteq V$ obtained under the DC power flow model, (i) how to test the feasibility of Ω_{DC} in preventing outages under the AC power flow model, and (ii) how to refine Ω_{DC} if needed to achieve our defense goal under the AC power flow model.

3.4.1 Testing a PMU Placement under AC Model

One challenge to answer the first question is the nonlinear and nonconvex nature of *AC power flow based SCED (AC-SCED)*, which invalidates the transformation of (3.23) into a single-level MILP through KKT conditions. Another challenge lies in formulating a single optimization to maximize the overloading of a target line after SCED (at t_3 in Fig. 3.1). Specifically, since solving nonlinear AC power flow equations usually requires iterative methods (e.g., Newton-Raphson method [31]), we cannot directly formulate the AC-SCED at t_2 and the corresponding ground-truth grid state at t_3 in an optimization problem. Existing works handled this challenge by approximating the grid state at t_3 by the DC power flow model [92, 93] or DC-based line outage distribution factors [40, 94]. However, such DC-based approximations cannot be directly used to compute the magnitude of currents, which determines the overloading and related tripping of lines.

In the following, we provide a method, as shown in Alg. 7, to check the existence of an AC-based CCPA that can cause overloading under a given PMU placement. To overcome the challenges discussed before, we first remove the optimality requirement in AC-SCED, similar to our derivation of "Attack-Denial" constraints in Section 3.3.4. Omitting this optimality requirement is equivalent to allowing the attacker to choose the objective for AC-SCED, which enlarges the feasible region for the attacker's optimization. To jointly model the current at t_3 and the AC-SCED at t_2, we adopt the *linearized approximation of AC power flow equations* [95]. Based on these two strategies, we formulate the following optimization problem for the attacker to maximize the magnitude of current on a given target line e_t under a given physical attack (i.e., $\boldsymbol{a}_p$):

$$\max \quad |\hat{I}_{3,e_t}|^2 \tag{3.38a}$$

$$\text{s.t.} \quad \text{Constraints on } \tilde{\boldsymbol{v}}_2, \tilde{\boldsymbol{\theta}}_2 \text{ to bypass BDD}, \tag{3.38b}$$

$$\text{ACOPF constraints on } \tilde{\boldsymbol{v}}_3, \tilde{\boldsymbol{\theta}}_3, \tag{3.38c}$$

$$\text{Constraints to solve } \hat{\boldsymbol{v}}_3, \hat{\boldsymbol{\theta}}_3, |\hat{\boldsymbol{I}}_3|, \tag{3.38d}$$

where $\tilde{\boldsymbol{v}}_2, \tilde{\boldsymbol{\theta}}_2$ denote the voltage magnitudes and phase angles estimated at t_2 by the control center based on falsified measurements, $\tilde{\boldsymbol{v}}_3, \tilde{\boldsymbol{\theta}}_3$ denote the same variables predicted by AC-SCED for t_3 (computed at t_2), and $\hat{\boldsymbol{v}}_3, \hat{\boldsymbol{\theta}}_3, |\hat{\boldsymbol{I}}_3|$ denote the approximated ground-truth of voltage magnitudes, phase angles and line current magnitude at t_3. The details of (3.38) are given in Appendix 3.7.4. Similar to Table 3.1, for a given variable x, we use $\tilde{x}_2$ to denote its estimate based on falsified measurements at t_2, x_2 to denote its

ground-truth value at t_2, $\tilde{x}_3$ to denote the value predicted by AC-SCED (at t_2) for t_3, and x_3 to denote the ground-truth value at t_3. Given the voltage magnitudes $\tilde{\boldsymbol{v}}_3$ and the phase angles $\tilde{\boldsymbol{\theta}}_3$, the approximated values of x at t_3 is denoted as $\hat{x}_3$.

In (3.38), we have the following three types of constraints and decision variables:

1. Constraint (3.38b) is the counterpart of (3.20) under the AC power flow model, in which the main decision variables are $\tilde{\boldsymbol{v}}_2$ and $\tilde{\boldsymbol{\theta}}_2$. Similar to (3.20), we use $\tilde{\boldsymbol{v}}_2$ and $\tilde{\boldsymbol{\theta}}_2$ as the decision variables to model the cyber attack that can bypass the BDD. Following [40], we adopt the quadratic convex (QC) relaxation [96] in (3.38b) to model the AC power flow equations.

2. As the counterpart of (3.21a)-(3.21c) under the AC power flow model, (3.38c) models the reaction of AC-SCED to the falsified measurements based on the QC relaxation.

3. The real grid state at t_3 is formulated in (3.38d) as the counterpart of (3.21d)-(3.21f), based on the approximation of AC power flow equations proposed in [95].

As we have enlarged the feasible region for the attacker in (3.38b)-(3.38c) by using the QC relaxation, (3.38) models a stronger attack, and hence a PMU placement that prevents overloading under this attack can prevent overloading under the original attack. We will use x^* to denote the value of decision variable x in the optimal solution to (3.38).

Based on (3.38), we develop an algorithm to check the feasibility of a PMU placement $\Omega \subseteq V$ in preventing outages under AC-based CCPA, shown in Algorithm 7. Specifically, at Lines 2, we compute $\boldsymbol{v}_2, \boldsymbol{\theta}_2, |\boldsymbol{I}_2|$ by solving power flow equations. Thus, the counterpart of (3.19) is no longer needed to compute the real grid states after physical attacks. Then, at Line 3, we obtain the optimal solution $(|\hat{I}^*_{3,e_t}|, \tilde{\boldsymbol{v}}^*_3, \tilde{\boldsymbol{\theta}}^*_3)$ to (3.38) for the given attack pair $(\boldsymbol{a}_p, e_t)$ (recall that $|\hat{I}^*_{3,e_t}|$ is the approximated current magnitude on line e_t at time t_3 while $|I^*_{3,e_t}|$ is the corresponding true value). Alg. 7 considers the PMU placement Ω to successfully defend against $(\boldsymbol{a}_p, e_t)$ (i.e., preventing overloading at line e_t under physical attack $\boldsymbol{a}_p$) if one of the following conditions hold:

1. no cyber attack $\boldsymbol{a}_c$ can bypass the BDD, i.e., (3.38) is infeasible, as checked in Line 9, or

2. $|\hat{I}^*_{3,e_t}| \leq \hat{I}_{max,e_t}$ and $|I^*_{3,e_t}| \leq \gamma_e I_{max,e_t}$, as checked in Lines 4–7, where $\hat{I}_{max,e_t}$ (derived in Theorem 3.4.1) is the threshold used by Alg. 7 to detect the tripping of line e_t based on the approximated current magnitude $|\hat{I}^*_{3,e_t}|$.

The use of $\hat{I}_{max,e}$ rather than $\gamma_e I_{max,e}$ allows us to compensate for the approximation error at t_3. As stated in Theorem 3.4.1, under a properly-set $\hat{I}_{max,e}$, a PMU placement Ω is guaranteed to achieve our defense goal under the AC model if Ω can pass the test of Alg. 7, i.e., no overloading is reported. How to bound the approximation errors as assumed in Theorem 3.4.1 is not the focus of this work; we refer interested readers to [95] for details.

Theorem 3.4.1. *Assume that the approximation used in* (3.38d) *satisfies* $|\hat{v}_{3,u} - v_{3,u}| \leq \epsilon_{v,u}, |\hat{\theta}_{3,u} - \theta_{3,u}| \leq \epsilon_{\theta,u}$, $\forall u \in V$ *and* $|\hat{p}_{3,f,e} - p_{3,f,e}| \leq \epsilon_{p,e}$, $|\hat{q}_{3,f,e} - q_{3,f,e}| \leq \epsilon_{q,e}$, $\forall e \in E$. *Then, there exists* $\epsilon_{I,e}, \forall e \in E$ *(see proof in Appendix 3.7.5 for details) and* $\hat{I}_{max,e} := \gamma_e I_{max,e} - \epsilon_{I,e}$ *such that any PMU placement passing the test of Alg. 7 can prevent overload-induced tripping under the AC power flow model.*

Algorithm 7: Test Feasibility of Ω under AC Model

for *each possible attack pair* $(\boldsymbol{a}_p, e_t)$ *under the given PMU placement* Ω **do**
  Obtain $\boldsymbol{v}_2, \boldsymbol{\theta}_2, |\boldsymbol{I}_2|$ from AC power flow equations;
  Solve (3.38) to obtain $|\hat{I}^*_{3,e_t}|$, $\tilde{\boldsymbol{v}}^*_3, \tilde{\boldsymbol{\theta}}^*_3$;
  if (3.38) *is feasible AND* $|\hat{I}^*_{3,e_t}| \leq \hat{I}_{max,e_t}$ **then**
    Compute $|I^*_{3,e_t}|$ from AC power flow equations;
    if $|I^*_{3,e_t}| \leq \gamma_e I_{max,e_t}$ **then**
      Continue;
    else Terminate and report overloading;
  else if (3.38) *is infeasible* **then**
    Continue;
  else Terminate and report overloading;

3.4.2 Refining PMU Placement

In the case that the DC-based PMU placement Ω_{DC} fails the test by Alg. 7, we provide a simple heuristic to augment it into a new placement Ω_{AC} that can achieve our defense goal under the AC model. The intuition is to iteratively augment Ω_{DC} by placing more PMUs until the resulting placement Ω_{AC} can pass the test of Alg. 7. The key question is which node to add. To answer this question, we first augment Ω_{DC} into a PMU placement $\Omega_C := \Omega(\boldsymbol{\beta}_C)$ that can achieve full observability by solving (3.39):

$$\min_{\boldsymbol{\beta}_C \in \{0,1\}^{|V|}} \quad \|\boldsymbol{\beta}_C\|_1 \tag{3.39a}$$

$$\text{s.t.} \qquad \boldsymbol{\beta}_C \geq \boldsymbol{\beta}(\Omega_{\text{DC}}), \tag{3.39b}$$

$$\underline{\boldsymbol{A}}\boldsymbol{\beta}_C \geq 1, \tag{3.39c}$$

where (3.39b) guarantees $\Omega_{\text{DC}} \subseteq \Omega_C$, and (3.39c) forces Ω_C to achieve full observability. Then equipped with Ω_C, we augment Ω_{DC} into Ω_{AC} by Alg. 8. If a PMU placement cannot defend against an attack pair $(\boldsymbol{a}_p, e_t)$ (Line 6), then we update the PMU placement by the following rules:

1. If there exists a node $u \in \Omega_C$ that can prevent the physical attack $\boldsymbol{a}_p$ as in (3.18b), we add node u to the current PMU placement (Line 8).

2. Otherwise, we add the node in Ω_C with the maximum deviation in phase angle due to false data injection (Line 11), with ties broken arbitrarily.

Algorithm 8: Augment PMU Placement for AC Model

```
1  Initialization: Ω_AC = Ω_DC;
2  while True do
3  |  Test Ω_AC through Alg. 7;
4  |  if No overloading is reported then Return Ω_AC;
5  |  else
6  |  |  Let (a_p, e_t) be the attack pair under which overloading is reported, and
   |  |   U := {u ∈ V : ∃e with a_{p,e} = 1, D_{u,e} ≠ 0} (all end-nodes of
   |  |   physically-attacked lines);
7  |  |  if Ω_C ∩ U ≠ ∅ then
8  |  |  |  Arbitrarily choose a node u ∈ Ω_C ∩ U;
9  |  |  else
10 |  |  |  Let θ̃_2, θ_2 be the falsified/true phase angles at t_2 under attack pair
   |  |  |   (a_p, e_t);
11 |  |  |  Set u := arg max_{v∈Ω_C} |θ̃_{2,v} − θ_{2,v}|;
12 |  |  Ω_AC ← Ω_AC ∪ {u};
```

3.5 Numerical Experiments

Simulation Settings: We evaluate our solution against benchmarks in several standard systems: IEEE 30-bus, IEEE 57-bus, IEEE 118-bus, and IEEE 300-bus system, where the system parameters as well as load profiles are obtained from [97]. The parameters for our evaluation are set as follows unless specified otherwise: We set $\alpha = 0.25$ according

to [17]. We allow $\tilde{\boldsymbol{\theta}}_3$ to take any value specified by the attacker subject to (3.5b)-(3.5d), which makes our defense effective under any SCED cost vector. The attacker's capability is set as $\xi_p = 2$, $\xi_c = \infty$ (no constraint on the number of manipulated meters). We set the overload-induced tripping threshold to $\gamma_e = 1.2, \forall e \in E$, which is slightly smaller than the one used in [17] to make the solution more robust. For Alg. 5, we set $K_c = K_A = K_L = 10$.

In the rest of this section, we will compare the performance of Alg. 4 (AONG or AODC) and Alg. 5 with the following benchmarks: (i) PMU placement to achieve full observability as proposed in [98]; (ii) greedily placing PMUs in the descending order of node degrees until attack-induced overload-induced tripping is prevented, referred to as "GreedyDegree". Benchmark (i) represents the current approach, and benchmark (ii) represents a baseline solution under the lowered goal of defense.

Savings in the Number of PMUs: In Table 3.4, we compare the number of secured PMUs required by the proposed algorithms (Alg. 4, Alg. 5) with the benchmarks under the nominal operating point [97]. The minimum number of PMUs required to avoid outages, given by Alg. 4 (either AONG or AODC), is significantly smaller than what is required to achieve full observability. Alg. 5 closely approximates the minimum for the tested systems, but a simple heuristic such as GreedyDegree does not. For IEEE 300-bus system, we have skipped Alg. 4 as neither AODC nor AONG can converge within 72 hours. The details of PMU locations are given in Table 3.2 for DC model and Table 3.3 for AC model. Specifically, in Table 3.2, we give the PMU locations according to the best proposed solution Ω_{PPOP} to PPOP, which is consistent with Table 3.4. In Table 3.3, we present the PMU locations of the solution that can pass the test of Alg. 7 under AC power flow model, obtained by Alg. 8.

First, in Table 3.2, we give the PMU locations according to the best proposed solution Ω_{PPOP} to PPOP, which is consistent with Table 3.4.

Table 3.2. PMU Locations of PPOP under DC Model

	Location of PMUs
IEEE 30-bus system	15, 23
IEEE 57-bus system	12,13,25
IEEE 118-bus system	17,34,37,42,49,72,85,100,118
IEEE 300-bus system	8,20,22,34,38,43,44,48,49,54,64,68, 74,77,79,89,90,94,99,109,119,132, 138,152,185,190,203,216,221,270,271

Then, we evaluate the scenario when the solution by PPOP is used as a temporary PMU placement that will eventually be augmented into a placement achieving full observability, as discussed at the end of Section 3.2 (Remark 2). To this end, we evaluate

Table 3.3. PMU Locations of PPOP under AC Model

	Location of PMUs
IEEE 30-bus system	5,15,23
IEEE 57-bus system	12,13,25
IEEE 118-bus system	17,34,37,42,49,62,72,85,100,118
IEEE 300-bus system	8,20,22,34,38,43,44,48,49,54,64,68, 74,77,79,81,89,90,94,99,109,119,132, 138,152,175,185,190,197,203,216, 221,270,271

Table 3.4. Comparison of the Required Number of PMUs

	30-bus	57-bus	118-bus	300-bus
Alg. 4 (optimal)	2	3	9	—
Alg. 5	2	3	10	31
GreedyDegree	3	3	14	85
Full observability	10	17	32	87

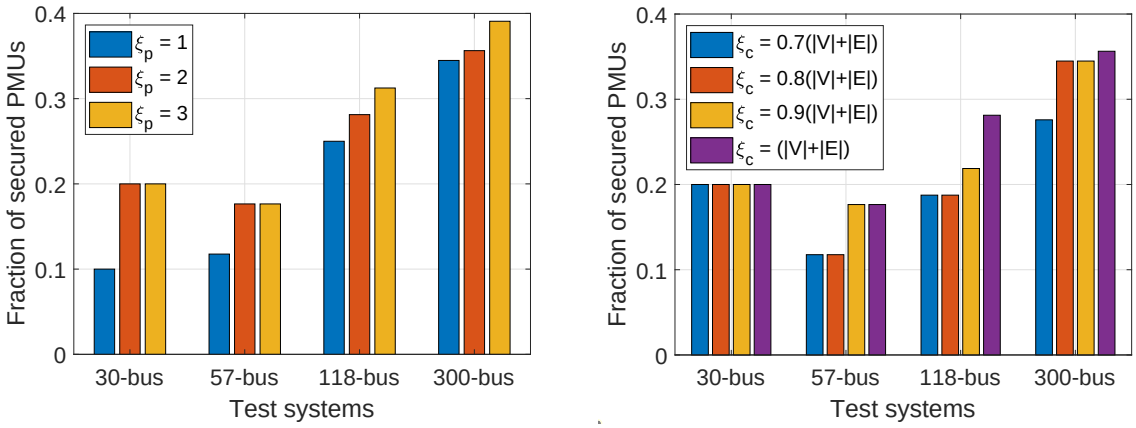
the following metrics: (i) the minimum number of PMUs required by PPOP $|\Omega_{\text{PPOP}}|$, (ii) the minimum number of PMUs for achieving full observability $|\Omega_{\text{FO}}|$, (iii) the size of a full-observability placement Ω_C augmented from Ω_{PPOP} given by (3.39), and (iv) the size of the optimal solution Ω'_{PPOP} to a variation of PPOP with the additional constraint that $\Omega'_{\text{PPOP}} \subseteq \Omega_{\text{FO}}$. In Table 3.5, we observe that (i) $|\Omega'_{\text{PPOP}}|$ is only slightly larger than $|\Omega_{\text{PPOP}}|$, i.e., most of the cost savings by PPOP is still achievable when its solution is required to be consistent with the optimal long-term solution that achieves full observability, but (ii) $|\Omega_C|$ can be notably larger than $|\Omega_{\text{FO}}|$ for large systems, i.e., augmenting an arbitrary solution to PPOP to achieve full observability may require notably more PMUs compared to a clean-slate solution.

Table 3.5. Comparison of #PMUs under Temporary/Long-term Placement

	30-bus	57-bus	118-bus	300-bus
$\lvert\Omega_{\text{PPOP}}\rvert$	2	3	9	31
$\lvert\Omega'_{\text{PPOP}}\rvert$	2	3	10	34
$\lvert\Omega_C\rvert$	10	17	33	95
$\lvert\Omega_{\text{FO}}\rvert$	10	17	32	87

Impact of System Parameters: We evaluate the impact of various system parameters on the number of PMUs required by PPOP, given by Alg. 4 (by Alg. 5 for the 300-bus system).

First, we study the effect of α introduced in (3.12), where a larger α implies a larger feasible region for the attacker. It can be seen from Table 3.6 that (i) PPOP can still



(a) Effect of ξ_p (b) Effect of ξ_c

Figure 3.2. $\frac{\#\text{PMUs required by PPOP}}{\#\text{PMUs required by full observability}}$ (ξ_c=$|V|$+$|E|$ means no ξ_c-constraint).

significantly reduce the required number of PMUs compared to "Full observability" (see Table 3.4) even if α is large, and (ii) PPOP benefits from a small value of α, which signifies the importance of precise load forecasting in defending against CCPA.

Table 3.6. Number of PMUs in PPOP under varying α

	30-bus	57-bus	118-bus	300-bus
$\alpha = 0.01$	1	1	4	24
$\alpha = 0.10$	1	2	6	30
$\alpha = 0.25$	2	3	9	31
$\alpha = 0.50$	3	3	11	34

Then, we vary ξ_p and ξ_c to evaluate the impact of the attacker's capability. As shown in Figure 3.2, (i) defending against a stronger attacker requires more PMUs as expected, (ii) PPOP still requires much fewer PMUs than "Full observability" when the attacker can disconnect multiple lines and manipulate all the meters (except for the secured PMUs), which is stronger than the attack model considered in [17, 74], and (iii) PPOP can save a larger fraction of PMUs in IEEE 57-bus system since $\boldsymbol{f}_{\max}$ given in [97] is large.

In addition, we consider the case that the load profile $\boldsymbol{p}_0$ can vary as shown in (3.25). We assume $\boldsymbol{p}_0 \in [\underline{\kappa}\boldsymbol{p}^{(0)}, \overline{\kappa}\boldsymbol{p}^{(0)}]$, where $\boldsymbol{p}^{(0)}$ is the nominal load profile from [97], $\underline{\kappa} = 0.5$ and $\overline{\kappa}$ is set to the maximum value that keeps (3.5) feasible under $\overline{\kappa}\boldsymbol{p}^{(0)}$. In our evaluations, we set $\overline{\kappa}$ as 1.95, 2.69, 2.41 and 1.61 for IEEE 30-bus, 57-bus, 118-bus and 300-bus systems, respectively. For the given range, PPOP requires 3, 4, 19, and 33 PMUs for the 30-bus, 57-bus, 118-bus, and 300-bus systems, which is more than what is required under a single load profile as expected. Nevertheless, PPOP can still save

PMUs compared to "Full observability" as shown in Table 3.4.

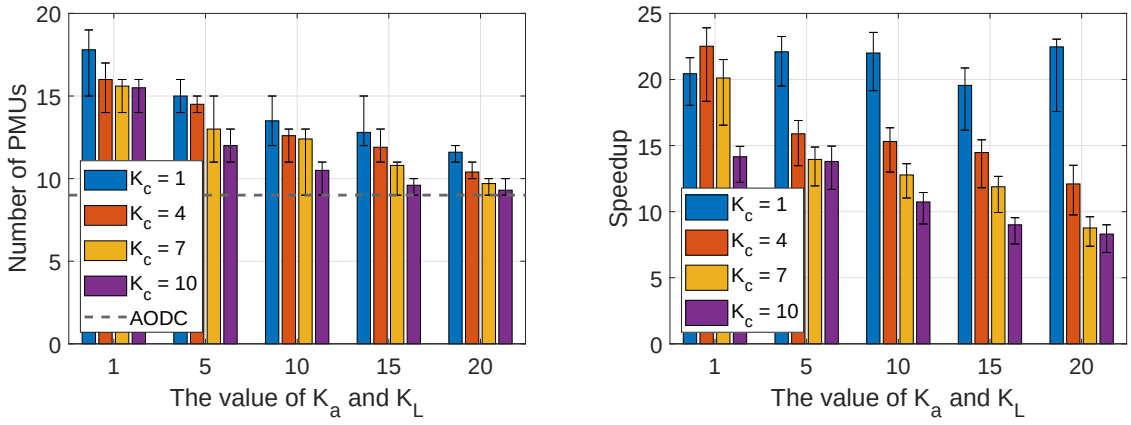
Computational Efficiency: We compare AODC and AONG in terms of the number of iterations (which is also the number of examined attack pairs) and the running time, which is evaluated in a platform with Intel i7-8700 CPU with Gurobi as the solver. Since any feasible solution to (3.27) can form an "No-Good" constraint, we set an upper-bound on the time for solving (3.27), which is 1200 seconds. As shown in Table 3.7, while the two algorithms perform similarly for small systems, AODC converges notably faster for larger systems such as the 118-bus system thanks to its reduced solution space due to the adoption of both "No-Good" and "Attack-Denial" constraints. Note that both algorithms converge after examining a small fraction of possible attack pairs (the total number of attack pairs is 33620, 252800, and 3200130 for these systems, respectively).

Table 3.7. Number of iterations/Convergence time (10^3 sec)

	30-bus	57-bus	118-bus
AODC	8/0.021	3/2.188	16/26.64
AONG	7/0.014	4/2.163	78/74.44

Moreover, we use IEEE 118-bus system as an example to demonstrate the trade-off in tuning the parameters K_c, K_A, K_L for Alg. 5 (assuming $K_A = K_L$). We run Alg. 5 for 5 times under each setting due to the randomness in solving (3.23) and breaking ties. The results are given in Fig. 3.3, where the bar denotes the mean and the error bar denotes the minimum/maximum. In Fig. 3.3 (b), we show the speedup of the heuristic compared to AODC in convergence time, i.e., (time of AODC)/(time of heuristic). We observe that (i) Alg. 5 can return a good solution when $K_c \geq \%10 \cdot |V|$ and $K_A = K_L \geq K_c$, and (ii) under this configuration, Alg. 5 is significantly faster than AODC at a small cost of requiring a couple of more PMUs.

Extension to AC model: We compare the solution Ω_{AC} obtained by Alg. 8 with the best previous solution Ω_{DC} obtained under the DC approximation. As shown in Table 3.8, although the DC-based solution may need augmentation to defend against AC-based CCPA, the gap (i.e., $|\Omega_{\text{AC}}| - |\Omega_{\text{DC}}|$) is small. More importantly, $|\Omega_{\text{AC}}|$ is still much smaller (by 60–80%) than the number of PMUs $|\Omega_{\text{FO}}|$ required to achieve full observability (see Table 3.5), indicating the efficacy of our approach of first computing an initial solution under the DC approximation and then augmenting it to achieve our defense goal under the AC model. We note that the values of $|\Omega_{\text{AC}}|$ in Table 3.8 are only upper bounds on the number of PMUs required to prevent outages under AC-based CCPA, suggesting great potential of saving PMUs by adopting the proposed defense goal.



(a) Number of secured PMUs (b) Computation time

Figure 3.3. The performance of Alg. 5 under different K_c, K_A, and K_L.

Table 3.8. Number of PMUs Under AC Power Flow Model

	30-bus	57-bus	118-bus	300-bus
$\lvert\Omega_{\mathrm{AC}}\rvert$	3	3	10	34
$\lvert\Omega_{\mathrm{DC}}\rvert$	2	3	9	31

3.6 Conclusion

We formulate a tri-level optimization problem under the DC power flow model to find the optimal secured PMU placement to defend against the coordinated cyber-physical attack (CCPA) in the smart grid. Rather than completely eliminating the attack, we propose to limit the impact of the attack by preventing overload-induced outages. To solve the proposed problem, we first transform it into a bi-level MILP and then propose an alternating optimization algorithm framework to obtain optimal solutions. The core of the proposed algorithm framework is constraint generation based on infeasible placements, for which we develop two constraint generation approaches. Furthermore, we propose a polynomial-time heuristic algorithm that can scale to large-scale grids. In addition, we demonstrate how to extend the obtained PMU placement to achieve our defense goal under the AC power flow model. Our experimental results on standard test systems demonstrate great promise of the proposed approach in reducing the requirement of PMUs. Our work lays the foundation for tackling a number of further questions in future work, e.g., how to characterize the optimal attack without solving MILPs, how to directly optimize the PMU placement for outage prevention under the AC model, and how to improve the robustness of the solution against the failures of PMUs themselves.

3.7 Appendices

3.7.1 Appendix A: MILP Formulation of Attacker's Problem

In this section, we will demonstrate how to transform (3.23) into a MILP, which can be efficiently solved by existing solver such as Gurobi.

To begin with, we give an overview of the PPOP, as shown in Fig. 3.4.

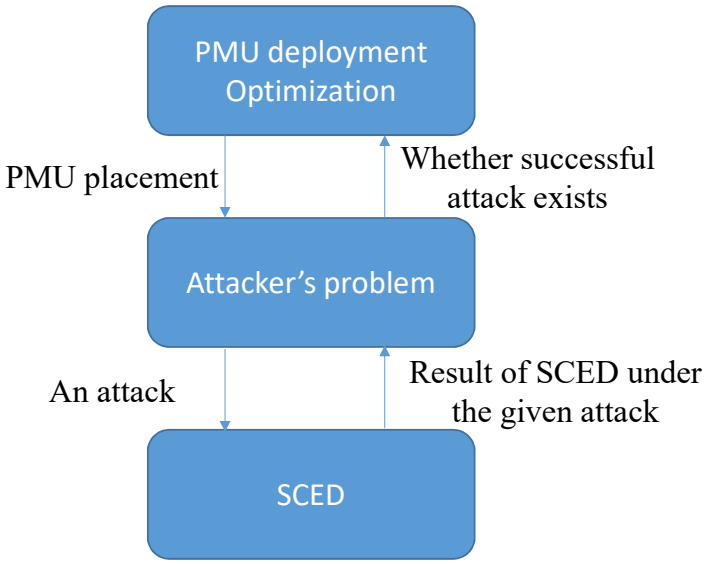


Figure 3.4. Overview of the PPOP

We first consider the case that lower-level optimization (3.5) returns the set of $\boldsymbol{\theta}$'s satisfying (3.5b)-(3.5d), i.e., it returns the feasible region of SCED rather than a single solution. In this case, (3.23) becomes a single-level problem.

Below, we show how to convert the single-level formulation of (3.23) into a MILP. To convert (3.18) and (3.23e) into linear constraints, we introduce a constant $M_{2,\theta}$ (defined in Appendix 3.7.2) such that (3.18a) holds if and only if the following holds:

$$\tilde{\theta}_{2,u} - \theta_{2,u} \leq M_{2,\theta} \cdot (1 - \boldsymbol{x}_{N,u}), \tag{3.40a}$$

$$\tilde{\theta}_{2,u} - \theta_{2,u} \geq -M_{2,\theta} \cdot (1 - \boldsymbol{x}_{N,u}), \tag{3.40b}$$

and similar conversion applies to (3.18b). As for (3.23e), by defining a sufficiently large constant $M_{\pi,e}$ (see Appendix 3.7.2) and two binary auxiliary variables $\pi_{n,e}, \pi_{p,e}$ to get

rid of the absolute value operation, (3.23e) is transformed into

$$-M_{\pi,e}\cdot(1-\pi_{p,e}) < \frac{f_{3,e}}{f_{\max,e}} - \gamma_e \le M_{\pi,e}\cdot\pi_{p,e}, \tag{3.41a}$$

$$-M_{\pi,e}\cdot(1-\pi_{n,e}) < \frac{-f_{3,e}}{f_{\max,e}} - \gamma_e \le M_{\pi,e}\cdot\pi_{n,e}. \tag{3.41b}$$

We claim that $\pi_e = \pi_{n,e} + \pi_{p,e}$. To see this, suppose that $f_{3,e} \ge 0$. Then, we must have $-\frac{f_{3,e}}{f_{\max,e}} - \gamma_e \le 0$ and thus $\pi_{n,e} = 0$, while $\frac{|f_{3,e}|}{f_{\max,e}} - \gamma_e = \frac{f_{3,e}}{f_{\max,e}} - \gamma_e$ and thus $\pi_{p,e} = \pi_e$. Notice that we must have $\pi_e = 1$ if $|f_{3,e}| - \gamma_e \cdot f_{\max,e} > 0$, while $|f_{3,e}| - \gamma_e \cdot f_{\max,e} \le 0$ leads to $\pi_e = 0$.

To linearize (3.20g), we introduce binary variables $\boldsymbol{w}_f \in \{0,1\}^{|m_L|}$ and $\boldsymbol{w}_p \in \{0,1\}^{|m_N|}$ for data injection on line measurements and node measurements, respectively. Then, (3.20g) can be transformed into (see definitions of $M_{c,f},\ M_{c,p}$ in Appendix 3.7.2)

$$-M_{c,f}\boldsymbol{w}_f \le \boldsymbol{\Lambda}_f\left(\tilde{\boldsymbol{f}}_2 - \boldsymbol{f}_2\right) \le M_{c,f}\boldsymbol{w}_f, \tag{3.42a}$$

$$-M_{c,p}\boldsymbol{w}_p \le \boldsymbol{\Lambda}_p\left(\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_2 - \boldsymbol{p}_0\right) \le M_{c,p}\boldsymbol{w}_p, \tag{3.42b}$$

$$\mathbf{1}^T\boldsymbol{w}_f + \mathbf{1}^T\boldsymbol{w}_p \le \xi_c. \tag{3.42c}$$

Together, the above techniques transform (3.23) into a MILP. Specifically, the binary decision variables are $\{\boldsymbol{\pi}_n, \boldsymbol{\pi}_p, \boldsymbol{a}_p, \boldsymbol{w}_f, \boldsymbol{w}_p\}$, continuous variables are $\{\tilde{\boldsymbol{\theta}}_2, \tilde{\boldsymbol{\theta}}_3, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3, \boldsymbol{f}_2, \boldsymbol{f}_3, \tilde{\boldsymbol{f}}_2, \boldsymbol{f}_{con}\}$, where $\boldsymbol{w}_f, \boldsymbol{w}_p$ are introduced auxiliary variables. Then, the full formulation without considering the optimality of (3.5) is given as follows.

$$\max \quad \|\boldsymbol{\pi}_p + \boldsymbol{\pi}_n\|_0 \tag{3.43a}$$

s.t.

$$\boldsymbol{\Delta}^{-1}\underline{\boldsymbol{A}}\boldsymbol{\beta} \le \boldsymbol{x}_N \le \boldsymbol{\Delta}^{-1}\underline{\boldsymbol{A}}\boldsymbol{\beta} + \frac{\|\boldsymbol{\Delta}\|_\infty - 1}{\|\boldsymbol{\Delta}\|_\infty}, \tag{3.43b}$$

$$\frac{1}{2}|\boldsymbol{D}|^T\boldsymbol{\beta} \le \boldsymbol{x}_L \le \frac{1}{2}|\boldsymbol{D}|^T\boldsymbol{\beta} + \zeta, \tag{3.43c}$$

$$-(\mathbf{1}-\boldsymbol{a}_p) \le \operatorname{diag}\left(\boldsymbol{\gamma}\odot\boldsymbol{f}_{\max}\right)^{-1}\boldsymbol{f}_2 \le \mathbf{1}-\boldsymbol{a}_p, \tag{3.43d}$$

$$\tilde{\boldsymbol{D}}\boldsymbol{f}_2 = \boldsymbol{p}_0, -M_{2,f}\boldsymbol{a}_p \le \boldsymbol{\Gamma}\boldsymbol{D}\boldsymbol{\theta}_2 - \boldsymbol{f}_2 \le M_{2,f}\boldsymbol{a}_p, \tag{3.43e}$$

$$-\boldsymbol{f}_{\max} \le \tilde{\boldsymbol{f}}_2 \le \boldsymbol{f}_{\max},\ \boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T\tilde{\boldsymbol{\theta}}_2 - \tilde{\boldsymbol{f}}_2 = 0, \tag{3.43f}$$

$$-\alpha|\boldsymbol{p}_0| \le \tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2 - \boldsymbol{p}_0 \le \alpha|\boldsymbol{p}_0|, \tag{3.43g}$$

$$\boldsymbol{\Lambda}_g\tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2 = \boldsymbol{\Lambda}_g\boldsymbol{p}_0, \tag{3.43h}$$

$$-M_{c,f}\boldsymbol{w}_f \le \boldsymbol{\Lambda}_f\left(\tilde{\boldsymbol{f}}_2 - \boldsymbol{f}_2\right) \le M_{c,f}\boldsymbol{w}_f, \tag{3.43i}$$

$$-M_{c,p}\boldsymbol{w}_p \le \tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_2 - \boldsymbol{p}_0 \le M_{c,p}\boldsymbol{w}_p, \tag{3.43j}$$

$$\mathbf{1}^T\boldsymbol{w}_f + \mathbf{1}^T\boldsymbol{w}_p \le \xi_c,\ \|\boldsymbol{a}_p\|_0 \le \xi_p, \tag{3.43k}$$

$$\tilde{\theta}_{2,u} - \theta_{2,u} \le M_{2,\theta}\cdot(1-\boldsymbol{x}_{N,u}), \tag{3.43l}$$

$$\tilde{\theta}_{2,u} - \theta_{2,u} \ge -M_{2,\theta}\cdot(1-\boldsymbol{x}_{N,u}), \tag{3.43m}$$

$$\boldsymbol{p}_{g,min} \le \boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3 \le \boldsymbol{p}_{g,max}, \tag{3.43n}$$

$$-\boldsymbol{f}_{\max} \le \boldsymbol{\Gamma}\boldsymbol{D}^T\tilde{\boldsymbol{\theta}}_3 \le \boldsymbol{f}_{\max}, \tag{3.43o}$$

$$\boldsymbol{\Lambda}_d\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3 = \boldsymbol{\Lambda}_d\tilde{\boldsymbol{D}}\tilde{\boldsymbol{f}}_2, \tag{3.43p}$$

$$-M_{3,a}(\mathbf{1}-\boldsymbol{a}_p) \le \boldsymbol{f}_3 \le M_{3,a}(\mathbf{1}-\boldsymbol{a}_p), \tag{3.43q}$$

$$\boldsymbol{\Lambda}_d\tilde{\boldsymbol{D}}\boldsymbol{f}_3 = \boldsymbol{\Lambda}_d\boldsymbol{p}_0, \quad \boldsymbol{\Lambda}_g\tilde{\boldsymbol{D}}\boldsymbol{f}_3 = \boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3, \tag{3.43r}$$

$$-M_{3,f}\boldsymbol{a}_p \le \boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T\boldsymbol{\theta}_3 - \boldsymbol{f}_3 \le M_{3,f}\boldsymbol{a}_p, \tag{3.43s}$$

$$\theta_{2,u_0} = \theta_{3,u_0} = \tilde{\theta}_{2,u_0} = \tilde{\theta}_{3,u_0} = 0, \tag{3.43t}$$

$$-\cdot(1-\pi_{p,e}) < \frac{f_{3,e}}{M_{\pi,e}f_{\max,e}} - \frac{\gamma_e}{M_{\pi,e}} \le \cdot\pi_{p,e}, \forall e, \tag{3.43u}$$

$$-\cdot(1-\pi_{n,e}) < \frac{-f_{3,e}}{M_{\pi,e}f_{\max,e}} - \frac{\gamma_e}{M_{\pi,e}} \le \cdot\pi_{n,e}, \forall e, \tag{3.43v}$$

$$\tilde{\boldsymbol{D}}_u\boldsymbol{f}_{con} = \begin{cases} |V|-1, & \text{if } u = u_0, \\ -1, & \text{if } u \in V\setminus\{u_0\}, \end{cases}, \tag{3.43w}$$

$$-|V|\cdot(1-a_{p,e}) \le f_{con,e} \le |V|\cdot(1-a_{p,e}) \tag{3.43x}$$

The constraints (3.43b)-(3.43c) correspond to (3.16)-(3.17), (3.43d)-(3.43e) correspond to (3.19a)-(3.19c), (3.43f)-(3.43k) correspond to (3.20), (3.43l)-(3.43m) correspond to (3.40), (3.43n)-(3.43s) correspond to (3.21), (3.43t) corresponds to (3.23c), (3.43u)-(3.43v) correspond to (3.23e), (3.43w)-(3.43x) correspond to (3.14).

If we do not relax the optimality requirements in (3.5), we need to introduce additional binary variables $\{\boldsymbol{r}_{fl}, \boldsymbol{r}_{fu}, \boldsymbol{r}_{gl}, \boldsymbol{r}_{gu}\}$ and continuous dual variables $\{\boldsymbol{\mu}_b, \boldsymbol{\mu}_c, \boldsymbol{\mu}_d, \boldsymbol{\mu}_e, \boldsymbol{\mu}_g\}$ to transform (3.5) into a linear system by using its KKT conditions [68]. Specifically, we add the following linear system into (3.43) for the completeness of KKT conditions of (3.5):

$$\tilde{\boldsymbol{B}}\boldsymbol{\Lambda}_d^T\boldsymbol{\mu}_b + \tilde{\boldsymbol{D}}\boldsymbol{\Gamma}\boldsymbol{\mu}_c + \tilde{\boldsymbol{D}}\boldsymbol{\Gamma}\boldsymbol{\mu}_d + \tilde{\boldsymbol{B}}\boldsymbol{\Lambda}_g^T\boldsymbol{\mu}_e - \tilde{\boldsymbol{B}}\boldsymbol{\Lambda}_g^T\boldsymbol{\mu}_g = -\tilde{\boldsymbol{B}}\boldsymbol{\Lambda}_g^T\boldsymbol{\phi} \tag{3.44a}$$

$$\boldsymbol{\mu}_c - M\boldsymbol{r}_{fl} \le \mathbf{0}, \tag{3.44b}$$

$$\boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T\tilde{\boldsymbol{\theta}}_3 + M\boldsymbol{r}_{fl} \le M - \boldsymbol{f}_{\max} \tag{3.44c}$$

$$\boldsymbol{\mu}_d - M\boldsymbol{r}_{fu} \le \mathbf{0}, \tag{3.44d}$$

$$-\boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T\tilde{\boldsymbol{\theta}}_3 + M\boldsymbol{r}_{fl} \leq M - \boldsymbol{f}_{\max} \tag{3.44e}$$

$$\boldsymbol{\mu}_e - M\boldsymbol{r}_{gl} \leq \mathbf{0}, \tag{3.44f}$$

$$\boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3 + M\boldsymbol{r}_{gl} \leq \boldsymbol{p}_{g,min} + M\mathbf{1} \tag{3.44g}$$

$$\boldsymbol{\mu}_g - M\boldsymbol{r}_{gu} \leq \mathbf{0}, \tag{3.44h}$$

$$-\boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3 + M\boldsymbol{r}_{gu} \leq -\boldsymbol{p}_{g,max} + M\mathbf{1} \tag{3.44i}$$

$$\boldsymbol{r}_{gl} + \boldsymbol{r}_{gu} \leq \mathbf{1} \tag{3.44j}$$

$$\boldsymbol{r}_{fl} + \boldsymbol{r}_{fu} \leq \mathbf{1} \tag{3.44k}$$

$$\boldsymbol{\mu}_c, \boldsymbol{\mu}_d, \boldsymbol{\mu}_e, \boldsymbol{\mu}_g \geq \mathbf{0} \tag{3.44l}$$

Compared to the attacker's formulations in [71, 88] that also optimize the location of physical attacks, the key advantage of (3.23) is avoiding McCormick's relaxation for bilinear terms (3.22) and reducing the numbers of variables and constraints. Specifically, McCormick's relaxation in [88] will introduce $2|E||V|$ additional continuous variables and $8|E||V|$ additional constrains. The cost of avoiding bilinear term in (3.23) is the additional variables $\boldsymbol{f}_2, \boldsymbol{f}_3, \tilde{\boldsymbol{f}}_2$ and the associated constraints, although the benefit usually outweighs the cost. For example, for the IEEE 118-bus system, the formulation in [88] has 44436 continuous variables and 178,596 constraints, while (3.23) only has 1216 continuous variables and 5,802 constraints.

3.7.2 Appendix B: Calculation of Big-M

In this section, we will explain how to calculate $M_{2,a,e}$ in (3.19a), $M_{2,f}$ in (3.19c), $M_{3,a}$ in (3.21d), $M_{2,\theta}$ in (3.40), $M_{3,f}$ in (3.21f), $M_{\pi,e}$ in (3.41), $\overline{M}_F, \underline{M}_F$ in (3.35), $M_{c,f}, M_{c,p}$ in (3.42) and M_q in (3.36c). In this section, we denote $\mathcal{N} = (V, E)$ as the graph before physical attack while $\mathcal{N}' = (V, E')$ as the graph after attack.

We first show how to calculate $M_{2,a}$, $M_{2,f}$ and $M_{2,\theta}$. Suppose that the power grid is designed to be robust to $N-k$ contingency. Then, the value of $M_{2,a}$ depends on $\xi_p - k$. If $\xi_p - k \leq 0$, then we can set $M_{2,a,e} := f_{\max,e}$ or $M_{2,a,e} := \gamma_e f_{\max,e}$, since no $\boldsymbol{a}_p$ can cause overloading. Otherwise, we set $M_{2,a,e} := C_{2,a}\gamma_e f_{\max,e}$ with a parameter $C_{2,a} > 1$. In our simulations, we find that $C_{2,a} := 3$ suffices since $\xi_p - k$ is usually small. Next, we bound $|\boldsymbol{\theta}_2|$ by defining $M_{\theta_2,u} \geq \max_{\boldsymbol{a}_p} |\theta_{2,u}|$ and $M_{\theta_2} \geq \max_{\boldsymbol{a}_p} \max_u |\theta_{2,u}|$ since the value of $\boldsymbol{\theta}_2$ depends on $\boldsymbol{a}_p$. An intuitive way of obtaining $M_{\theta_2,u}$ is enumerating all possible values of $\boldsymbol{a}_p$, whose time complexity is polynomial in $|E|$ and $|V|$ if $\xi_p = \mathcal{O}(1)$. Here we provide another way of bounding $M_{\theta_2,u}$. Due to our assumption of the connected $\mathcal{N}$, there exists at least one path in $\mathcal{N}$ connecting the reference node u_0 to each node $u \in V$. Moreover,

for each path connecting u_0 and u, say $Pa(u_0, u) := (e_0, e_1, \cdots, e_J)$ where $e_0 = (u_0, v_1)$, $e_j = (v_j, v_{j+1})$ and $e_J = (v_J, u)$, we have $\theta_{2,u} - \theta_{2,u_0} = \theta_{2,s} - \theta_{2,v_1} + \theta_{2,v_1} - \cdots + \theta_{2,v_J} - \theta_{2,t}$, which leads to

$$\begin{aligned} \max_{\boldsymbol{a}_p} |\theta_{2,u}| &= \max_{\boldsymbol{a}_p} |\theta_{2,u} - \theta_{2,u_0}| \\ &\leq \sum_{j=0}^{J} r_{e_j} M_{2,a,e_j} := M_{Pa}(\theta_{2,u}) \end{aligned} \tag{3.45}$$

since $\theta_{2,u_0} = 0$ in our assumption and $|\theta_{2,v_j} - \theta_{2,v_{j+1}}| \leq r_{e_j} M_{2,a,e_j}$ due to (3.19a). Denote n_p as the number of different paths connecting u and u_0. Then, since the physical attack will disconnect at most ξ_p lines, we set $M_{\theta_2,u} := \max\{M_{Pa_i}(\theta_{2,u})\}_{i=1}^{\min\{\xi_p+1, n_p\}}$.

Equipped with $M_{\theta_2,u}, u \in V$, we can calculate $M_{2,f}$ and $M_{2,\theta}$. We define an intermediate constant $M_{2,f,e}$ for each line such that $M_{2,f} = \max_{e\in E} M_{2,f,e}$. Then, for $e = (u, v)$ we can set $M_{2,f,e} := r_e(M_{\theta_2,u} + M_{\theta_2,v})$ since $|\Gamma_e \tilde{\boldsymbol{d}}_e^T \boldsymbol{\theta}_2 - f_{2,e}| > 0$ only if $a_{p,e} = 1$ and $f_{2,e} = 0$.

To obtain $M_{2,\theta}$, we first bound $M_{\tilde{\theta}_2,u} \geq \max_{\boldsymbol{a}_p, \boldsymbol{a}_c} |\tilde{\theta}_{2,u}|, u \in V$ in a similar way as that in (3.45). Specifically, since $\tilde{\boldsymbol{\theta}}_2$ is estimated by CC based on the topology $\mathcal{N}$, we can arbitrarily choose one path $(e_0, e_1, \cdots, e_J)$ in $\mathcal{N}$ that connects u and u_0 and set

$$M_{\tilde{\theta}_2,u} := \sum_{j=0}^{J} r_{e_j} f_{max,e_j} \geq \max_{\boldsymbol{a}_p, \boldsymbol{a}_c} |\tilde{\theta}_{2,u}|. \tag{3.46}$$

Then, we can set $M_{2,\theta} := \max_{u\in V}(M_{\tilde{\theta}_2,u} + M_{\theta_2,u})$.

Now, we are ready to demonstrate the calculation of $M_{3,a}$ and $M_{3,f}$. As for $M_{3,a}$, we only require $M_{3,a,e} > \gamma_e f_{\max,e}$ and $M_{3,a} \geq \max_{e\in E} \gamma_e f_{\max,e}$ so that the attacker can cause outages over any lines. In practice, we can set $M_{3,a} := c \max_{e\in E} \gamma_e f_{\max,e}$ with $c > 1$. As for $M_{3,f}$, we again first show that we can bound $|\theta_{3,u}| \leq M_{\theta_3,u}$ without hurting the attacker's objective. We notice that the topology of grid at t_3 before lines facing outage automatically disconnect themselves is still $\mathcal{N}'$. Thus, we can set $M_{\theta_3,u}$ similarly as $M_{\theta_2,u}$, except that (3.45) becomes:

$$\max_{\boldsymbol{a}_p} |\theta_{3,u}| \leq \sum_{j=0}^{J} r_{e_j} M_{a,e_j} := M_{Pa}(\theta_{3,u}). \tag{3.47}$$

Then, we can set $M_{\theta_3,u} := \max\{M_{Pa_i}(\theta_{3,u})\}_{i=1}^{\min\{\xi_p+1, n_p\}}$, $M_{3,f,e} := r_e(M_{\theta_3,u} + M_{\theta_3,v})$ for $e = (u, v) \in E$, and $M_{3,f} = \max_{e\in E} M_{3,f,e}$.

Equipped with $M_{3,a,e}$, $M_{\pi,e}$ can be easily set as $c \cdot (\frac{M_{3,a,e}}{f_{\max,e}} + \gamma_e)$ with any constant

$c > 1$.

We can set $\overline{M}_F$ as 0 since $\boldsymbol{q}_2 \geq \mathbf{0}$ and $F_{3,i,u} \in \{0, -M_{2,\theta}\}$, $\forall i, u$. There is no simple guidelines for $\underline{M}_F$ in (3.35) since it is the bound for dual variables. In practice, we can initialize $\underline{M}_F$ to a given value and solve (3.33) for each attack pair separately. Then, we iteratively decrease $\underline{M}_F$ until (3.33) is feasible under each attack pair separately. In our simulations, we set $\underline{M}_F := -M_{2,\theta}^2$. Equipped with $\underline{M}_F$, we can set $M_q := \frac{2\underline{M}_F}{M_{2,\theta}}$.

Finally, we demonstrate how to set $M_{c,f}$ and $M_{c,p}$. Due to (3.19a) and (3.20a), we have $|\tilde{f}_{2,e} - f_{2,e}| \leq (1+\gamma_e) f_{\max,e}$, which implies that we can set $M_{c,f} := \max_{e \in E}(1+\gamma_e) f_{\max,e}$. Similarly, we can set $M_{c,p} := \alpha \|\boldsymbol{p}_0\|_\infty$ due to (3.20e).

3.7.3 Appendix C: The Coefficient Matrices in Attacker's Problem

In this section, we will demonstrate the details of (3.30). The linear system (3.30a) is the composition of (3.20f), (3.21e) and (3.21c), which can be expanded into:

$$\begin{bmatrix} \boldsymbol{\Lambda}_g \tilde{\boldsymbol{B}} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \boldsymbol{\Lambda}_d \boldsymbol{B} \\ \mathbf{0} & -\boldsymbol{\Lambda}_g \tilde{\boldsymbol{B}} & \boldsymbol{\Lambda}_g \boldsymbol{B} \\ \boldsymbol{\Lambda}_d \tilde{\boldsymbol{B}} & -\boldsymbol{\Lambda}_d \tilde{\boldsymbol{B}} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{\theta}}_2 \\ \tilde{\boldsymbol{\theta}}_3 \\ \boldsymbol{\theta}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{\Lambda}_g \boldsymbol{p}_0 \\ \boldsymbol{\Lambda}_d \boldsymbol{p}_0 \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix} \tag{3.48}$$

as well as $\tilde{\theta}_{2,u_0} = \tilde{\theta}_{3,u_0} = \theta_{3,u_0} = 0$. For a given attack pair $(\boldsymbol{a}_p, e)$ and the corresponding $\boldsymbol{\theta}_2$, the expansion of (3.30b) is

$$\begin{array}{c} \begin{matrix} \tilde{\boldsymbol{\theta}}_2 & \tilde{\boldsymbol{\theta}}_3 & \boldsymbol{\theta}_3 & \boldsymbol{s}_2 + \boldsymbol{F}_3 \boldsymbol{x}_N \end{matrix} \\ \begin{bmatrix} \tilde{\boldsymbol{B}} & \mathbf{0} & \mathbf{0} & \boldsymbol{p}_0 + \alpha |\boldsymbol{p}_0| \\ -\tilde{\boldsymbol{B}} & \mathbf{0} & \mathbf{0} & -\boldsymbol{p}_0 + \alpha |\boldsymbol{p}_0| \\ \boldsymbol{I}_{|V|} & \mathbf{0} & \mathbf{0} & \boldsymbol{\theta}_2 + M_\theta (1 - \boldsymbol{x}_N) \\ -\boldsymbol{I}_{|V|} & \mathbf{0} & \mathbf{0} & -\boldsymbol{\theta}_2 + M_\theta (1 - \boldsymbol{x}_N) \\ 0 & 0 & -\Gamma_e \boldsymbol{d}_e^T & -\gamma_e f_{\max,e} \\ \mathbf{0} & \tilde{\boldsymbol{D}}^T \boldsymbol{\Gamma} & \mathbf{0} & \boldsymbol{f}_{\max} \\ \mathbf{0} & -\tilde{\boldsymbol{D}}^T \boldsymbol{\Gamma} & \mathbf{0} & \boldsymbol{f}_{\max} \\ \mathbf{0} & \boldsymbol{\Lambda}_g \tilde{\boldsymbol{B}} & \mathbf{0} & \boldsymbol{p}_{g,max} \\ \mathbf{0} & -\boldsymbol{\Lambda}_g \tilde{\boldsymbol{B}} & \mathbf{0} & -\boldsymbol{p}_{g,min} \\ \tilde{\boldsymbol{D}}^T \boldsymbol{\Gamma} & \mathbf{0} & \mathbf{0} & \boldsymbol{f}_{\max} \\ -\tilde{\boldsymbol{D}}^T \boldsymbol{\Gamma} & \mathbf{0} & \mathbf{0} & \boldsymbol{f}_{\max} \end{bmatrix} \end{array} \tag{3.49}$$

Table 3.9. Notations for AC power flow

Notation	Description
$\boldsymbol{p}/\boldsymbol{q} \in \mathbb{C}^{\lvert V\rvert}$	Active/reactive power injection
$\vec{v}_u = v_u e^{j\cdot\theta_u}$	node voltage
$\tilde{\boldsymbol{Y}}_{bus} = \tilde{\boldsymbol{G}}_{bus} + j\tilde{\boldsymbol{B}}_{bus}$	Bus admittance matrix
$\tilde{\boldsymbol{Y}}_f/\tilde{\boldsymbol{Y}}_t \in \mathbb{C}^{\lvert E\rvert\times\lvert V\rvert}$	*From*/*to* end admittance matrix
$\boldsymbol{C}_f/\boldsymbol{C}_t \in \{0,1\}^{\lvert E\rvert\times\lvert V\rvert}$	*From*/*to* end incidence matrix
$\boldsymbol{p}_f/\boldsymbol{p}_t \in \mathbb{C}^{\lvert E\rvert}$	*From*/*to* end active power flow
$\boldsymbol{q}_f/\boldsymbol{q}_t \in \mathbb{C}^{\lvert E\rvert}$	*From*/*to* end reactive power flow
$\lvert\boldsymbol{I}_f\rvert^2/\lvert\boldsymbol{I}_t\rvert^2 \in \mathbb{C}^{\lvert E\rvert}$	Square of *from*/*to* end current magnitude
$\boldsymbol{I}_{max} \in \mathbb{R}^{\lvert E\rvert}$	Limit on line current magnitude
$\hat{\boldsymbol{I}}_{max} \in \mathbb{R}^{\lvert E\rvert}$	Threshold for line tripping
$\tilde{\boldsymbol{Y}}_c = \tilde{\boldsymbol{G}}_c + j\tilde{\boldsymbol{B}}_c \in \mathbb{C}^{\lvert E\rvert}$	line charging
$\tilde{\boldsymbol{Z}} = \tilde{\boldsymbol{Z}}_R + j\tilde{\boldsymbol{Z}}_I \in \mathbb{C}^{\lvert E\rvert}$	line impedance
$\tilde{\boldsymbol{Y}}_L = \tilde{\boldsymbol{G}}_L + j\tilde{\boldsymbol{B}}_L \in \mathbb{C}^{\lvert E\rvert}$	line admittance
$\boldsymbol{V}_{max}/\boldsymbol{V}_{min} \in \mathbb{R}^{\lvert V\rvert}$	Limit on node voltage magnitude
$\boldsymbol{\theta}_{max}/\boldsymbol{\theta}_{min} \in \mathbb{R}^{\lvert E\rvert}$	Limit on phase angle difference for lines
$\hat{\boldsymbol{p}}_3/\hat{\boldsymbol{q}}_3 \in \mathbb{R}^{\lvert V\rvert}$	approximated power injections at t_3
$\hat{\boldsymbol{p}}_{f,3}/\hat{\boldsymbol{q}}_{f,3} \in \mathbb{R}^{\lvert E\rvert}$	approximated line power flow at t_3

Specifically, the first two rows of (3.49) correspond to (3.20e), the next two rows correspond to (3.40), the 5-th row indicates the outage at the target line, the 6-th and 7-th rows correspond to (3.21b), the 8-th and 9-th rows correspond to (3.21a), and the last two rows correspond to (3.20a).

3.7.4 Appendix D: Details of the Attacker's Problem Under AC Power Flow Model

For completeness, we summarize the necessary notations for presenting AC power flow model in Table 3.9. Specifically, we denote $\boldsymbol{C}_f$ as the *From* end incidence matrix, in which $C_{f,e,i} = 1$ if and only if we have $e = (i,k) \in E$. The *To* end incidence matrix $\boldsymbol{C}_t$ is defined similarly, where $C_{t,e,k} = 1$ if and only if we have $e = (i,k) \in E$.

We provide details about (3.38), where we adopt QC relaxation proposed in [96] for (3.38c) and linearized approximation proposed in [95] for (3.38d). As for the constraint on false data injection to bypass BDD (3.38b), we follow [40] to formulate QC relaxation-based constraints.

To begin with, we demonstrate the basics on QC relaxation for AC power flow

equations. Recall from Table 3.9 that the complex voltage on node i is $\vec{v_i} := v_i e^{j\cdot\theta_i}$. Then, we introduce auxiliary variables c_{ii}, c_{ik} and s_{ik} in the hope that

$$c_{ii} = v_i^2, \tag{3.50a}$$

$$c_{ik} = v_i v_k \cos\theta_{ik} \tag{3.50b}$$

$$s_{ik} = v_i v_k \sin\theta_{ik}, \tag{3.50c}$$

where $\theta_{ik} = \theta_i - \theta_k$. As proposed in [96], we first introduce the notation $\langle x\rangle^{\cdot}$, which indicates an auxiliary variable as well as the associated constraints with x as input. Concretely, $\langle x^2\rangle^T$ indicates the auxiliary variable $\breve{x}$ together with the following constraints:

$$\left\langle x^2 \right\rangle^T \equiv \begin{cases} \breve{x} \geqslant x^2 \\ \breve{x} \leqslant (x_u + x_l)\, x - x_u x_l \end{cases}, \tag{3.51}$$

where $x \in [x_l, x_u]$ is pre-assigned bound. Similarly, we have

$$\langle xy\rangle^M := \begin{cases} \breve{xy} \geqslant x_l y + y_l x - x_l y_l \\ \breve{xy} \geqslant x_u y + y_u x - x_u y_u \\ \breve{xy} \leqslant x_l y + y_u x - x_l y_u \\ \breve{xy} \leqslant x_u y + y_l x - x_u y_l \end{cases} \tag{3.52a}$$

$$\langle \sin x\rangle^S := \begin{cases} \breve{sx} \leqslant \cos\left(\frac{x_u}{2}\right)\left(x - \frac{x_u}{2}\right) + \sin\left(\frac{x_u}{2}\right) \\ \breve{sx} \geqslant \cos\left(\frac{x_u}{2}\right)\left(x + \frac{x_u}{2}\right) - \sin\left(\frac{x_u}{2}\right) \end{cases} \tag{3.52b}$$

$$\langle \cos x\rangle^C := \begin{cases} cx \leqslant 1 - \frac{1-\cos(x_u)}{(x_u)^2} x^2 \\ \breve{cx} \geqslant \cos(x_u) \end{cases} \tag{3.52c}$$

Equipped with (3.51) and (3.52), the QC relaxation-based constraints on c_{ii} for each $i \in V$ can be written as $c_{ii} \in \langle v_i^2\rangle^T$, while the constraints on c_{ik} and s_{ik} for each $e = (i,k) \in E$ are

$$c_{ik} = c_{ki}, \tag{3.53a}$$

$$s_{ik} = -s_{ki}, \tag{3.53b}$$

$$c_{ik}^2 + s_{ik}^2 \leq c_{ii} c_{kk}, \tag{3.53c}$$

$$c_{ik} \in \left\langle \langle v_i v_k\rangle^M \cdot \langle \cos\theta_{ik}\rangle^C \right\rangle^M, \tag{3.53d}$$

$$s_{ik} \in \left\langle \langle v_i v_k\rangle^M \cdot \langle \sin\theta_{ik}\rangle^S \right\rangle^M. \tag{3.53e}$$

For simplicity, we will omit the auxiliary variables and the associated constraints for modeling (3.53d) and (3.53e). We assume that (3.53d) and (3.53e) are imposed when QC relaxation is adopted. For (3.38b), the decision variables we focus are $\tilde{c}_{2,ii}, \forall i \in V, \tilde{c}_{2,ik}, \tilde{s}_{2,ik}, \forall e = (i,k) \in E, e = (k,i) \in E$, $\tilde{\boldsymbol{\theta}}_2, \tilde{\boldsymbol{v}}_2$ and $|\tilde{\boldsymbol{I}}_{2,f}|^2, |\tilde{\boldsymbol{I}}_{2,t}|^2$. Then, the constraints (3.38b) can be written as

$$\boldsymbol{\Lambda}_g(\tilde{\boldsymbol{p}}_2 - \boldsymbol{p}_0) = 0, \boldsymbol{\Lambda}_g(\tilde{\boldsymbol{q}}_2 - \boldsymbol{q}_2) = 0 \tag{3.54a}$$

$$\boldsymbol{\Lambda}_g(\tilde{\boldsymbol{v}}_2 - \boldsymbol{v}_2) = \boldsymbol{0}, \tag{3.54b}$$

$$\tilde{c}_{2,ii} = v_{2,i}^2, \forall i \in V_g, \tag{3.54c}$$

$$-\boldsymbol{\Lambda}_d|\tilde{\boldsymbol{p}}_0| \leq \alpha\boldsymbol{\Lambda}_d(\tilde{\boldsymbol{p}}_{2,i} - \tilde{\boldsymbol{p}}_0) \leq \alpha\boldsymbol{\Lambda}_d|\tilde{\boldsymbol{p}}_0|, \tag{3.54d}$$

$$-\boldsymbol{\Lambda}_d|\tilde{\boldsymbol{q}}_0| \leq \alpha\boldsymbol{\Lambda}_d(\tilde{\boldsymbol{q}}_{2,i} - \tilde{\boldsymbol{q}}_0) \leq \alpha\boldsymbol{\Lambda}_d|\tilde{\boldsymbol{q}}_0|, \tag{3.54e}$$

$$(1-\eta)\boldsymbol{V}_{min} \leq \tilde{\boldsymbol{v}}_2 \leq (1+\eta)\boldsymbol{V}_{max} \tag{3.54f}$$

$$(1-\eta)\theta_{min,e} \leq \tilde{\theta}_{2,e} \leq (1+\eta)\theta_{max,e}, \forall e \in E \tag{3.54g}$$

$$|\tilde{\boldsymbol{I}}_{2,f}| \leq \boldsymbol{I}_{max}, |\tilde{\boldsymbol{I}}_{2,t}| \leq \boldsymbol{I}_{max} \tag{3.54h}$$

$$\tilde{p}_{2,i} = \sum_{k=1,\ldots,n} \tilde{G}_{ik}\tilde{c}_{2,ik} - \tilde{B}_{ik}\tilde{s}_{2,ik}, \tag{3.54i}$$

$$\tilde{q}_{2,i} = \sum_{k=1,\ldots,n} -\tilde{B}_{ik}\tilde{c}_{2,ik} - \tilde{G}_{ik}\tilde{s}_{2,ik}, \tag{3.54j}$$

$$\tilde{p}_{2,f,e} = \tilde{G}_{f,e,i}\tilde{c}_{2,ii} + \tilde{G}_{f,e,k}\tilde{c}_{2,ik} - \tilde{B}_{f,e,k}\tilde{s}_{2,ik}, \tag{3.54k}$$

$$\tilde{q}_{2,f,e} = -\tilde{B}_{f,e,i}\tilde{c}_{2,ii} - \tilde{B}_{f,e,k}\tilde{c}_{2,ik} - \tilde{G}_{f,e,k}\tilde{s}_{2,ik}, \tag{3.54l}$$

$$\tilde{p}_{2,t,e} = \tilde{G}^*_{t,e,k}\tilde{c}_{2,kk} + \tilde{G}_{t,e,i}\tilde{c}_{2,ik} + \tilde{B}_{t,e,i}\tilde{s}_{2,ik}, \tag{3.54m}$$

$$\tilde{q}_{2,t,e} = -\tilde{B}^*_{t,e,k}\tilde{c}_{2,kk} - \tilde{B}_{t,e,i}\tilde{c}_{2,ik} + \tilde{G}_{t,e,i}\tilde{s}_{2,ik}, \tag{3.54n}$$

$$\tilde{p}_{2,i} = \boldsymbol{c}_{f,i}^T\tilde{\boldsymbol{p}}_{f,2} + \boldsymbol{c}_{t,i}^T\tilde{\boldsymbol{p}}_{t,2} + \mathcal{R}(Y_{sh,i}\tilde{c}_{2,ii}) \tag{3.54o}$$

$$\tilde{q}_{2,i} = \boldsymbol{c}_{f,i}^T\tilde{\boldsymbol{q}}_{f,2} + \boldsymbol{c}_{t,i}^T\tilde{\boldsymbol{q}}_{t,2} - \mathcal{I}(Y_{sh,i}\tilde{c}_{2,ii}) \tag{3.54p}$$

$$\tilde{c}_{2,ii} = v_{2,i}^2, \tilde{v}_{2,i} = v_{2,i}, \tilde{\theta}_{2,i} = \theta_{2,i}, \forall i \text{ with } x_{N,i} = 1, \tag{3.54q}$$

$$\tilde{c}_{2,ik} = v_{2,i}v_{2,k}\cos\theta_{ik}, \forall e = (i,k) \text{ with } x_{L,e} = 1, \tag{3.54r}$$

$$\tilde{p}_{2,e} = p_{2,e}, \tilde{q}_{2,e} = q_{2,e}, \forall e = (i,k) \text{ with } x_{L,e} = 1, \tag{3.54s}$$

$$\tilde{I}_{2,f,e} = I_{2,f,e}, \tilde{I}_{2,t,e} = I_{2,t,e}, \forall e = (i,k) \text{ with } x_{L,e} = 1, \tag{3.54t}$$

where $\boldsymbol{p}_0$ and $\boldsymbol{q}_0$ indicates the ground-truth power injections at t_0, (3.54i)-(3.54j) are imposed for each node $i \in V$, (3.54k)-(3.54n) are imposed for all $e = (i,k) \in E$, $\boldsymbol{c}_{f,i}/\boldsymbol{c}_{t,i}$ is the i-th column of $\boldsymbol{C}_f/\boldsymbol{C}_t$, $\boldsymbol{Y}_{sh}$ denotes the diagonal matrix of node shunt, $\mathcal{R}(x)/\mathcal{I}(x)$ denotes the real/imaginary part of x, (3.54q)-(3.54t) indicates the protection

effect of PMUs, and $\eta \in [0,1)$ is a manually assigned factor for $\tilde{v}_2$ and $\tilde{\theta}_2$ not to raise alarms in control center. Besides (3.54), we impose the following constraints according to [99, Chapter 5] for each $e=(i,k) \in E$ into (3.38b) :

$$|\tilde{\boldsymbol{I}}_{2,f,e}|^2 = \frac{1}{|Z_e|^2}\left(\tilde{c}_{2,ii}+\tilde{c}_{2,kk}-2\tilde{c}_{2,ik}\right)+2\tilde{G}_{c,e}\tilde{p}_{2,f,e}-2\tilde{B}_{c,e}\tilde{q}_{2,f,e}-|\boldsymbol{Y}_{c,e}|^2\,\tilde{c}_{2,ii}, \tag{3.55a}$$

$$\begin{aligned}\tilde{p}_{2,f,e}+\tilde{q}_{2,f,e} = \tilde{Z}_{R,e}\Big(&|\tilde{\boldsymbol{I}}_{2,f,e}|^2-2(\tilde{G}_{c,e}\tilde{p}_{2,f,e}-\tilde{B}_{c,e}\tilde{q}_{2,f,e})\\ &+|\boldsymbol{Y}_{c,e}|^2\,\tilde{c}_{2,ii}\Big)+\tilde{G}_{c,e}(\tilde{c}_{2,ii}+\tilde{c}_{2,kk}),\end{aligned} \tag{3.55b}$$

$$\begin{aligned}\tilde{p}_{2,f,e}+\tilde{q}_{2,f,e} = \tilde{Z}_{I,e}\Big(&|\tilde{\boldsymbol{I}}_{2,f,e}|^2-2(\tilde{G}_{c,e}\tilde{p}_{2,f,e}-\tilde{B}_{c,e}\tilde{q}_{2,f,e})\\ &+|\boldsymbol{Y}_{c,e}|^2\,\tilde{c}_{2,ii}\Big)-\tilde{B}_{c,e}(\tilde{c}_{2,ii}+\tilde{c}_{2,kk}),\end{aligned} \tag{3.55c}$$

$$\begin{aligned}&\left(1+2\tilde{Z}_{R,e}\tilde{G}_{c,e}-2\tilde{Z}_{I,e}\tilde{B}_{c,e}\right)\tilde{c}_{2,ii}-\tilde{c}_{2,kk}=2(\tilde{Z}_{R,e}\tilde{p}_{2,f,e}\\ &+\tilde{Z}_{I,e}\tilde{q}_{2,f,e})-|\tilde{Z}_e|^2\Big(|\tilde{\boldsymbol{I}}_{2,f,e}|^2-2(\tilde{G}_{c,e}\tilde{p}_{2,f,e}-\tilde{B}_{c,e}\tilde{q}_{2,f,e})\\ &+\left|\tilde{Y}_{c,e}\right|^2\tilde{c}_{2,ii}\Big)\end{aligned} \tag{3.55d}$$

All equations in (3.55) should hold simultaneously.

Similarly, the decision variables we will focus on in (3.38c) are $\tilde{c}_{3,ii}, \forall i \in V, \tilde{3}_{2,ik}, \tilde{s}_{3,ik}, \forall e = (i,k) \in E, e=(k,i) \in E$, $\tilde{\boldsymbol{\theta}}_3, \tilde{\boldsymbol{v}}_3$ and $|\tilde{\boldsymbol{I}}_{3,f}|^2, |\tilde{\boldsymbol{I}}_{3,t}|^2$. Then, the constraints (3.38c) are similar to (3.54) and (3.55), with (3.54a)-(3.54h) changed into

$$\boldsymbol{p}_{g,min} \le \boldsymbol{\Lambda}_g\tilde{\boldsymbol{p}}_3 \le \boldsymbol{p}_{g,max}, \boldsymbol{q}_{g,min} \le \boldsymbol{\Lambda}_g\tilde{\boldsymbol{q}}_3 \le \boldsymbol{q}_{g,max}, \tag{3.56a}$$

$$\boldsymbol{\Lambda}_d(\tilde{\boldsymbol{p}}_{3,i}-\tilde{\boldsymbol{p}}_{2,i})=0, \quad \boldsymbol{\Lambda}_d(\tilde{\boldsymbol{q}}_{3,i}-\tilde{\boldsymbol{q}}_{2,i})=0, \tag{3.56b}$$

$$\boldsymbol{\Lambda}_g(\tilde{\boldsymbol{p}}_{3,i}-\tilde{\boldsymbol{p}}_{2,i})=0 \tag{3.56c}$$

$$\boldsymbol{V}_{min} \le \tilde{\boldsymbol{v}}_3 \le \boldsymbol{V}_{max}, \theta_{min,e} \le \tilde{\theta}_{3,e} \le \theta_{max,e}, \forall e \in E, \tag{3.56d}$$

$$|\tilde{\boldsymbol{I}}_{3,f}| \le \boldsymbol{I}_{max}, |\tilde{\boldsymbol{I}}_{3,t}| \le \boldsymbol{I}_{max}. \tag{3.56e}$$

Following [95], the decision variables in (3.38d) are $\hat{v}_{3,i}^2, \hat{\theta}_{3,i}, \hat{p}_{3,i}, \hat{q}_{3,i}, \forall i \in V$, $\hat{\boldsymbol{p}}_{f,3} \in \mathbb{R}^{|E|}, \hat{\boldsymbol{q}}_{f,3} \in \mathbb{R}^{|E|}$ and $|\hat{\boldsymbol{I}}_3|^2 \in \mathbb{R}^{|E|}$. Next, we define $p_{f,3,e}^L$ and $q_{f,3,e}^L$ for $e=(i,k) \in E$ with $a_{p,e}=0$ as follows:

$$p_{f,3,e}^L = \tilde{G}_{L,e}\left(\hat{\theta}_{ik,0}\hat{\theta}_{3,ik}-\frac{\hat{\theta}_{ik,0}^2}{2}+\frac{\hat{v}_{i,0}-\hat{v}_{k,0}}{\hat{v}_{i,0}+\hat{v}_{k,0}}(\hat{v}_{3,i}^2-\hat{v}_{3,k}^2)-\frac{(\hat{v}_{i,0}-\hat{v}_{k,0})^2}{2}\right)+\mathcal{R}(\tilde{Y}_{c,e})\hat{v}_{3,i}^2 \tag{3.57a}$$

$$q^L_{f,3,e} = -\tilde{B}_{L,e}\left(\hat{\theta}_{ik,0}\hat{\theta}_{3,ik} - \frac{\hat{\theta}^2_{ik,0}}{2} + \frac{\hat{v}_{i,0}-\hat{v}_{k,0}}{\hat{v}_{i,0}+\hat{v}_{k,0}}(\hat{v}^2_{3,i}-\hat{v}^2_{3,k}) - \frac{(\hat{v}_{i,0}-\hat{v}_{k,0})^2}{2}\right) - \mathcal{I}(\tilde{Y}_{c,e})\hat{v}^2_{3,i}, \tag{3.57b}$$

where $\hat{v}_{ik,0}$ and $\hat{\theta}_{ik,0}$ are obtained from any base case system operating condition. In our work, we set it as $\hat{v}_{ik,0} = v_{2,ik}$ and $\hat{\theta}_{ik,0} = \theta_{2,ik}$ for each given $(\boldsymbol{a}_p, e_t)$. Then, we have three types of constraints in (3.38d). Specifically, by appropriately setting $\eta_{3,p,i}$ and $\eta_{3,q,i}$ (see proof of Theorem 3.4.1 for details) to tolerate the approximation error, for each $i \in V$, we have

$$-\eta_{3,p,i} \leq \boldsymbol{D}_i\hat{\boldsymbol{p}}_{3,f} + \hat{v}^2_{3,i}\sum_{k=1}^{|V|}\tilde{G}_{ik} - p_{0,i} \leq \eta_{3,p,i}. \tag{3.58}$$

For each $i \in V_d$, we have

$$-\eta_{3,q,i} \leq \boldsymbol{D}_i\hat{\boldsymbol{q}}_{3,f} - \hat{v}^2_{3,i}\sum_{k=1}^{|V|}\tilde{B}_{ik} - \tilde{q}_{3,i} \leq \eta_{3,q,i},. \tag{3.59}$$

For each $e = (i,k) \in E$ with $a_{p,e} = 0$, we have

$$p_{f,3,e} = \tilde{G}_{L,e}\frac{\hat{v}^2_{3,i}-\hat{v}^2_{3,k}}{2} - \tilde{B}_{L,e}\hat{\theta}_{ik} + p^L_{f,3,e}, \tag{3.60a}$$

$$q_{f,3,e} = -\tilde{B}_{L,e}\frac{\hat{v}^2_{3,i}-\hat{v}^2_{3,k}}{2} - \tilde{G}_{L,e}\hat{\theta}_{ik} + q^L_{f,3,e}, \tag{3.60b}$$

$$\begin{aligned}\left(1 + 2\tilde{Z}_{R,e}\tilde{G}_{c,e} - 2\tilde{Z}_{I,e}\tilde{B}_{c,e}\right)\hat{v}^2_{3,i} - \hat{v}^2_{3,k} = 2(\tilde{Z}_{R,e}\hat{p}_{3,f,e} \\ + \tilde{Z}_{I,e}\hat{q}_{3,f,e}) - |\tilde{Z}_e|^2\left(|\hat{\boldsymbol{I}}_{3,f,e}|^2 - 2(\tilde{G}_{c,e}\hat{p}_{3,f,e} \\ - \tilde{B}_{c,e}\hat{q}_{3,f,e}) + \left|\tilde{Y}_{c,e}\right|^2\hat{v}^2_{3,i}\right)\end{aligned} \tag{3.60c}$$

3.7.5 Appendix E: Additional Proofs

Theorem 3.3.1. We will reduce the dominating set problem to PPOP. Given a graph $\mathcal{N} = (V, E)$, the dominating set problem aims to find a minimum set of vertices $V_1 \in V$ such that $\forall u \in V \setminus V_1$, u has at least one neighbor in V_1. The dominating set problem is known to be NP-hard. Notice that given the grid $\mathcal{N} = (V, E)$ the parameters for the proposed problem (3.24)-(3.23) are $\boldsymbol{p}_0, \boldsymbol{\Gamma}, \xi_p, \xi_c, \boldsymbol{\alpha}$ and $\{\gamma_e\}_{e\in E}$. We will prove for any given connected grid and the associated dominating set problem, there exists a parameter

setting for the proposed problem such that V_1 is a minimal dominating set if and only if V_1 is an optimal solution to (3.24), i.e., $\forall u \in V, x_{N,u} = 1$.

Given any $\boldsymbol{p}_0$, suppose that $\boldsymbol{\theta}_0$ is the associated phase angle without attack, i.e., $\boldsymbol{p}_0 = \tilde{\boldsymbol{B}}\boldsymbol{\theta}_0$, and $\hat{\boldsymbol{\theta}}_0$ is the the solution to (3.5), i.e., $\hat{\boldsymbol{\theta}}_0 = \psi_s(\boldsymbol{p}_0, \tilde{\boldsymbol{D}})$, which gives $\hat{\boldsymbol{f}}_0 := \boldsymbol{\Gamma}\tilde{\boldsymbol{D}}^T\hat{\boldsymbol{\theta}}_0$.

Then, we set $\boldsymbol{p}_0 = \boldsymbol{0}$, $\xi_p = 0$, $\xi_c = \infty$, $\alpha = \infty$ and $\boldsymbol{\Gamma}$ as identity matrix, which results in $\boldsymbol{\theta}_0 = \hat{\boldsymbol{\theta}}_0 = \boldsymbol{0}$ and $\hat{\boldsymbol{f}}_0 = \boldsymbol{0}$. In addition, we set $\gamma_e = 0, \forall e \in E$, which transform (3.23e) to

$$|\Gamma_e \boldsymbol{d}_e^T \boldsymbol{\theta}_3| = 0 \leftrightarrow \pi_e = 0. \tag{3.61}$$

Next, we show by contradiction that $|\Gamma_e \boldsymbol{d}_e^T \boldsymbol{\theta}_3| = 0$ holds for all $e \in E$ only if $\tilde{\boldsymbol{\theta}}_2 = \boldsymbol{0} = \boldsymbol{\theta}_0$. Suppose $\tilde{\boldsymbol{\theta}}_2 \neq \boldsymbol{0}$, we must have $\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_2 \neq \boldsymbol{0}$, which leads to $\boldsymbol{0} \neq \tilde{\boldsymbol{\theta}}_3 = \psi_s(\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_2, \tilde{\boldsymbol{D}})$ and thus $\boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3 \neq \boldsymbol{0}$ due to the constraint (3.23c). The non-zero $\boldsymbol{\Lambda}_g\tilde{\boldsymbol{B}}\tilde{\boldsymbol{\theta}}_3$ implies that $\exists e \in E$ such that $\Gamma_e \boldsymbol{d}_e^T \boldsymbol{\theta}_3 \neq 0$. That is to say, the constraint (3.24b) holds only when $\tilde{\boldsymbol{\theta}}_2 = \boldsymbol{\theta}_0 = \boldsymbol{0}$, which indicates that the defender has to place PMUs to guarantee that the only feasible solution to (3.23) is $\boldsymbol{a}_c = \boldsymbol{0}$. In another word, $\boldsymbol{\beta}$ needs to satisfy $\forall u \in V, x_{N,u} = 1$, which completes the proof. □

Theorem 3.3.3. First, we introduce some definitions: $\mathcal{B} := \{\boldsymbol{\beta} | \psi_a(\boldsymbol{\beta}) = 0\}$ denotes the set of feasible solutions, $\mathcal{B}^c := \{\boldsymbol{\beta} | \psi_a(\boldsymbol{\beta}) \geq 1\}$ the infeasible solutions, $\mathcal{M}(\mathcal{B}^c) := \{\boldsymbol{\beta} | (\boldsymbol{\beta}, \boldsymbol{\beta}' \in \mathcal{B}^c) \wedge (\boldsymbol{\beta}' \geq \boldsymbol{\beta}) \rightarrow (\boldsymbol{\beta}' = \boldsymbol{\beta})\}$ the maximal infeasible solutions, and $\mathcal{P} := \{\check{\boldsymbol{\beta}} \in [0,1]^{|V|} | \forall \boldsymbol{\beta} \in \mathcal{M}(\mathcal{B}^c) : \sum_{u:\beta_u=0} \check{\beta}_u \geq 1\}$ the polytope excluding all the maximal infeasible solutions.

Then, based on the results in [100], we have the following characterization:

Lemma 3.7.1. *The following statements hold: (i) $\mathcal{P} \cap \{0,1\}^{|V|} = \mathcal{B}$; (ii) $\forall \boldsymbol{\beta}' \in \mathcal{M}(\mathcal{B}^c)$, $\sum_{u:\beta'_u=0} \beta_u \geq 1$ defines a facet of $\mathcal{P}$.*

Proof. To prove statement (i), we first prove that $\mathcal{B} \subseteq (\mathcal{P} \cap \{0,1\}^{|V|})$ by contradiction. Suppose $\exists \boldsymbol{\beta}_1 \in \mathcal{B}$ but $\boldsymbol{\beta}_1 \notin \mathcal{P}$. Then by definition of $\mathcal{P}$, there must exist $\boldsymbol{\beta}'_1 \in \mathcal{B}^c$ such that $\sum_{u:\beta'_{1,u}=0} \beta_{1,u} = 0$, which implies $\Omega(\boldsymbol{\beta}_1) \subseteq \Omega(\boldsymbol{\beta}'_1)$. By Lemma 3.3.2, we must have $\boldsymbol{\beta}_1 \in \mathcal{B}^c$, which contradicts with the assumption that $\boldsymbol{\beta}_1 \in \mathcal{B}$. Thus, $\mathcal{B} \subseteq (\mathcal{P} \cap \{0,1\}^{|V|})$. Then, we prove $(\mathcal{P} \cap \{0,1\}^{|V|}) \subseteq \mathcal{B}$ by contradiction. Suppose there exists $\check{\boldsymbol{\beta}} \in (\mathcal{P} \cap \{0,1\}^{|V|})$ but $\check{\boldsymbol{\beta}} \notin \mathcal{B}$, which implies that $\check{\boldsymbol{\beta}} \in \mathcal{B}^c$. That is to say, $\exists \check{\boldsymbol{\beta}}' \geq \check{\boldsymbol{\beta}}$ such that $\check{\boldsymbol{\beta}}' \in \mathcal{M}(\mathcal{B}^c)$. Then by definition of $\mathcal{P}$, we have $\sum_{u:\check{\beta}'_u=0} \check{\beta}_u \geq 1$. However, since $\check{\boldsymbol{\beta}}' \geq \check{\boldsymbol{\beta}}$, $\forall u : \check{\beta}_u = 0$, we must have $\check{\beta}_u = 0$ and leads to $\sum_{u:\check{\beta}'_u=0} \check{\beta}_u = 0$, which introduces contradiction. In summary, $\mathcal{P} \cap \{0,1\}^{|V|} = \mathcal{B}$.

We then prove statement (ii) by contradiction, i.e., $\exists \check{\boldsymbol{\beta}}' \in \mathcal{M}(\mathcal{B}^c)$ such that when we remove the inequality $\sum_{u:\check{\beta}'_u=0} \beta_u \geq 1$ from $\mathcal{P}$, we still have $\mathcal{P}$. By definition of $\mathcal{M}(\mathcal{B}^c)$, we must have $\check{\boldsymbol{\beta}}' \in \mathcal{B}^c$, which implies $\sum_{u:\check{\beta}'_u=0} \check{\beta}'_u = 0$, i.e., $\check{\boldsymbol{\beta}}' \notin \mathcal{P}$. That is to say, there exists some inequality to cut $\check{\boldsymbol{\beta}}'$ out from $\mathcal{P}$, i.e., $\exists \check{\boldsymbol{\beta}}'' \in \mathcal{M}(\mathcal{B}^c)$ and $\check{\boldsymbol{\beta}}'' \neq \check{\boldsymbol{\beta}}'$ such that $\sum_{u:\check{\beta}''_u=0} \check{\beta}'_u = 0$. Notice that $\forall u : (\check{\beta}''_u = 0) \rightarrow (\check{\beta}'_u = 0)$, which implies $\Omega(\check{\boldsymbol{\beta}}') \subseteq \Omega(\check{\boldsymbol{\beta}}'')$. By definition of $\mathcal{M}(\mathcal{B}^c)$, we must have $\check{\beta}''_u = \check{\beta}'_u$, which contradicts with $\check{\beta}''_u \neq \check{\beta}'_u$ and completes the proof. □

We now prove Theorem 3.3.3 based on Lemma 3.7.1. First notice that each $\hat{\boldsymbol{\beta}} \in \mathcal{B}^c$ will be enumerated at most once in Alg. 4 due to the "no-good" constraints, and hence the algorithm will converge in finite time. Then, consider an arbitrary $\hat{\boldsymbol{\beta}}'$ obtained through (3.27). The generated "no-good" constraint $\sum_{i:\hat{\beta}'_i=0} \beta_i \geq 1$ must be satisfied by all the feasible solutions in $\mathcal{B}$, as any PMU placement violating this constraint must be infeasible according to Lemma 3.3.2. Finally, for any $\boldsymbol{\beta}_1, \boldsymbol{\beta}_2 \in \mathcal{B}$ with $\|\boldsymbol{\beta}_1\|_0 < \|\boldsymbol{\beta}_2\|_0$, $\boldsymbol{\beta}_1$ will be found by Alg. 4 before $\boldsymbol{\beta}_2$, since each guess of PMU placement is obtained by minimizing $\|\boldsymbol{\beta}\|_0$ in (3.24), which completes the proof. □

Theorem 3.3.4. As Alg. 4 always returns a feasible solution that defends against all attack pairs, we only need to prove that the solution $\boldsymbol{\beta}_1$ returned by AODC requires the minimum number of PMUs. We will prove this by contradiction. Suppose that there exists $\boldsymbol{\beta}_2$ such that $\|\boldsymbol{\beta}_2\|_0 < \|\boldsymbol{\beta}_1\|_0$ and $\psi_a(\boldsymbol{\beta}_2) = 0$. Then $\boldsymbol{\beta}_2$ must be feasible to the instance of (3.33) constructed based on the attack pairs $\{(\boldsymbol{a}_p^{(k)}, e^{(k)})\}_{k=1}^K$ and the maximal infeasible solutions $\{\hat{\boldsymbol{\beta}}'^{(k)}\}_{k=1}^K$ found by AODC as it defends against all attacks. This contradicts with the fact that $\boldsymbol{\beta}_1$ is optimal to (3.33). □

Lemma 3.3.3. We first observe that $\boldsymbol{x}_N$ and $\boldsymbol{x}_L$ are unique under the constraints (3.16)-(3.17). Thus, we will use $\boldsymbol{x}_N(\boldsymbol{\beta})$ and $\boldsymbol{x}_L(\boldsymbol{\beta})$ to denote the values of $\boldsymbol{x}_N$ and $\boldsymbol{x}_L$ satisfying (3.16)-(3.17) for a given $\boldsymbol{\beta} \in \{0,1\}^{|V|}$.

For a given attack pair $(\boldsymbol{a}_p, e)$, $(\check{\boldsymbol{q}}_1, \check{\boldsymbol{q}}_2, \check{\boldsymbol{\beta}})$ can be feasible to (3.36) in two different cases. The first case is that

$$\sum_{a_{p,e}=1} x_{L,e}(\lceil \check{\boldsymbol{\beta}} \rceil) \geq 1, \tag{3.62}$$

which makes $(\boldsymbol{q}_1 = \boldsymbol{0}, \boldsymbol{q}_2 = \boldsymbol{0}, \lceil \check{\boldsymbol{\beta}} \rceil, \boldsymbol{x}_N(\lceil \check{\boldsymbol{\beta}} \rceil), \boldsymbol{x}_L(\lceil \check{\boldsymbol{\beta}} \rceil))$ feasible for (3.32) with $w_a = 1$.

The second case is that $x_{L,e}(\lceil \check{\boldsymbol{\beta}} \rceil) = 0$ for all e with $a_{p,e} = 1$, in which case we must have $(\check{\boldsymbol{q}}_1, \check{\boldsymbol{q}}_2, \lceil \check{\boldsymbol{\beta}} \rceil, \boldsymbol{x}_N(\lceil \check{\boldsymbol{\beta}} \rceil), \boldsymbol{x}_L(\lceil \check{\boldsymbol{\beta}} \rceil))$ feasible to (3.32) with $w_a = 0$. To prove this, we

only need to show that

$$\left(\boldsymbol{F}_3\boldsymbol{x}_N(\lceil\check{\boldsymbol{\beta}}\rceil)\right)^T\check{\boldsymbol{q}}_2 \leq \boldsymbol{F}_3\check{\boldsymbol{q}}_2. \tag{3.63}$$

According to (3.49), $F_{3,i,u}$ is either 0 or $-M_\theta$, which together with the fact that $\boldsymbol{x}_{N,u}(\lceil\check{\boldsymbol{\beta}}\rceil) \geq 0$ and $\check{q}_{2,i} \geq 0$ implies that

$$\left(\boldsymbol{F}_3\boldsymbol{x}_N(\lceil\check{\boldsymbol{\beta}}\rceil)\right)^T\check{\boldsymbol{q}}_2 = \sum_{u\in V}\boldsymbol{x}_{N,u}(\lceil\check{\boldsymbol{\beta}}\rceil)\left(\sum_{i=1}^{m_2}F_{3,i,u}\check{q}_{2,i}\right) \tag{3.64}$$

$$\leq \sum_{u\in V}\mathbf{1}\left(\sum_{i=1}^{m_2}F_{3,i,u}\check{q}_{2,i}\right) = \boldsymbol{F}_3\check{\boldsymbol{q}}_2, \tag{3.65}$$

which completes the proof. □

Theorem 3.3.5. Under the assumption of $\xi_p = \mathcal{O}(1)$, the number of possible attack pairs is $|E|\left(\sum_{i=1}^{\xi_p}\binom{|E|}{i}\right) \leq \xi_p|E|^{\xi_p+1} = \mathcal{O}\left(|E|^{\xi_p+1}\right)$. Therefore, the time complexity of solving (3.23) for a given $\boldsymbol{\beta}$ is polynomial in $|E|$ and $|V|$, since in the worst case (3.23) can be solved by checking the feasibility of (3.30) for all the $\mathcal{O}\left(|E|^{\xi_p+1}\right)$ attack pairs.

We first characterize the complexity of Alg. 6. Since each candidate placement Ω_i either has one more node or can defend against all attack pairs in $\mathcal{A}$ after one iteration of the while loop, Alg. 6 converges within $|V|$ iterations. Each iteration of Alg. 6 is dominated by solving (3.37) (Line 8) for at most K_c times. Since the numbers of variables and constraints of (3.37) are both $\mathcal{O}((|E|+|V|)|\mathcal{A}|)$ and $|\mathcal{A}| = \mathcal{O}\left(|E|^{\xi_p+1}\right)$, the complexity of solving (3.37) is polynomial [4] in $|V|$ and $|E|$. In summary, the complexity of Alg. 6 is polynomial in $|V|$, $|E|$, and K_c since it solves a polynomial-sized LP for at most $K_c|V|$ times. It is worth noting that the effect of K_A and K_L in Alg. 6's complexity is dominated by $|V|$ and $|E|$. To see this, we note that K_L only appears in Line 7 of Alg. 6, in which we must have $K_L \leq |E|$. Then, K_A only appears in Line 9 of Alg. 6, in which we must have $K_A \leq |V|$. Thus, we do not consider the effect of K_A and K_L in Alg. 6's computational complexity.

The complexity of Alg. 5 comes from: (i) solving (3.23) $\mathcal{O}(|E|^{\xi_p+1})$ times (Line 3 and Line 12); (ii) solving (3.37) for $|\mathcal{A}_0| = \mathcal{O}(|E|^{\xi_p+1})$ times (Line 5), each of which deals with an LP containing $\mathcal{O}((|E|+|V|)|\mathcal{A}_0|)$ variables and constraints and thus takes polynomial time; (iii) calling Alg. 6 at Line 8 for 1 time and at Line 14 for $\mathcal{O}(|E|^{\xi_p+1})$ times, whose complexity is polynomial in $|V|$, $|E|$, and K_c. In summary, Alg. 5 is a polynomial-time algorithm in terms of $|V|$, $|E|$, and K_c. □

[4] The exact order depends on the specific algorithm used to solve LP [53].

Theorem 3.4.1. According to [95, 99] and (3.60), we have

$$|\hat{I}_{3,f,e}|^2 = \frac{1}{|Z_e|^2}\Big(2(\tilde{Z}_{R,e}\hat{p}_{3,f,e} + \tilde{Z}_{I,e}\hat{q}_{3,f,e}) + \hat{v}_k^2 - (1 + 2\tilde{Z}_{R,e}\tilde{G}_{c,e} - 2\tilde{Z}_{I,e}\tilde{B}_{c,e})\hat{v}_i^2\Big) + 2(\tilde{G}_{c,e}\hat{p}_{3,f,e} - \tilde{B}_{c,e}\hat{q}_{3,f,e}) - |\tilde{Y}_{c,e}|^2\hat{v}_i^2 \tag{3.66}$$

for each $e = (i,k) \in E$ with $a_{p,e} = 0$. Based on (3.66) and the assumption on $\boldsymbol{\epsilon_\theta} = (\epsilon_{\theta,u})_{u\in V}$, $\boldsymbol{\epsilon_v} = (\epsilon_{v,u})_{u\in V}$, $\boldsymbol{\epsilon_p} = (\epsilon_{p,e})_{e\in E}$ and $\boldsymbol{\epsilon_q} = (\epsilon_{q,e})_{e\in E}$, we can easily derive an upperbound $\epsilon_{I,e} \geq ||\hat{I}_{3,e}| - |I_{3,e}||, \forall e \in E$. Specifically, we can set

$$\epsilon_{I,e} := \frac{1}{|Z_e|^2}\Big(2(|\tilde{Z}_{R,e}|\epsilon_{p,e} + |\tilde{Z}_{I,e}|\epsilon_{q,e}) + \epsilon_{v,i}^2 + |1 + 2\tilde{Z}_{R,e}\tilde{G}_{c,e} + 2\tilde{Z}_{I,e}\tilde{B}_{c,e}|\epsilon_{v,i}^2\Big) + 2(|\tilde{G}_{c,e}|\epsilon_{p,e} + |\tilde{B}_{c,e}|\epsilon_{q,e}) + |\tilde{Y}_{c,e}|^2\epsilon_{v,i}^2. \tag{3.67}$$

If there exists an successful attack pair $(\boldsymbol{a}_p, e)$ that cannot be found by Alg. 7 for a given PMU placement, we must have one of the following cases:

1. There exists $\tilde{\boldsymbol{v}}_2, \tilde{\boldsymbol{\theta}}_2$ such that $|I_{3,e}| > \gamma_e I_{max,e}$. In the meantime, at least one of (3.58) and (3.59) are violated.

2. Let $|\hat{I}^*_{3,f,e}|$ be the optimal solution of (3.38). There exists $\tilde{\boldsymbol{v}}_2^{(1)}, \tilde{\boldsymbol{\theta}}_2^{(1)}, \tilde{\boldsymbol{v}}_3^{(1)}, \tilde{\boldsymbol{\theta}}_3^{(1)}$ such that $|I_{3,e}^{(1)}| > \gamma_e I_{max,e}$. Let $|\hat{I}_{3,f,e}^{(1)}|$ be the corresponding approximated solution for $\tilde{\boldsymbol{v}}_2^{(1)}, \tilde{\boldsymbol{\theta}}_2^{(1)}, \tilde{\boldsymbol{v}}_3^{(1)}, \tilde{\boldsymbol{\theta}}_3^{(1)}$. Then we must have $\hat{I}_{max,e} \geq |\hat{I}^*_{3,f,e}| \geq |\hat{I}^{(1)}_{3,f,e}|$.

We first show that the case one can be avoided if we properly set $\eta_{3,p,i}$ in (3.58) and $\eta_{3,q,i}$ in (3.59). Specifically, according to (3.58), we must have

$$\boldsymbol{D}_i\hat{\boldsymbol{p}}_{3,f} + \hat{v}_{3,i}^2 \sum_{k=1}^{|V|} \tilde{G}_{ik} - p_{0,i} \leq \eta_{3,p,i} \tag{3.68}$$

if we set

$$\eta_{3,p,i} \geq (\Delta_{ii} - 1)\epsilon_{p,i} + |\sum_{k=1}^{|V|} \tilde{G}_{ik}|\epsilon_{v,i}, \tag{3.69}$$

where $(\Delta_{ii} - 1)$ denotes the number of neighbors of node i as defined in (3.16). Similarly, we can define $\eta_{3,q,i}$ to avoid the first case. Then, we will show how to set $\hat{I}_{max,e}$ so that

the second case will not happen. In case two, we must have

$$\hat{I}_{max,e} \geq |\hat{I}^{*}_{3,f,e}| \geq |\hat{I}^{(1)}_{3,f,e}| \geq |I^{(1)}_{3,e}| - \epsilon_{I,e} > \gamma_e I_{max,e} - \epsilon_{I,e} \tag{3.70}$$

Thus, if we set $\hat{I}_{max,e} \leq \gamma_e I_{max,e} - \epsilon_{I,e}$, (3.70) cannot hold, which rules out the possibility of case two. In summary, by properly setting $\eta_{3,p,i}, \eta_{3,q,i}$ and set $\hat{I}_{max,e} \leq \gamma_e I_{max,e} - \epsilon_{I,e}$, a PMU placement that can pass the test of Alg. 7 will achieve our defense goal. □

Chapter 4
Overlay Routing Over an Uncooperative Underlay

4.1 Introduction

Overlay networks, referring to logical distributed systems running on top of a physical communication underlay, have been widely adopted to enhance the existing network infrastructure due to the difficulty of deploying infrastructure-wide upgrades. Frequently, overlay networks are used to provide value-adding functionalities that a best-effort IP-based underlay network cannot provide, such as caching, traffic engineering, fast failover, and attack mitigation [101]. Meanwhile, the performance of an overlay network heavily relies on the proper control of overlay routing. For instance, caching overlay requires efficient routing between origin servers and edge servers to provide notable performance gain in the case of cache misses [101]. Large-scale applications spreading across multiple datacenters need careful routing of inter-datacenter flows to avoid congestion [102]. For mission-critical overlay applications, a proper selection of backup routes between overlay nodes that are maximally disjoint with primary routes is necessary for maintaining high Quality of Service (QoS) in the case of failures [103].

Due to its importance, tremendous efforts have been devoted to the design of overlay routing, e.g., [101–104]. Compared to classical routing problems, one of the unique challenges in overlay routing is the lack of knowledge about the underlay, which can lead to incorrect overlay routing decisions. As a concrete example, consider the overlay-underlay network in Fig. 4.1, where link labels denote their (propagation) delays, and each overlay link maps to the shortest path (in delay) between its endpoints in the underlay. Suppose that the overlay needs to route two large flows with source-destination pairs (a, e) and (b, d), respectively. Further suppose that each link in the underlay has

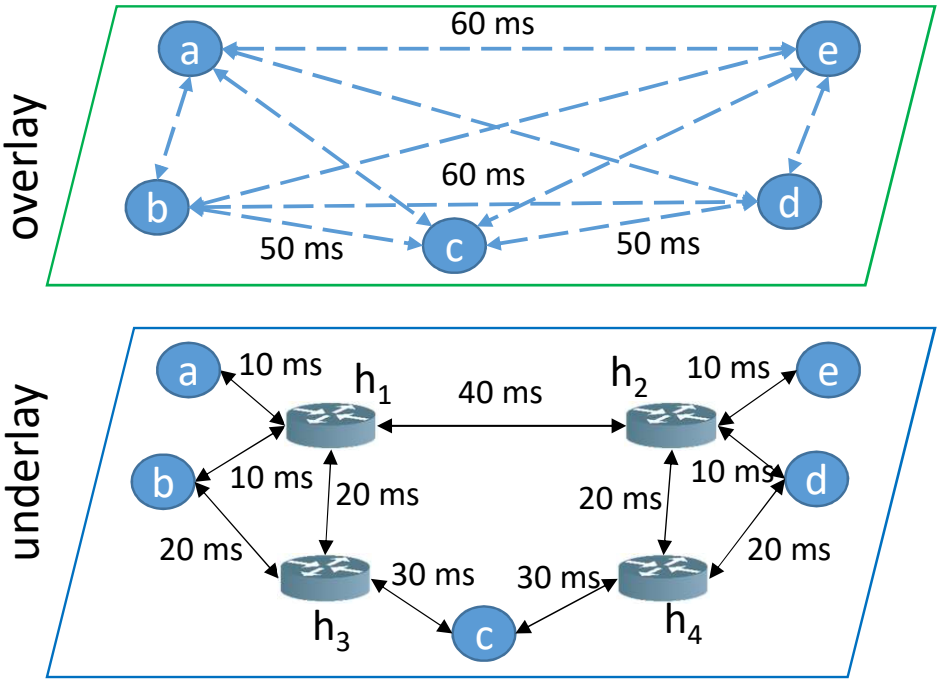


Figure 4.1. Example of underlay-aware overlay routing.

sufficient capacity for one of the flows but not both. Given the objective of minimizing the total delay, an underlay-agnostic routing algorithm that is only aware of the individual delays and capacities of overlay links will route both flows over the direct overlay paths $a \to e$ and $b \to d$. However, as these paths share a common link (h_1, h_2) in the underlay, this routing solution will cause congestion and hence poor performance. Meanwhile, an underlay-aware routing algorithm that has knowledge of how the overlay links share links in the underlay will choose the overlay paths $a \to e$ and $b \to c \to d$, which will minimize the total delay while avoiding congestion.

The need for overlay routing to be aware of the internal parameters of the underlay (e.g., topology, routing protocol, link characteristics) has been widely recognized. However, most of the existing works either assume such information to be directly provided by the underlay [105, 106], or avoid explicitly requiring such knowledge by performing overlay routing on a trial-and-error basis [103]. The former approach is often inapplicable in practice due to the lack of cooperation from the underlay, and the latter approach is inefficient due to the exponentially large search space.

In this work, we aim at addressing these limitations by developing a framework for *underlay-aware overlay routing* that can systematically optimize the routing among overlay nodes without cooperation from the underlay. The core of our framework is a set

of network inference algorithms that can extract the necessary information about the underlay from measurements within the overlay to enable overlay route optimization while avoiding congestion.

4.1.1 Related Work

Overlay routing: Overlay routing aims at controlling data forwarding among overlay nodes to optimize certain performance metrics while avoiding congestion. Typical performance metrics include routing cost [104, 107], route update cost [105], and completion time of peer-to-peer data distribution [108] or inter-datacenter data transfer [102, 106]. Most of these works either assumed a cooperative underlay network whose internal parameters can be directly observed by the overlay [105, 106, 108], or ignored the sharing of underlay links by the logical links between overlay nodes [102, 107]. *In contrast, we address the more challenging problem of overlay routing over an uncooperative underlay network, while accounting for the underlay link sharing between overlay links.*

The works most related to ours are [109, 110], which encoded the knowledge about the underlay as *linear capacity constraints (LCCs)*. Intuitively, each LCC, with a form similar to (4.1b), ensures that the total traffic load from the overlay does not exceed the capacity of an underlay link. Thus, the complete set of LCCs contains all the information an overlay needs to optimize its routing without incurring congestion. However, [109, 110] only focused on overlay routing based on given LCCs, leaving the challenging problem of inferring LCCs to a simple heuristic based on an existing technique for *shared bottleneck detection (SBD)* [111]. This heuristic was vulnerable to the error of SBD and could only discover the LCCs corresponding to the bottlenecks shared by tunnels ending at the same overlay node. The resulting LCCs were thus incomplete and inaccurate, causing suboptimal routing decisions in the overlay. *In this work, we address these issues by developing algorithms that can infer the minimum necessary set of LCCs from observations at the overlay with guaranteed accuracy.*

Network (topology) tomography: One key piece of information for routing is network topology. In the face of an uncooperative underlay, the overlay can try to infer its topology from end-to-end measurements on the underlay routing paths between overlay nodes, known as *network topology inference/tomography* [112].

Although extensively studied, topology inference is far from being completely solved. Earlier works tried to construct a tree topology based on various measurements from multicast probes [113–115] or unicast probes [116–118] sent by a single source. Later works aimed at combining measurements from multiple sources [119–122], but still made

the assumption that the routing paths for each source/destination form a tree. See [123] for a more detailed summary. The assumption of tree-based routing is frequently violated due to round-trip probing, load balancing, and network function traversals, but removing this assumption significantly complicates topology inference, for which only a few results exist [123–125]. Without the assumption of tree-based routing, the routing topology can no longer be uniquely identified from end-to-end measurements [123]. However, it is still possible to detect the existence of links shared by a subset of paths [123, 125]. Although not enough for identifying the topology, the information about which subsets of paths (i.e., overlay links) share links in the underlay is very useful for overlay routing as explained in Section 4.3.1. However, the existing solutions in [123, 125] both had exponential complexity in the number of paths. *In this regard, we improve network tomography by developing the first polynomial-complexity algorithm for inferring how a set of arbitrary paths share links from end-to-end measurements.*

Available capacity estimation: Another key piece of information for overlay routing is the available capacities of underlay links (after subtracting background traffic). Available capacity estimation is a classical problem for which many tools have been developed; see [126] for a comprehensive survey. In an uncooperative underlay as considered in our work, tools based on ICMP such as `pathchar` [127] are inapplicable. Instead, the focus was on inferring the available capacity at the bottleneck link of a path from end-to-end measurements. To this end, one line of works was based on the *probe gap model (PGM)* [128, 129], which calculated the path capacity based on the interarrival time at the receiver between a pair of back-to-back probes. Another line of works was based on the *probe rate model (PRM)* [130], which gradually varied the probing rate while measuring the end-to-end delays, and estimated the available capacity as the probing rate associated with a turning point in the delays. All these methods considered a single path. *To support overlay routing, we leverage the existing single-path capacity estimation methods as subroutines and develop an algorithm to estimate the total available capacity over multiple paths with possibly shared links.*

In this regard, our work is related to *shared bottleneck detection (SBD)* [111, 131–133], which aims to detect which subset of flows share a bottleneck. However, most SBD methods only detect the existence of a shared bottleneck without estimating its available capacity. Although the algorithm in [111] can estimate the bottleneck capacity, it was only applicable to paths with the same destination. More importantly, *SBD can only estimate the capacities of the links acting as bottlenecks under a given flow assignment, and cannot characterize the feasible region for all flow assignments, which is the focus of our work.*

4.1.2 Summary of Contributions

We study the problem of overlay routing over an uncooperative underlay, with the following contributions:

1) We identify the minimum information about the underlay that is both sufficient for congestion-free overlay routing and uniquely identifiable from measurements between overlay nodes.

2) We develop the first polynomial-complexity algorithm to detect the existence of underlay links shared (exclusively) by each subset of paths from end-to-end measurements between overlay nodes, under arbitrary routing in the underlay. We also develop an alternative algorithm to detect the same from the metrics of 1-by-2 components in the special case of symmetric tree-based routing. Furthermore, we develop a greedy algorithm to estimate the effective capacity of the detected links based on existing single-path available capacity estimation methods.

3) We prove that our detection results have error probabilities that decay exponentially with the sample size, and our estimation results are no more than a constant factor away from the ground truth.

4) We test our solution against benchmarks via packet-level simulations in NS3 based on real network topologies and link parameters. Our results show that despite facing inference errors, our algorithms can still better characterize the feasible region for overlay routing than existing solutions, which leads to notably less congestion and better communication performance.

Roadmap. Section 4.2 formulates our overlay routing problem, for which Section 4.3 addresses how to infer the required information about the underlay, and Section 4.4 shows how to use the inferred information in overlay routing. Both solutions are evaluated in Section 4.5. Finally, Section 4.6 concludes the paper. All the proofs can be found in Appendix 4.7.1.

4.2 Problem Formulation

4.2.1 Network Model

The underlay network is modeled as a connected undirected graph $\underline{G} = (\underline{V}, \underline{E})$, where $\underline{V}$ denotes the set of underlay nodes and $\underline{E}$ the set of underlay links. Each link $\underline{e} \in \underline{E}$ has a finite capacity $C_{\underline{e}}$.

The overlay network, managed by a centralized entity such as an *overlay network operation center (ONOC)* [134] or a software defined wide area network (SD-WAN) controller [104], is modeled as a connected directed graph $G = (V, E)$, where $V \subseteq \underline{V}$ is the set of nodes that are part of the overlay (e.g., running the overlay application), and each overlay link $e = (i, j) \in E$ denotes a tunnel between two overlay nodes that maps to the underlay routing path $\underline{p}_{i,j}$ from node i to node j. We do not impose any limiting assumption on the underlay routes, and allow asymmetric routing (i.e., $\underline{p}_{i,j}$ and $\underline{p}_{j,i}$ may not be the same). In the sequel, we will use "tunnel" and "overlay link" interchangeably.

Remark: The assumption of centralized management of the overlay is used to study the overlay routing problem without worrying about coordination within the overlay; the extension to distributed solutions is left to future work.

4.2.2 Objective of Overlay Routing

Given a set of flow demands H, the goal of overlay routing is to optimally satisfy these demands by controlling the data forwarding among overlay nodes. We consider an uncooperative underlay by assuming that: (i) the overlay can control how to route its flows among the overlay nodes, but not how to route between adjacent overlay nodes within the underlay; (ii) the overlay can observe the overlay topology G and the parameters of overlay links, but not the underlay topology $\underline{G}$, its routing paths $\{\underline{p}_{i,j}\}_{(i,j)\in E}$, or the parameters of underlay links.

In the above context, a basic need of the overlay is to route its flows to optimize certain performance metric of interest, subject to capacity constraints imposed by the underlay. As a concrete example, consider the objective of minimizing the overlay routing cost as formulated below. Suppose that each flow demand $h \in H$ specifies a source $s_h \in V$, a destination $t_h \in V$, and a fixed flow rate d_h. Sending a unit of flow over tunnel $(i, j) \in E$ incurs a routing cost of $c_{ij} \geq 0$, which can model considerations like bandwidth leasing cost or QoS degradation cost (e.g., delay). The overlay can control how the flows traverse overlay nodes through a decision variable $x_{ij}^h \in \{0, 1\}$, which indicates whether flow $h \in H$ traverses tunnel (i, j) (in the direction of $i \to j$). Define b_i^h as 1 if $i = s_h$, -1 if $i = t_h$, and 0 otherwise. The *minimum cost overlay routing* problem can be formulated as follows:

$$\min_{\boldsymbol{x}} \quad \sum_{(i,j)\in E} c_{ij} \sum_{h\in H} d_h x_{ij}^h \tag{4.1a}$$

$$\text{s.t.} \quad \sum_{(i,j)\in E:\, \underline{e}\in \underline{p}_{i,j}} \sum_{h\in H} d_h x_{ij}^h \le C_{\underline{e}}, \quad \forall \underline{e} \in \underline{E}, \tag{4.1b}$$

$$\sum_{j\in V} x_{ij}^h = \sum_{j\in V} x_{ji}^h + b_i^h, \quad \forall h \in H, i \in V, \tag{4.1c}$$

$$x_{ij}^h \in \{0,1\}, \quad \forall h \in H, (i,j) \in E. \tag{4.1d}$$

The objective (4.1a) is the total routing cost for the overlay. Constraint (4.1b) is the *per-link capacity constraint* to ensure that the load on each underlay link is within its capacity, constraint (4.1c) is the flow conservation constraint to ensure that the overlay links in $\{(i,j) \in E : x_{ij}^h = 1\}$ form a path from s_h to t_h ($\forall h \in H$), and constraint (4.1d) ensures that only one path is selected for each flow (assuming single-path routing is required). Therefore, the optimal solution to (4.1) provides the set of overlay paths to route the flows in H that achieves the minimum routing cost without causing congestion.

The optimization (4.1) is NP-hard, as it is a generalization of the *minimum-cost multiple-source unsplittable flow problem (MMUFP)* that is NP-hard [135]. Nevertheless, as an integer linear programming (ILP) problem, it can be tackled by a number of heuristics developed for MMUFP, e.g., greedy and LP relaxation with randomized rounding [135], and the optimal solution can also be computed for small instances by existing ILP solvers via algorithms such as branch-and-price-and-cut [136].

Remark 1: The formulation (4.1) is just an example of the possible objectives of overlay routing. Other formulations can also be considered. For instance, in addition to the routing cost (4.1a), there may also be a cost in setting up tunnels as considered in [104], and instead of fixing the flow rate d_h, the overlay may want to design d_h to finish data transfer as soon as possible [102, 106]. We will focus on the formulation (4.1) in this work for concreteness and leave the study of other formulations to future work.

Remark 2: The optimization (4.1) assumes that there exists at least one solution $\boldsymbol{x}$ that can satisfy all the demands in H within the capacity constraint (4.1b), i.e., (4.1) is feasible. When this assumption is violated, we can relax the constraint (4.1b) into

$$\sum_{(i,j)\in E:\, \underline{e}\in \underline{p}_{i,j}} \sum_{h\in H} d_h x_{ij}^h \le C_{\underline{e}}\omega, \quad \forall \underline{e} \in \underline{E}, \tag{4.2}$$

by introducing a new variable $\omega \ge 1$ to denote the maximum overloading factor for the underlay links. We can ensure feasibility while discouraging overloading by adding a penalty term "$c_\omega(\omega - 1)$" to the objective function (4.1a), where the parameter $c_\omega \ge 0$ controls the tradeoff between cost and congestion. Setting c_ω to a large value will make

congestion avoidance the primary objective and cost minimization the secondary objective, which reduces the relaxed formulation to (4.1) in underloaded cases and to a *minimum overload overlay routing* problem, i.e., min ω s.t. (4.2), (4.1c), (4.1d), in overloaded cases.

4.2.3 Problem Statement

From (4.1), we can see that overlay routing depends on the underlay primarily through the capacity constraint (4.1b), which requires two pieces of information: (i) how the tunnels are routed through the underlay $(\underline{p}_{i,j})_{(i,j)\in E}$, and (ii) the underlay link capacities $(C_{\underline{e}})_{\underline{e}\in\underline{E}}$. While the overlay may have other considerations requiring further information about the underlay, satisfying the capacity constraint is a basic requirement, and is thus the focus of our work.

Compared to routing in flat networks, the main challenge for routing in overlay networks is the lack of information about the underlay. In contrast to existing works on overlay routing that resorted to either the underlay's cooperation or heuristic inference methods to obtain the information they required (see Section 4.1.1), we aim at developing a complete solution that infers the minimum information needed for overlay routing based on measurements at overlay nodes with guaranteed accuracy, and then optimizes overlay routing based on the inferred information, using the minimum cost overlay routing problem (4.1) as a concrete example.

4.3 Overlay-based Inference

We will first analyze the minimum information the overlay needs about the underlay and then address how to infer this information.

4.3.1 Minimum Information for Overlay Routing

A straightforward implementation of (4.1) requires detailed knowledge of the underlay topology in terms of the routes $(\underline{p}_{i,j})_{(i,j)\in E}$ and the link capacities $(C_{\underline{e}})_{\underline{e}\in\underline{E}}$, in order to formulate constraint (4.1b). A natural question is thus whether we can directly apply solutions from topology inference to obtain this information. At a first look, the answer seems negative without further assumptions, because topology inference faces an inherent ambiguity that the routing topology capable of generating a given set of end-to-end measurements is generally not unique [123]. However, to support overlay routing, there

is actually no need to infer the underlay topology. Instead, it suffices to infer just enough information to compute the feasible region defined by constraint (4.1b).

To formalize this idea, we introduce the following notion, adapted from [123, 125] to our problem.

Definition 4.3.1. *A **category of links traversed by** F **out of** E ($F \subseteq E$) is the set of underlay links traversed by and only by the tunnels in F out of all the tunnels in E, i.e,*[1]

$$\Gamma_F(E) \coloneqq \Big(\bigcap_{(i,j)\in F} \underline{p}_{i,j} \Big) \setminus \Big(\bigcup_{(i,j)\in E\setminus F} \underline{p}_{i,j} \Big). \tag{4.3}$$

A straightforward implication of the above definition is that the paths measurable by the overlay induce the following partition of the underlay links:

$$\underline{E} = \bigcup_{F\subseteq E} \Gamma_F(E). \tag{4.4}$$

For example, in Fig. 4.1, if E contains all the tunnels between the nodes $\{a, b, c, d, e\}$, then link $(h_1, h_2) \in \Gamma_F(E)$ for $F \coloneqq \{(a, e), (e, a), (a, d), (d, a), (b, e), (e, b), (b, d), (d, b)\}$, because (h_1, h_2) is traversed by all the tunnels in F but no other tunnel in E.

Our key observation is that since all the links in the same category are traversed by the same set of tunnels, they must carry the same traffic load from the overlay. Therefore, we can reduce the per-link capacity constraint (4.1b) to the following *per-category capacity constraint*:

$$\sum_{(i,j)\in F} \sum_{h\in H} d_h x_{ij}^h \le C_F, \quad \forall F \subseteq E \text{ with } \Gamma_F(E) \neq \emptyset, \tag{4.5}$$

where C_F, referred to as the *category capacity*, is the minimum capacity of all the links in category $\Gamma_F(E)$, i.e.,

$$C_F \coloneqq \min_{\underline{e}\in\Gamma_F(E)} C_{\underline{e}}. \tag{4.6}$$

The new constraint (4.5) is equivalent to the original constraint (4.1b) in that an overlay routing solution satisfies one of these constraints if and only if it satisfies the other. However, instead of requiring detailed information about the underlay (i.e., $(\underline{p}_{i,j})_{(i,j)\in E}$ and $(C_{\underline{e}})_{\underline{e}\in\underline{E}}$), *implementing constraint* (4.5) *only requires the knowledge of the nonempty*

[1] We abuse the notation a little to use $\underline{p}$ to denote the set of links traversed by path $\underline{p}$.

categories and their capacities.

4.3.2 Identification of Nonempty Categories

The identification of nonempty categories from end-to-end measurements has been used as an intermediate step in topology inference [123, 125]. The idea is to define an additive metric such that the path-level metrics can be estimated from end-to-end measurements and the category-level metrics can be estimated from the path-level metrics. Then under the following assumption, we can detect the nonempty categories as those with non-zero metrics.

Assumption 2. *All nonempty categories have non-zero metrics.*

This assumption holds as long as all the underlay links have positive metrics, which intuitively means that every link imposes non-zero performance degradation (e.g., loss, delay, delay variation) to packets traversing it. This assumption is reasonable, as a link with no impact on communication performance will not be detectable from end-to-end measurements.

4.3.2.1 Defining Additive Metrics

We first need to define a performance metric $\theta_{\cdot}$ such that: (i) the link metrics are nonnegative and additive, and (ii) the corresponding path metrics can be reliably inferred from end-to-end performance measurements. Following [123], we adopt a metric of the form:

$$\theta_{\underline{e}} := -\log \alpha_{\underline{e}}, \tag{4.7}$$

where $\alpha_{\underline{e}} \in (0, 1)$ denotes the probability for a packet transmitted over link $\underline{e}$ to experience the "good state", with different versions of this metric for different definitions of $\alpha_{\underline{e}}$. For example, if $\alpha_{\underline{e}}$ is the probability for a packet to successfully traverse $\underline{e}$ without being lost, then (4.7) is the *loss-based metric* [116], and if $\alpha_{\underline{e}}$ is the probability for a packet to traverse $\underline{e}$ without incurring queueing delay, then (4.7) is the *utilization-based metric* [116].

To make this metric additive, we assume that the states of different underlay links are independent of each other, which is a common assumption in topology inference [116, 119, 123, 125]. Moreover, to discover shared links, we adopt a commonly-used probing method of sending batches of concurrent probes over all the tunnels. Due to the fact that packets arriving at a link in quick succession experience very similar performance, probes

in the same batch are assumed to experience the same link state when traversing a shared link, which is again a common assumption [116, 117, 123, 125]. Let $S_F \in \{0, 1\}$ indicate whether the probes in a batch experience good states on all the tunnels in $F \subseteq E$. As $S_F = 1$ if and only if all the underlay links in $\bigcup_{(i,j)\in F} \underline{p}_{i,j}$ are in good states, we have

$$\rho_F \coloneqq -\log \Pr\{S_F = 1\} = -\log \left(\prod_{\underline{e} \in \bigcup_{(i,j)\in F} \underline{p}_{i,j}} \alpha_{\underline{e}} \right) = \sum_{\underline{e} \in \bigcup_{(i,j)\in F} \underline{p}_{i,j}} \theta_{\underline{e}}, \tag{4.8}$$

which means that $\theta_{\underline{e}}$ defined in (4.7) is an additive metric over a union of simultaneously probed paths. Here ρ_F denotes the metric for the union of paths for the tunnels in F, which can be estimated consistently by the overlay from observations of S_F.

Remark: The assumptions of independent states at different links and identical states at a shared link for probes in the same batch are simplifying assumptions that may not hold strictly in practice. Nevertheless, solutions derived from these assumptions have been validated in Internet experiments [116]. We will stress-test our solution derived from these assumptions in NS3 simulations where the assumptions may not hold (see Section 4.5).

4.3.2.2 Inferring Category Metrics

The overlay cannot directly generate equation (4.8) as it does not know the routing path $\underline{p}_{i,j}$ for each tunnel $(i, j) \in E$. Nevertheless, the overlay can utilize the estimate of ρ_F to infer the following information about the categories without any knowledge of the routing paths.

Definition 4.3.2. *For a given category $\Gamma_F(E)$, the associated **category metric** $w_F(E)$ is defined as the sum metric for all the links in category $\Gamma_F(E)$, i.e., $w_F(E) \coloneqq \sum_{\underline{e}\in\Gamma_F(E)} \theta_{\underline{e}}$.*

The key is to note that by the definition of category, we have

$$\bigcup_{(i,j)\in F} \underline{p}_{i,j} = \bigcup_{F'\subseteq E: F'\cap F\neq\emptyset} \Gamma_{F'}(E), \quad \forall F \subseteq E, \tag{4.9}$$

which allows (4.8) to be rewritten as an equation of category metrics:

$$\rho_F = \sum_{F'\subseteq E:\ F'\cap F\neq\emptyset} w_{F'}(E), \quad \forall F \subseteq E. \tag{4.10}$$

Equations like (4.10) can be generated without prior knowledge of the underlay topology. Moreover, these equations are known to uniquely determine the category metrics.

Theorem 4.3.1 (Theorem III.1 in [124]). *Given the path metrics* $(\rho_F)_{F\subseteq E, F\neq\emptyset}$*, the category metrics* $(w_F)_{F\subseteq E, F\neq\emptyset}$ *are uniquely determined by* (4.10).

This theorem, together with the fact that link metrics affect path metrics only through category metrics, implies that the category metrics are the metrics of the finest granularity that can be uniquely identified by the overlay.

Example: Consider the network in Fig. 4.1. If only considering the tunnels in $E = \{(a, e), (a, d)\}$, we can partition the traversed underlay links into three nonempty categories: $\Gamma_{F_1}(E) = \{(h_2, e)\}$ for $F_1 \coloneqq \{(a, e)\}$, $\Gamma_{F_2}(E) = \{(h_2, d)\}$ for $F_2 \coloneqq \{(a, d)\}$, and $\Gamma_E(E) = \{(a, h_1), (h_1, h_2)\}$ (the other links are in category $\Gamma_\emptyset(E)$). Thus, the category metrics are $w_{F_1}(E) = \theta_{(h_2,e)}$, $w_{F_2}(E) = \theta_{(h_2,d)}$, and $w_E(E) = \theta_{(a,h_1)} + \theta_{(h_1,h_2)}$. Based on (4.10), we have a linear system:

$$\rho_{F_1} = w_E(E) + w_{F_1}(E), \tag{4.11a}$$

$$\rho_{F_2} = w_E(E) + w_{F_2}(E), \tag{4.11b}$$

$$\rho_E = w_E(E) + w_{F_1}(E) + w_{F_2}(E), \tag{4.11c}$$

which uniquely determines $(w_{F_1}(E), w_{F_2}(E), w_E(E))$. Meanwhile, the same path metrics $(\rho_{F_1}, \rho_{F_2}, \rho_E)$ can be generated by many different topologies (e.g., there may be multiple links between h_2 and e, or the tunnels (a, e) and (a, d) may join/branch multiple times) as long as the category metrics $(w_{F_1}(E), w_{F_2}(E), w_E(E))$ remain the same, making the category metrics the finest granularity information that the overlay can reliably infer from its measurements.

4.3.2.3 Taming Exponential Complexity

A straightforward solution for detecting nonempty categories based on solving (4.10) faces a severe limitation that the complexity grows at $\mathcal{O}(2^{|E|})$, where $|E| = \mathcal{O}(|V|^2)$, as the number of equations/variables is $\mathcal{O}(2^{|E|})$. This renders the straightforward solution inapplicable beyond overlays with just a few nodes. To address this limitation, we develop a novel polynomial-complexity algorithm for category metric inference.

Our solution is based on dynamic programming. Instead of considering all the tunnels in one shot, we start with only a small subset of tunnels, for which (4.10) can be solved within acceptable time/space to obtain coarse-grained category metrics, and then we

Algorithm 9: CategOry IdeNtification (COIN)

input : Set of all tunnels E, metric $\boldsymbol{\rho}$, detection threshold η
output : Set of detected nonempty categories $\mathcal{F}$

solve (4.10) to compute $\boldsymbol{w}(E_0)$ for an initial set of tunnels $E_0 \subseteq E$;
for $t = 1, \ldots, |E| - |E_0|$ **do**
  $E_t \leftarrow E_{t-1} \cup \{e\}$ for an arbitrary tunnel $e \in E \setminus E_{t-1}$;
  $w_{\{e\}}(E_t) \leftarrow \rho_{E_t} - \rho_{E_{t-1}}$;
  for $F \in supp(\boldsymbol{w}(E_{t-1}))$ *in increasing order of* $|F|$ **do**
    $w_{F\cup\{e\}}(E_t) \leftarrow$
    $\rho_{(E_{t-1}\setminus F)\cup\{e\}} - \rho_{E_{t-1}\setminus F} - w_{\{e\}}(E_t) - \sum_{F'\subset F: F'\in \mathrm{supp}(\boldsymbol{w}(E_{t-1}))} w_{F'\cup\{e\}}(E_t)$;
    $w_F(E_t) \leftarrow w_F(E_{t-1}) - w_{F\cup\{e\}}(E_t)$;
return $\mathcal{F} = \{F \in \mathrm{supp}(\boldsymbol{w}(E)) : w_F(E) > \eta\}$;

gradually expand the set of considered tunnels to refine the category metrics until all the tunnels are included. Our approach is motivated by the following observations:

Lemma 4.3.1. *The number of nonempty categories is upper-bounded by the number of links in the underlay, i.e.,* $|\{\Gamma_F(E') : F \subseteq E', \Gamma_F(E') \neq \emptyset\}| \leq |\{\underline{e} : \underline{e} \in \cup_{(u,v)\in E'} \underline{p}_{u,v}\}| \leq |\underline{E}|$ *for any* $E' \subseteq E$.

Lemma 4.3.2. *For any* $E' \subset E$ *and* $e \in E \setminus E'$, $w_F(E') = 0$ *implies* $w_F(E' \cup \{e\}) = w_{F\cup\{e\}}(E' \cup \{e\}) = 0$, *for all* $F \subseteq E'$ *and* $F \neq \emptyset$.

Lemma 4.3.3. *For any* $E' \subset E$ *and* $e \in E\setminus E'$, $w_F(E') = w_F(E'\cup\{e\}) + w_{F\cup\{e\}}(E'\cup\{e\})$.

Lemma 4.3.1 means that the vector of category metrics is sparse, and Lemma 4.3.2 means that the sparsity pattern of this vector for a subset of tunnels can be used to estimate its sparsity pattern as we consider more tunnels. Lemma 4.3.3 allows us to use the previously computed category metrics defined for a subset of tunnels to solve for the new category metrics when considering one more tunnel.

Algorithm: Based on the above observations, we develop *CategOry IdeNtification (COIN)*, a dynamic programming algorithm for identifying the nonempty categories, as shown in Algorithm 9. We ignore estimation error in $\boldsymbol{\rho}$ for now to focus on the main idea; how to handle estimation error will be discussed later in Section 4.3.2.4. Here, E_t denotes the set of tunnels considered in iteration t, $\boldsymbol{w}(E_t) \coloneqq (w_F(E_t))_{F\subseteq E_t, F\neq\emptyset}$, and $\mathrm{supp}(\boldsymbol{w}(E_t)) \coloneqq \{F \subseteq E_t : F \neq \emptyset, w_F(E_t) \neq 0\}$. The algorithm first uses measurements from a small set of tunnels E_0 to compute a vector of coarse-grained category metrics $\boldsymbol{w}(E_0)$ by directly solving (4.10). It then gradually refines the solution by expanding the set of considered tunnels. In iteration t, the equations corresponding to $E_t = E_{t-1} \cup \{e\}$

can be classified into two types:

$$\rho_F = \sum_{F' \subseteq E_{t-1}, F' \cap F \neq \emptyset} \Big(w_{F'}(E_t) + w_{F' \cup \{e\}}(E_t) \Big), \tag{4.12}$$

$$\rho_{F \cup \{e\}} = \sum_{F' \subseteq E_{t-1}, F' \cap F \neq \emptyset} \Big(w_{F'}(E_t) + w_{F' \cup \{e\}}(E_t) \Big) + \sum_{F' \subseteq E_{t-1} \setminus F} w_{F' \cup \{e\}}(E_t), \tag{4.13}$$

where (4.12) is $\forall F \subseteq E_{t-1}$, $F \neq \emptyset$ and (4.13) is $\forall F \subseteq E_{t-1}$. Given the solution $\boldsymbol{w}(E_{t-1})$ from the previous iteration, equations of type (4.12) become redundant, as their information is already contained in the simpler equations $w_F(E_{t-1}) = w_F(E_t) + w_{F \cup \{e\}}(E_t)$ based on Lemma 4.3.3. Equations of type (4.13) can be rewritten as

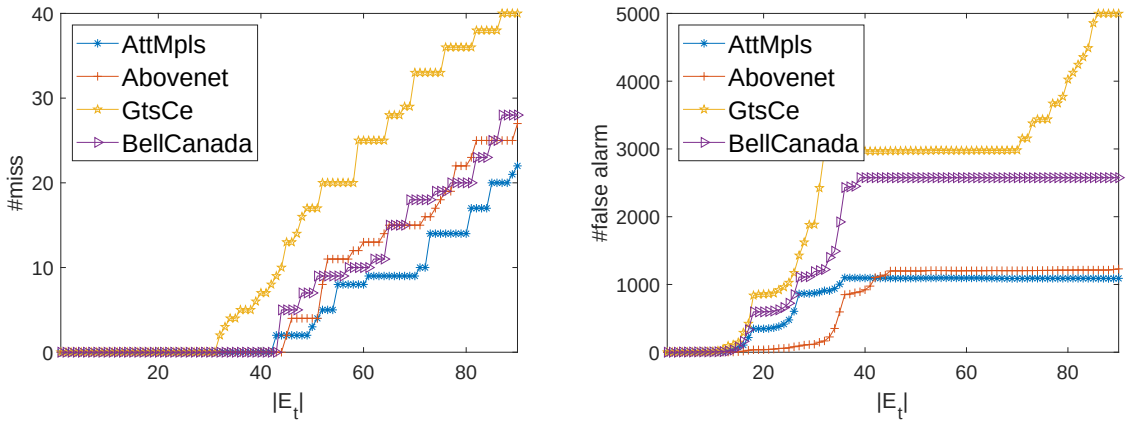
$$\sum_{F' \subseteq F} w_{F' \cup \{e\}}(E_t) = \rho_{(E_{t-1} \setminus F) \cup \{e\}} - \rho_{E_{t-1} \setminus F}. \tag{4.14}$$

For $F = \emptyset$, (4.14) contains only one unknown variable $w_{\{e\}}(E_t)$, and hence can be used to compute $w_{\{e\}}(E_t)$ as in line 4. Based on this initial solution, we can use (4.14) to gradually solve $w_{F \cup \{e\}}(E_t)$ in the increasing order of $|F|$ as in line 6, because when we try to solve $w_{F \cup \{e\}}(E_t)$, the values of $w_{F' \cup \{e\}}(E_t)$ for any $F' \subset F$ have been obtained. Once $w_{F \cup \{e\}}(E_t)$ is obtained, we can apply Lemma 4.3.3 to compute $w_F(E_t)$ as in line 7. In this process, we use the observation in Lemma 4.3.2 to reduce complexity by only computing $w_F(E_t)$ and $w_{F \cup \{e\}}(E_t)$ for $F \subseteq E_{t-1}$ satisfying $F \neq \emptyset$ and $w_F(E_{t-1}) \neq 0$. Note that $E_{|E|-|E_0|} = E$.

Complexity: Algorithm 9 significantly improves the complexity of category metric inference compared to the straightforward solution. Specifically, under perfect estimation of the path metrics, each iteration (lines 2–7) incurs $O(|\underline{E}|^2)$ operations, stores $O(|\underline{E}|)$ variables, and performs $O(|\underline{E}|)$ estimations of path metrics, because the number of non-zero category metrics $|\text{supp}(\boldsymbol{w}(E_{t-1}))| \leq |\underline{E}|$ by Lemma 4.3.1. As there are $O(|E|)$ iterations, the total complexity is $O(|E| \cdot |\underline{E}|^2)$ in time, $O(|\underline{E}|)$ in space (reused across iterations), and $O(|E| \cdot |\underline{E}|)$ in the number of path metric estimations.

4.3.2.4 Handling Estimation Errors

In practice, errors in the estimated path metrics $\hat{\boldsymbol{\rho}}$ are inevitable, which will cause errors in the inferred category metrics $\hat{\boldsymbol{w}}(E_t)$. In particular, such errors may even cause some of the inferred category metrics to be negative. To mitigate the impact of estimation



(a) misses (b) Computation time

Figure 4.2. Errors of COIN with varying #tunnels under the same settings as Tables 4.2-4.3 in Section 4.5.

errors on nonempty category identification, we propose the following two enhancements for Algorithm 9.

The first enhancement is based on the observation that Lines 4-7 in Algorithm 9 is equivalent to solving the linear system below:

$$\begin{pmatrix} \hat{\boldsymbol{\rho}}_t \\ \boldsymbol{w}_{t-1} \end{pmatrix} = \begin{pmatrix} \boldsymbol{A}_{t,1} \\ \boldsymbol{A}_{t,2} \end{pmatrix} \boldsymbol{w}_t. \tag{4.15}$$

Here, $\boldsymbol{w}_t$ is a vector containing $w_{\{e\}}(E_t)$ and $w_{F_{t-1}}(E_t)$, $w_{F_{t-1}\cup\{e\}}(E_t)$, $\forall F_{t-1} \in \text{supp}(\boldsymbol{w}(E_{t-1}))$; $\hat{\boldsymbol{\rho}}_t = \boldsymbol{A}_{t,1}\boldsymbol{w}_t$ is (4.13) in matrix form with $\hat{\boldsymbol{\rho}}_t$ containing $\hat{\rho}_{(E_{t-1}\setminus F_{t-1})\cup\{e\}}$ for all $F_{t-1} \in \text{supp}(\boldsymbol{w}(E_{t-1})) \cup \{\emptyset\}$; $\boldsymbol{w}_{t-1} = \boldsymbol{A}_{t,2}\boldsymbol{w}_t$ contains the equations $w_{F_{t-1}}(E_{t-1}) = w_{F_{t-1}}(E_t) + w_{F_{t-1}\cup\{e\}}(E_t), \forall F_{t-1} \in \text{supp}(\boldsymbol{w}(E_{t-1}))$. Considering the errors in $\hat{\boldsymbol{\rho}}_t$, we propose to infer $\boldsymbol{w}_t$ by solving the following non-negative least squares problem instead of (4.15):

$$\min_{\boldsymbol{w}_t \geq \boldsymbol{0}} \left\| \begin{pmatrix} \hat{\boldsymbol{\rho}}_t \\ \boldsymbol{w}_{t-1} \end{pmatrix} - \begin{pmatrix} \boldsymbol{A}_{t,1} \\ \boldsymbol{A}_{t,2} \end{pmatrix} \boldsymbol{w}_t \right\|_2. \tag{4.16}$$

The second enhancement is based on the observation in Fig. 4.2, that COIN is highly accurate when the number of considered tunnels $|E_t|$ is small. Closer examination shows that COIN is particularly accurate when $E_t \subseteq E^{(s)}$, where $E^{(s)} \coloneqq \{(s,t) \in E : \forall t \in V\}$ is the subset of tunnels with the same source s. Thus, we propose to use the inferred $\text{supp}\big(\boldsymbol{w}(E^{(s)})\big)$ $(\forall s \in V)$ to further improve (4.16). To this end, we have the following observations.

Lemma 4.3.4. *For any E_t and $F \subseteq E' \subseteq E_t$, if $\nexists F' \subseteq \mathit{supp}\,(\boldsymbol{w}(E_t))$ such that $F \subseteq F'$, then $w_F(E') = 0$.*

Lemma 4.3.5. *Given E_0, E_1 with $E_0 \cap E_1 = \emptyset$, we must have $w_{F_0 \cup F_1}(E_0 \cup E_1) = 0$ if $w_{F_0}(E_0) = 0$ or $w_{F_1}(E_1) = 0$, for all $F_0 \subseteq E_0$ and $F_1 \subseteq E_1$.*

Theorem 4.3.2. *Given $E_t \coloneqq E_{t-1} \cup \{e\}$ for $e = (s,t)$, then $\forall F' = F \cup \{e\}$ for $F \in \mathit{supp}(\boldsymbol{w}(E_{t-1}))$, we must have $w_{F'}(E_t) = 0$ if there is no $F'' \subseteq \mathit{supp}\Big(\boldsymbol{w}(E^{(s)})\Big)$ such that $\Big(F' \cap E^{(s)}\Big) \subseteq F''$.*

Theorem 4.3.2 implies that we can reduce the number of variables in each iteration with the knowledge of $(\mathrm{supp}\Big(\boldsymbol{w}(E^{(s)})\Big))_{s \in V}$, by ignoring $w_{F \cup \{e\}}(E_t)$ for $F \in \mathrm{supp}(\boldsymbol{w}(E_{t-1}))$ that is known to be zero.

We refer to the variation of Algorithm 9 with the above two enhancements as *Robust CategOry IdeNtification (R-COIN)*, which is presented in Algorithm 10. R-COIN follows the same idea of COIN in Algorithm 9 with the following differences:

- During initialization, instead of computing the category metrics for a single initial set of tunnels E_0, R-COIN computes the category metrics for all the sets of tunnels with the same source $(E^{(v_i)})_{v_i \in V}$, as in Lines 1–2.
- R-COIN solves for a smaller set of variables in each iteration based on Theorem 4.3.2, as shown in Line 6.
- R-COIN estimates the category metrics by solving the non-negative least squares problem (4.16) as in Line 7.

4.3.2.5 Performance Analysis

We now quantify the error in detecting nonempty categories using the inferred category metrics. Let $\hat{w}_F(E)$ denote the inferred metric of category $\Gamma_F(E)$ and $\eta > 0$ denote the detection threshold. To gain explicit insights, our analysis will focus on the vanilla case where $\hat{\boldsymbol{w}}(E)$ is obtained by directly solving (4.10) based on the estimated path metrics $\hat{\boldsymbol{\rho}}$. The modifications introduced in Sections 4.3.2.3–4.3.2.4 make it difficult to obtain explicit insights through analysis, and thus will be evaluated empirically (see Section 4.5).

All the errors originate from the error in estimating the path metric ρ_F defined in (4.8). As common in the literature [116,117], we assume that ρ_F is estimated by plugging the empirical probability $\overline{S}_F \coloneqq \frac{1}{T}\sum_{t=1}^{T} S_{F,t}$ into (4.8):

$$\hat{\rho}_F \coloneqq -\log \overline{S}_F, \tag{4.17}$$

Algorithm 10: Robust CategOry IdeNtification (R-COIN)

input : All tunnels E, estimated metric $\hat{\boldsymbol{\rho}}$, detection threshold η
output : Set of detected nonempty categories $\mathcal{F}$

foreach $v_i \in V$, $i = 0, \ldots, |V| - 1$ **do**
  apply Algorithm 9 to obtain $\boldsymbol{w}(E^{(v_i)})$;
set $E_0 \leftarrow E^{(v_0)}$, $t = 1$;
for $i = 1, \cdots, |V| - 1$ **do**
  while $\exists e \in E^{(v_i)} \setminus E_{t-1}$ **do**
    for $E_t \leftarrow E_{t-1} \cup \{e\}$, construct $\boldsymbol{w}_t$ containing $\{w_F(E_t), w_{F\cup\{e\}}(E_t) : F \in \mathrm{supp}(\boldsymbol{w}(E_{t-1}))\}$, except those ruled out by Theorem 4.3.2;
    solve (4.16) for $\boldsymbol{w}_t$;
    $\mathrm{supp}(\boldsymbol{w}(E_t)) \leftarrow \{F \subseteq E : w_F(E_t)$ contained in $\boldsymbol{w}_t, w_F(E_t) > \eta\}$;
    $t \leftarrow t + 1$;
return $\mathcal{F} = \mathrm{supp}(\boldsymbol{w}(E))$;

where $S_{F,t} \in \{0, 1\}$ indicates whether the probes in the t-th batch experience good states on all the tunnels in F. We now analyze the error in identifying nonempty categories as a function of the sample size T and other parameters.

We start by deriving the solution to (4.10) in closed form.

Lemma 4.3.6. *Each category metric is related to the path metrics by*

$$w_F(E) = \sum_{F' \subseteq F} (-1)^{|F'|+1} \rho_{(E\setminus F)\cup F'}, \ \forall F \subseteq E, F \neq \emptyset. \tag{4.18}$$

We then analyze the error in estimating ρ_F by (4.17). Let $s_F \coloneqq \Pr\{S_F = 1\}$ for ease of presentation.

Lemma 4.3.7. *For $T \gg 1$, the bias of* (4.17) *satisfies*

$$\mathbb{E}[\hat{\rho}_F] - \rho_F \approx \frac{1 - s_F}{2 s_F T}, \tag{4.19}$$

and the variance satisfies

$$\mathrm{var}[\hat{\rho}_F] \approx \frac{1 - s_F}{s_F T}, \tag{4.20}$$

where smaller terms at the order of $o(1/T)$ have been ignored.

By the central limit theorem, the distribution of $\overline{S}_F$ is asymptotically Gaussian. While due to the nonlinear transform $-\log(\cdot)$, the distribution of $\hat{\rho}_F$ is not exactly

Gaussian, the delta method [137] suggested that it is well approximated by the Gaussian distribution for large T. Formally, the delta method [137] states that for a sequence of random variables $(X_n)_{n\geq 1}$ satisfying $\sqrt{n}\,(X_n-\mu)\xrightarrow{D}\mathcal{N}(0,\sigma^2)$ and a function $f(x)$ such that the first derivative $f'(x)$ exists and is non-zero, we have

$$\sqrt{n}\,(f(X_n)-f(\mu))\xrightarrow{D}\mathcal{N}\left(0,(f'(\mu))^2\sigma^2\right), \tag{4.21}$$

where $\xrightarrow{D}$ denotes the convergence in distribution. Our problem satisfies these conditions with $\sqrt{T}(\overline{S}_F-s_F)\xrightarrow{D}\mathcal{N}(0,s_F(1-s_F))$, $f(x)=-\log(x)$, $f(\overline{S}_F)=\hat{\rho}_F$, and $f(s_F)=\rho_F$. Under the Gaussian approximation, we can analyze the error in nonempty category identification in closed form as follows.

Theorem 4.3.3. *Suppose that the path metric estimation errors $\{\hat{\rho}_F-\rho_F\}_{F\subseteq E,F\neq\emptyset}$ can be modeled as independent Gaussian random variables with mean and variance given by Lemma 4.3.7. If $w_F(E)=0$, then*

$$\Pr\{\hat{w}_F(E)>\eta\}=1-\Phi\left(\frac{\eta\sqrt{T}-\tilde{\delta}_F(E)/\sqrt{T}}{\delta_F(E)}\right) \tag{4.22}$$

$$\approx\frac{\delta_F(E)}{\eta\sqrt{2\pi T}}\exp\left(-\frac{\eta^2}{2\delta_F(E)^2}T\right), \tag{4.23}$$

and if $w_F(E)>\eta$, then

$$\Pr\{\hat{w}_F(E)\leq\eta\}=\Phi\left(\frac{(\eta-w_F(E))\sqrt{T}-\tilde{\delta}_F(E)/\sqrt{T}}{\delta_F(E)}\right) \tag{4.24}$$

$$\approx\frac{\delta_F(E)}{(w_F(E)-\eta)\sqrt{2\pi T}}\exp\left(-\frac{(\eta-w_F(E))^2}{2\delta_F(E)^2}T\right), \tag{4.25}$$

where $\Phi(\cdot)$ is the CDF of the standard Gaussian distribution,

$$\delta_F(E)\coloneqq\sqrt{\sum_{F':E\setminus F\subseteq F'}(1-s_{F'})/s_{F'}}, \tag{4.26}$$

$$\tilde{\delta}_F(E)\coloneqq\sum_{F':E\setminus F\subseteq F'}(-1)^{|F'|-|E\setminus F|+1}(1-s_{F'})/(2s_{F'}), \tag{4.27}$$

and the "$\approx$" in (4.23) and (4.25) holds for $T\gg 1$.

Remark: Theorem 4.3.3 states that both the false alarm probability (4.23) and the miss probability (4.25) decay exponentially with the sample size T, with the error exponent controlled by the detection threshold η. The threshold η essentially controls what

kinds of categories are detectable, in the sense that a category must have at least one link in the "bad state" (e.g., with backlogged queue) with probability $> 1 - e^{-\eta}$ to be detectable with exponentially decaying error.

4.3.2.6 Special Case: Symmetric Tree-based Routing

Although the previous solution is applicable under arbitrary routing, it has limited accuracy in large networks due to the difficulty in accurately estimating $\boldsymbol{\rho}$. Below, we will present a more accurate solution for the special case of symmetric tree-based routing, where there is a unique path followed by all the routes traversing any two nodes $u, v \in \underline{V}$, $\underline{p}_{u,v} = \underline{p}_{v,u}$, and $\theta_{\underline{s},\underline{t}} = \theta_{\underline{t},\underline{s}}, \forall(\underline{s},\underline{t}) \in \underline{E}$. In this special case, the underlay routes originated from each overlay node $v \in V$ form a tree, and the routing topology connecting the overlay nodes is a union of the trees rooted at each of the overlay nodes. Tree-based routing is a common assumption in topology inference (see Section 4.1.1). This special case is particularly attractive because:

1. The union of trees was believed to be uniquely determined by all the 1-by-2 components[2] formed by the routes from each source to each pair of destinations [119].

2. The additive metrics for 1-by-2 components are much easier to estimate than those for general categories, as they only require us to probe two tunnels at a time that share the same source.

These observations motivate us to detect nonempty categories based on the estimated metrics of 1-by-2 components. Note that in our context, a "1-by-2 component" refers to the union of underlay routes from an overlay node to two other overlay nodes.

Algorithm: Let b^s_{tv} denote the branching point between the tunnels (s,t) and (s,v), ℓ^s_{tv} denote the additive metric for the path between s and b^s_{tv} (i.e., the metric of the shared path between (s,t) and (s,v)), and ℓ^s_t denote the additive metric for the entire path between s and t. Algorithm 11 uses this information to detect categories with sufficiently large ($> \eta$) metrics. For each tunnel $(s,t) \in E$, it detects the categories of the links on path $\underline{p}_{s,t}$ through the following steps: (1) locate the branching/joining points between (s,t) and every other tunnel (lines 4–12), indicated by their distances **from** s **on path** $\underline{p}_{s,t}$; (2) go through these branching/joining points from s to t (lines 13–15) to identify the set of tunnels traversing the links between each pair of consecutive points

[2]The original statement in [119, Theorem 1] requires all the 1-by-2 and all the 2-by-1 components, but the latter become redundant under the assumption of route symmetry.

Algorithm 11: Tree-based CategOry IdeNtification (T-COIN)

input : Set of tunnels E, estimated path metrics $(\ell_t^s)_{(s,t)\in E}$, estimated shared path metrics $(\ell_{tv}^s)_{(s,t),(s,v)\in E}$, detection threshold η

output : Set of detected nonempty categories $\mathcal{F}$

1 $\mathcal{F} \leftarrow \emptyset$;
2 expand E if needed to include each tunnel in both directions;
3 **for** *each* $(s,t) \in E$ **do**
4 **for** *each* $(s,v) \in E$ $(v \neq t)$ **do**
5 record a branching point with (s,v) at distance ℓ_{tv}^s;
6 **for** *each* $(v,t) \in E$ $(v \neq s)$ **do**
7 record a joining point with (v,t) at distance $\ell_t^s - \ell_{sv}^t$;
8 **for** *each* $(i,j) \in E$ $(\{i,j\} \cap \{s,t\} = \emptyset)$ **do**
9 **if** $\ell_{tj}^s > \ell_j^s - \ell_{si}^j$ **then**
10 record a joining point with (i,j) at distance $\ell_j^s - \ell_{si}^j$, and a branching point with (i,j) at distance ℓ_{tj}^s ;
11 **else if** $\ell_{tj}^i > \ell_t^i - \ell_{si}^t$ **then**
12 record a joining point with (i,j) at distance $\ell_t^s - \ell_{si}^t$, and a branching point with (i,j) at distance $\ell_t^s - (\ell_t^i - \ell_{tj}^i)$;
13 sort the set B_t^s of branching/joining points on $\underline{p}_{s,t}$ into $\{b^{(1)}, \ldots, b^{(|B_t^s|)}\}$ with increasing distances from s;
14 set $b^{(0)} \leftarrow s$ and $b^{(|B_t^s|+1)} \leftarrow t$;
15 **for** $k = 1, \ldots, |B_t^s| + 1$ **do**
16 **if** $b^{(k-1)} = s$ **then**
17 $F \leftarrow \{(s,v) : (s,v) \in E\}$;
18 **else if** $b^{(k-1)}$ *is a branching point with tunnel* (i,j) **then**
19 $F \leftarrow F \setminus \{(i,j)\}$;
20 **else if** $b^{(k-1)}$ *is a joining point with tunnel* (i,j) **then**
21 $F \leftarrow F \cup \{(i,j)\}$;
22 **if** *distance between* $b^{(k)}$ *and* $b^{(k-1)} > \eta$ **then**
23 $\mathcal{F} \leftarrow \mathcal{F} \cup \{F\}$;
24 return $\mathcal{F}$;

(lines 16–21), and record this set as the index of a nonempty category if the corresponding category metric exceeds a given threshold η (lines 22–23).

Example: To understand the idea in Algorithm 11, consider the topology spanned by a 4-node overlay in Fig. 4.3 (a). When considering the tunnel $(1,2)$, Algorithm 11 first locates all the branching/joining points between $(1,2)$ and the other tunnels based on the given metrics of 1-by-2 components, as shown in Fig. 4.3 (b). Each tunnel $(1,v)$ $(v = 3,4)$ branches from $(1,2)$ at distance ℓ_{2v}^1 from node 1, and each tunnel $(v,2)$ $(v = 3,4)$ joins $(1,2)$ at distance $\ell_2^1 - \ell_{1v}^2$ from node 1. For tunnel $(3,4)$ not sharing any endpoint with $(1,2)$, we need to consider a tunnel like $(1,4)$ that shares an endpoint with each. Since $(1,4)$ co-

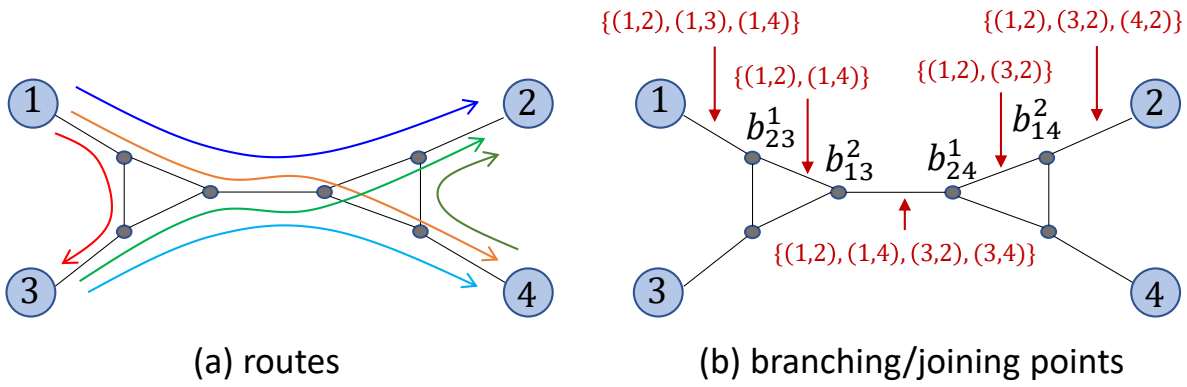


Figure 4.3. Example for tree-based category identification: Overlay nodes $V := \{1, \ldots, 4\}$; link labels in (b) denote the detected sets of traversing tunnels.

incides with $(1, 2)$ for the first portion of length ℓ^1_{24}, and $(3, 4)$ joins $(1, 4)$ after a portion of length $\ell^1_4 - \ell^4_{13}$, we know that $(3, 4)$ has non-zero overlap with $(1, 2)$ if $\ell^1_{24} > \ell^1_4 - \ell^4_{13}$, which is satisfied in this case. Then Algorithm 11 identifies the set of tunnels traversing each pair of consecutive branching/joining points via dynamic programming: starting from the set $\{(1, 2), (1, 3), (1, 4)\}$ of all the tunnels originating from node 1, it updates this set by excluding a tunnel branching from $(1, 2)$ when moving past a branching point (e.g., excluding $(1, 3)$ when moving past b^1_{23}), and including a tunnel joining $(1, 2)$ when moving past a joining point (e.g., including $(3, 2)$ and $(3, 4)$ when moving past b^2_{13}, which is a joining point between $(1, 2)$ and both of these tunnels). The distance between consecutive points indicates the metric of the corresponding category, which is then used to detect nonempty categories.

Complexity: For any $(s, t) \in E$, there are at most one branching point and one joining point with every other tunnel, i.e., $|B^s_t| = O(|E|)$. Thus, it takes $O(|E| \log |E|)$ time to locate the branching/ joining points and sort them in order (lines 4–13). Moreover, as $|F| = O(|E|)$, the **for** loop in lines 15–23 takes $O(|E|^2)$ time. This leads to a total time complexity of $O(|E|^3)$. Meanwhile, since for each of the $|E|$ tunnels, lines 15–23 may construct $O(|E|)$ sets of tunnels, each of size $O(|E|)$, the total space complexity is also $O(|E|^3)$.

Accuracy: Although [119] claimed that the 1-by-2 components (and 2-by-1 components in the case of asymmetric routing) can uniquely determine the topology of a union of routing trees, their analysis is insufficient for our algorithm for two reasons: (i) [119] only considered a special case where the sets of sources and destinations are disjoint, while Algorithm 11 handles a more general case where the destinations of some tunnels can be the sources of some other tunnels; (ii) the proof of [119] contains a critical flaw that it ignored the existence of branching/joining points that are not branching/joining

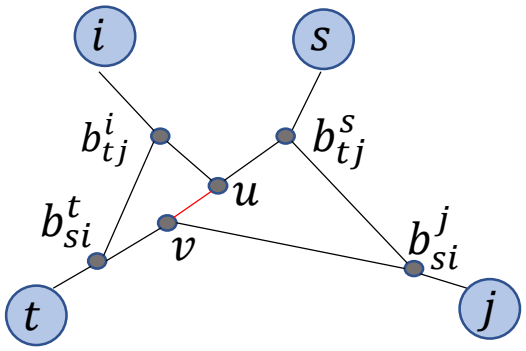


Figure 4.4. Counterexample: The links shared by tunnels (s,t) and (i,j) between u and v are undetectable from 1-by-2 components formed by the nodes in $\{s,t,i,j\}$.

points of any 1-by-2 or 2-by-1 components. It is therefore necessary to independently analyze the accuracy of Algorithm 11, as stated below.

Theorem 4.3.4. *Given accurate estimates of the path metrics* $(\ell^s_t)_{(s,t)\in E}$ *and the shared path metrics* $(\ell^s_{tv})_{(s,t),(s,v)\in E}$*, Algorithm 11 will correctly detect all the nonempty categories in* $\{\Gamma_F(E)\}_{F\subseteq E, F\neq\emptyset}$ *if (i)* $\forall (s,t),(i,j)\in E$ *such that* $\{s,t\}\cap\{i,j\}=\emptyset$*, every branching/joining point between* (s,t) *and* (i,j) *is a branching point of some 1-by-2 component formed by three nodes from* $\{s,t,i,j\}$*, and (ii) every link* $\underline{e}\in\underline{E}$ *has a metric* $\theta_{\underline{e}}>\eta$*.*

Remark 1: Assumption (ii) in Theorem 4.3.4 is not a critical assumption as one can tune the detection threshold η to achieve any desired detection sensitivity (while trading off with false alarms). Assumption (i), however, is a limiting assumption that may be violated, in which case Algorithm 11 will miss some categories. Consider the example in Fig. 4.4, even if tunnels (s,t) and (i,j) share some links between points u and v (each being an underlay node), the existence of these branching/joining points is undetectable from 1-by-2 components formed by $\{s,t,i,j\}$ as neither of them coincides with the branching point of any such component. Nevertheless, as shown in Table 4.2-4.3 in Section 4.5, Algorithm 11 can detect the majority of nonempty categories under symmetric tree-based routing, even if the ground-truth topology may violate assumptions (i–ii).

Remark 2: The assumption of symmetric link metrics $\theta_{\underline{s},\underline{t}}=\theta_{\underline{t},\underline{s}},\forall(\underline{s},\underline{t})\in\underline{E}$ can be relaxed by measuring the 2-by-1 components in addition to the 1-by-2 components, at the cost of requiring calibration of the probe transmission times between each pair of overlay nodes. Detailed study of this case is left to future work.

4.3.3 Estimation of Category Capacities

We now address the estimation of the capacity for each detected nonempty category. In the sequel, we will simply denote a category $\Gamma_F(E)$ as Γ_F and a category metric $w_F(E)$ as w_F since they are always defined with respect to all the tunnels in E.

In absence of any prior knowledge, the overlay has to measure the category capacities. However, the minimum link capacity C_F for a nonempty category Γ_F will not be measurable by the overlay if no flow assignment in the overlay can saturate the minimum-capacity link. To address this issue, we define a notion called *effective category capacity* $\tilde{C}_F$ as follows.

Definition 4.3.3. *For each $F \subseteq E$, the* effective category capacity $\tilde{C}_F$ *is the maximum flow the overlay can send through the tunnels in F, i.e.,*

$$\tilde{C}_F := \max_{(f_e)_{e\in E}} \sum_{e\in F} f_e \tag{4.28a}$$

$$s.t. \sum_{e'\in F'} f_{e'} \le C_{F'}, \quad \forall F' \subseteq E,\ \Gamma_{F'} \neq \emptyset, \tag{4.28b}$$

$$f_e \ge 0, \quad \forall e \in E. \tag{4.28c}$$

The effective category capacity is equivalent to the category capacity defined in (4.6) in that they induce the same feasible region for overlay routing, except that the effective category capacity is always achievable by the overlay. Thus, it suffices for the overlay to estimate the effective capacity of each nonempty category.

Although how to estimate the combined capacity over multiple tunnels (i.e., paths) has not been solved systematically, how to estimate the available capacity of a single tunnel has been well understood [129, 130]. Thus, we will build on top of these existing solutions to develop an algorithm for estimating $\tilde{C}_F$. Our algorithm assumes *a subroutine that can estimate the residual capacity of a tunnel e under an existing flow assignment $\boldsymbol{f}$*, which can be implemented by any of the existing available capacity estimation methods. Let $\tilde{C}_e(\boldsymbol{f})$ denote the true residual capacity of e under $\boldsymbol{f}$ and $\hat{C}_e(\boldsymbol{f})$ the estimate given by the subroutine.

Algorithm: Given this subroutine, we propose an algorithm in Algorithm 12. For each tunnel set of interest F, this algorithm goes through the tunnels in F in an arbitrary order, and tries to assign as much flow as possible onto each tunnel e_{i_j} according to the residual capacity estimated by the subroutine, without backtracking the flow assignment for $e_{i_1}, \ldots, e_{i_{j-1}}$ (lines 2–4). The effective category capacity is then estimated as the sum

Algorithm 12: Effective Category Capacity Estimation

input : set $\mathcal{F}$ of category indices of interest (e.g., $\mathcal{F} \coloneqq \{F \subseteq E : \hat{w}_F > \eta\}$

output : Estimated effective category capacities $\{\hat{C}_F\}_{F\in\mathcal{F}}$

1 **for** *each* $F \coloneqq \{e_{i_1}, \cdots, e_{i_{|F|}}\} \in \mathcal{F}$ **do**

2 $\quad f_{e_{i_1}} \leftarrow \hat{C}_{e_{i_1}}(\mathbf{0})$;

3 $\quad$ **for** $j = 2, \cdots, |F|$ **do**

4 $\quad\quad f_{e_{i_j}} \leftarrow \hat{C}_{e_{i_j}}(\boldsymbol{f})$;

5 $\quad \hat{C}_F \leftarrow \sum_{j=1}^{|F|} f_{e_{i_j}}$;

6 return $\{\hat{C}_F\}_{F\in\mathcal{F}}$;

flow (line 5).

Complexity: As Algorithm 12 invokes an existing single-path available capacity estimation method as subroutine, its exact complexity will depend on the complexity of the subroutine. Nevertheless, as the complexity of the subroutine is independent of the size of the overlay-underlay network, we can analyze the complexity of Algorithm 12 in terms of the number of invocations of the subroutine, which equals $O(|\mathcal{F}| \cdot |E|)$. As the number of nonempty categories is upper-bounded by the number of underlay links $|\underline{E}|$, under reasonably accurate nonempty category identification, the number of detected nonempty categories $|\mathcal{F}|$ will be $O(|\underline{E}|)$, and thus the complexity of Algorithm 12 will be $O(|\underline{E}| \cdot |E|)$.

Accuracy: We now analyze how the estimated effective category capacity $\hat{C}_F$ provided by Algorithm 12 compares to the true value. Under the assumption that the subroutine does not overestimate the residual capacities of individual tunnels, which is typical for PGM-based methods [129], it is easy to see that the flow assignment in Algorithm 12 is feasible for the underlay link capacities, i.e., feasible for (4.28). Thus, the achieved sum rate can only underestimate the effective category capacity, i.e., $\hat{C}_F \leq \tilde{C}_F$. Meanwhile, if the subroutine is accurate, then the estimate can only be a constant factor smaller as stated below.

Theorem 4.3.5. *If the estimation for single-tunnel residual capacity is accurate (i.e., $\hat{C}_e(\boldsymbol{f}) = \tilde{C}_e(\boldsymbol{f})$), then Algorithm 12 achieves $1/q_F$-approximation. More precisely, $\tilde{C}_F \geq \hat{C}_F \geq \tilde{C}_F/q_F$, where*

$$q_F \coloneqq \max_{e\in F} |\{F' \subseteq E : e \in F', \Gamma_{F'} \neq \emptyset, |F' \cap F| > 1\}| \tag{4.29}$$

is the maximum number of nonempty categories a tunnel in F traverses that are shared by at least another tunnel in F.

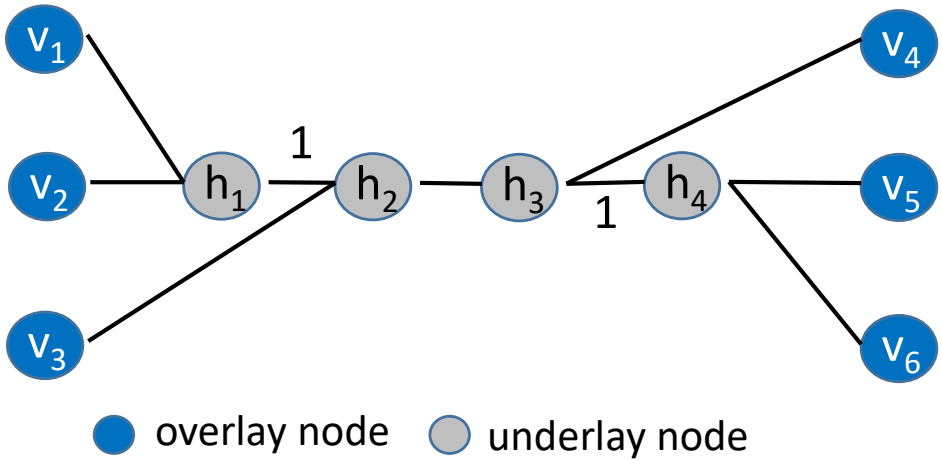


Figure 4.5. Example for estimating the effective category capacity for $F = \{(v_1, v_4), (v_2, v_5), (v_3, v_6)\}$ (suppose that links (h_1, h_2) and (h_3, h_4) have unit capacity, and other links have unlimited capacities).

Even if the subroutine incurs error, Algorithm 12 still achieves a constant-factor approximation under the following condition.

Corollary 4.3.5.1. *If the estimate $\hat{C}_e(\boldsymbol{f})$ for any single-tunnel residual capacity $\tilde{C}_e(\boldsymbol{f})$ satisfies $\tilde{C}_e(\boldsymbol{f}) \geq \hat{C}_e(\boldsymbol{f}) \geq \tilde{C}_e(\boldsymbol{f})/q$, then Algorithm 12 achieves $1/(q \cdot q_F)$-approximation. More precisely, $\tilde{C}_F \geq \hat{C}_F \geq \tilde{C}_F/(q \cdot q_F)$, where q_F is defined in* (4.29).

We note that the approximation ratio in Theorem 4.3.5 is tight, i.e., there exist instances where Algorithm 12 underestimates the effective capacity of a category by a factor arbitrarily close to $1/q_F$. Consider the example in Fig. 4.5. The effective category capacity for $F = \{(v_1, v_4), (v_2, v_5), (v_3, v_6)\}$ is $\tilde{C}_F = 2$, achieved by sending one unit of flow on tunnel (v_1, v_4) and one unit of flow on tunnel (v_3, v_6). Note that the category capacity $C_F = \infty$ as category Γ_F only contains one link (h_2, h_3) which has unlimited capacity. However, following the order of (v_2, v_5), (v_1, v_4), and (v_3, v_6), Algorithm 12 will only obtain $\hat{C}_F = 1$, as assigning a unit flow on tunnel (v_2, v_5) will reduce the residual capacities of the other tunnels to zero. In this case, $q_F = 3$, as tunnel (v_2, v_5) traverses shared links (h_1, h_2), (h_2, h_3), (h_3, h_4) that belong to three different categories. This example can be extended to the scenario where a given tunnel e_0 shares unit-capacity links with each of n other tunnels $e_1, \ldots, e_n$ as well as an unlimited-capacity link with all these tunnels. For $F = \{e_i\}_{i=0}^n$, we have $\tilde{C}_F = n$, $q_F = n + 1$, and $\hat{C}_F = 1$ by Algorithm 12 if following the order of $e_0, e_1, \ldots, e_n$, yielding an approximation ratio of $\hat{C}_F/\tilde{C}_F = 1/n \approx 1/q_F$ for $n \gg 1$.

Remark: Although the above example shows that the greedy procedure of Algorithm 12

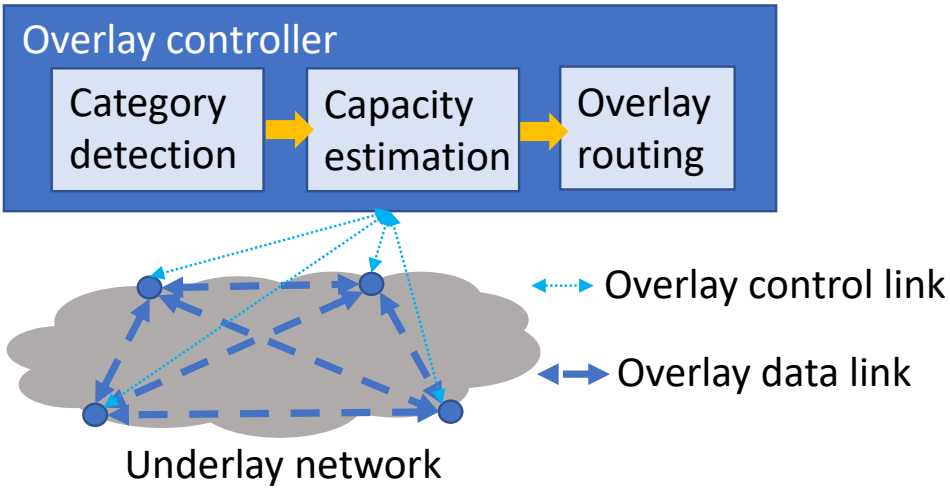


Figure 4.6. Illustration of overall solution.

can lead to substantial underestimation in the worst case, we note that such a worst case is only achieved under a specific topology and a specific order of assigning flows to the tunnels. In practice, we have observed that the worst case rarely occurs and Algorithm 12 is accurate as long as its subroutine for estimating single-tunnel residual capacity is accurate (see Table 4.4).

4.4 Underlay-aware Overlay Routing

We now discuss how to leverage the inference techniques discussed in Section 4.3 in overlay routing.

4.4.1 Overall Solution

By detecting the nonempty categories $\mathcal{F}$ (via Algorithm 9) and estimating the effective category capacities $\{\hat{C}_F\}_{F\in\mathcal{F}}$ (via Algorithm 12), the overlay can generate capacity constraints in the form of

$$\sum_{(i,j)\in F}\sum_{h\in H} d_h x_{ij}^h \le \hat{C}_F, \quad \forall F \in \mathcal{F}, \tag{4.30}$$

and use them in place of (4.1b) in overlay routing optimizations such as (4.1). Fig. 4.6 illustrates the workflow of the overall proposed solution. Note that the centralized controller is just an illustration of the fact that the current work does not focus on the coordination within the overlay (which is left to future work).

4.4.2 Performance Analysis

Both nonempty category identification and category capacity estimation are subject to inference errors, which will affect the accuracy of the generated constraint (4.30) and thus the performance of overlay routing. We now analyze the impact of these errors.

There are four types of inference errors: false alarm/miss in nonempty category identification and under/over-estimation of category capacity. A false alarm in nonempty category identification will cause the generation of a superfluous constraint in overlay routing, which may lead to suboptimal routing decisions. Meanwhile, a miss will cause a constraint to be missing, which may lead to an infeasible routing decision that causes congestion in the underlay. Similarly, an underestimated category capacity will lead to a constraint that is too tight, potentially causing suboptimality, while an overestimated category capacity will lead to a constraint that is too loose, potentially causing congestion. While the extent of suboptimality will depend on the specific routing objective and network instance, which is hard to characterize analytically, the congestion probability can be analyzed in closed form. Specifically, as discussed in Section 4.3.3, the category capacity estimation typically incurs only underestimation errors, which will not cause congestion. Thus, the only cause of congestion is the failure in detecting some nonempty category, the probability of which will decay exponentially in T (#batches of probes for estimating the path metrics) as follows.

Theorem 4.4.1. *Let $\mathcal{F}^* := \{F \subseteq E : \Gamma_F \neq \emptyset\}$ be the true set of nonempty categories. Suppose that the (effective) category capacity estimation is performed by Algorithm 12 with a subroutine for single-tunnel residual capacity estimation that has no overestimation error, and every nonempty category satisfies $w_F > \eta$, where η is the threshold for nonempty category identification. Then under the assumption of Theorem 4.3.3, the probability for the proposed underlay-aware overlay routing to cause congestion is upper-bounded by*

$$\begin{aligned} &|\mathcal{F}^*|\Phi\left(\frac{(\eta - w_{F^*})\sqrt{T} - \tilde{\delta}_{F^*}/\sqrt{T}}{\delta_{F^*}}\right) \\ &\approx \frac{|\mathcal{F}^*|\delta_{F^*}}{(w_{F^*} - \eta)\sqrt{2\pi T}} \exp\left(-\frac{(w_{F^*} - \eta)^2}{2\delta_{F^*}^2} T\right), \end{aligned} \tag{4.31}$$

where δ_F, $\tilde{\delta}_F$, and Φ are defined as in Theorem 4.3.3 (omitting "(E)"), and $F^* := \arg\max_{F \in \mathcal{F}^*} \Pr\{\hat{w}_F \leq \eta\}$.

	AttMpls	AboveNet	GTS-CE	BellCanada
$\lvert\underline{V}\rvert$	25	23	149	48
$\lvert\underline{E}\rvert$	114	62	386	130
$C_{\underline{e}}$ (Gbps)	1	1	1	1
link delays (μs)	[206,4973]	[100, 13800]	[5,1081]	[78, 6160]

Table 4.1. Characteristics of the tested underlay topologies.

4.5 Performance Evaluation

4.5.1 Evaluation Setup

In this section, we will test the proposed solutions via packet-level simulations in NS3, which is a widely used discrete event simulator. To construct diverse and realistic scenarios, we simulate the underlay network according to four real networks from the Internet Topology Zoo [138] with different densities and sizes, and set the link capacities and delays according to [139]. The characteristics of each topology are summarized in Table 4.1.

Each underlay is assumed to follow shortest path routing based on hop count. Following [140], we generate cross traffic on each underlay link according to an ON-OFF process, where the duration of each ON period follows a truncated Pareto distribution, with shape parameter 2.04 and scale/upper-bound parameter set to the minimum/maximum round-trip time (RTT) of the tunnels traversing this link. The duration of each OFF period follows the same distribution with a different scale parameter, configured to yield a link utilization randomly drawn from $[10\%, 40\%]$. Following [141], we randomly draw the sizes of cross-traffic packets from 50, 576, and 1460 bytes with probabilities 0.4, 0.2, and 0.4, respectively. We set the overlay packet size to 50 bytes for probing and 1000 bytes for routing.

To create the overlay, we select 10 nodes with the lowest degree as the overlay nodes while maintaining a pairwise distance of at least two hops, which leads to 90 (directed) overlay tunnels and 2^{90} potential categories. The number of nonempty categories for each topology is given in Table 4.2. As for the demands $\boldsymbol{d} = (d_h)_{h\in H}$ in (4.1), we first generate an initial demand $\boldsymbol{d}_0$ based on the gravity model [142]. Then, we scale it by a factor α to ensure that there exists a routing solution to satisfy $\alpha\boldsymbol{d}_0$ with a given maximum link utilization. All results are the median of 20 Monte Carlo runs with randomized background traffic and demands.

	AttMpls	AboveNet	GTS-CE	BellCanada
#nonempty cat.	68	52	56	52
COIN	22	27	40	28
R-COIN	8	14	32	16
T-COIN	1	1	17	5

Table 4.2. Misses in category identification ($|V| = 10$).

	AttMpls	AboveNet	GTS-CE	BellCanada
#empty cat.	$2^{90} - 68$	$2^{90} - 52$	$2^{90} - 56$	$2^{90} - 52$
COIN	1093	1230	4997	2576
R-COIN	1013	1202	2293	2564
T-COIN	815	1072	1493	828

Table 4.3. False alarms in category identification ($|V| = 10$).

4.5.2 Benchmarks

We evaluate the following solutions:

1. "Agnostic": an underlay-agnostic solution that treats all the tunnels as independent logical links, i.e., ignoring their link sharing in the underlay;

2. "LCC": the state-of-the-art solution from [109], where we make two optimistic assumptions: (i) perfect clustering of the detected flows based on their shared dominant bottlenecks, and (ii) improved accuracy in the capacity constraints based on the residual capacities instead of the total capacities as originally estimated in [109];

3. "x-COIN": our proposed solution as depicted in Section 4.4.1, which contains three variations differentiated by the adopted category identification algorithms (i.e., COIN, R-COIN, T-COIN).

4.5.3 Evaluation Results

4.5.3.1 Nonempty Category Identification

We estimate ρ_F as in (4.17), with implementation details presented in Appendix 4.7.2. A category Γ_F is detected to be nonempty if its estimated metric satisfies $\hat{w}_F > \eta$. As our estimation of the effective category capacities is accurate (see Table 4.4), false alarms will not hurt overlay routing, and thus we set η to a small value (10^{-10} in our simulation) to minimize misses. The resulting numbers of false alarms/misses are given in Table 4.2-4.3, with each Monte Carlo run containing 2×10^4 batches of probes.

	AttMpls	AboveNet	GTS-CE	BellCanada
ideal subroutine	0.10%	0.13%	0.13%	0.4%
pathload	1.07%	1.18%	1.15%	1.49%

Table 4.4. Errors in effective category capacity estimation.

Despite the large number of false alarms, the false alarm rate is very low due to the exponentially many categories that are empty. Meanwhile, the miss rates are non-negligible, although T-COIN outperforms R-COIN, which in turn outperforms COIN. Such errors come from two causes: (i) for tunnels with different sources, probes in the same batch may arrive at a shared link at different times and experience different queueing delays; (ii) a link shared by a large number of tunnels will receive many probes in a batch, where the earlier probes will experience different queueing delays from the later ones. In this regard, T-COIN performs much better since it only requires the estimation of ρ_F for F containing two tunnels with the same source.

4.5.3.2 Category Capacity Estimation

Next, we evaluate the normalized mean absolute error ($|\hat{C}_F - \tilde{C}_F|/\tilde{C}_F$) of Algorithm 12. To separate the impact of errors in its subroutine, we evaluate two versions of Algorithm 12, one using the true value of $C_{e_{i_j}}(\boldsymbol{f})$ in Line 5 (i.e., ideal subroutine) and the other using the estimated $\hat{C}_{e_{i_j}}(\boldsymbol{f})$ obtained from *pathload*[3] [130]. The results averaged over 20 Monte Carlo runs are given in Table 4.4. Surprisingly, Algorithm 12 can estimate the effective category capacities almost perfectly when the subroutine estimates the single-tunnel residual capacities correctly, indicating that the worst-case ratio in Theorem 4.3.5 is rarely achieved. Slightly more error is incurred when using a realistic subroutine, but the overall estimation remains highly accurate.

4.5.3.3 Approximation of Feasible Region

Despite the large numbers of misses and false alarms, the feasible region induced by the inferred capacity constraint (4.30) may still approximate the true feasible region induced by (4.1b) (equivalently (4.5)), if the superfluous constraints caused by false alarms have similar effect as the missing constraints caused by misses. Denote the true feasible region of the rates through the tunnels as $\mathcal{P} := \{\boldsymbol{y} \geq \boldsymbol{0} : \sum_{(i,j)\in E:\underline{e}\in\underline{p}_{i,j}} y_{ij} \leq C_{\underline{e}},\ \forall \underline{e} \in \underline{E}\}$, and the inferred feasible region as $\hat{\mathcal{P}} := \{\boldsymbol{y} \geq \boldsymbol{0} : \sum_{(i,j)\in F} y_{ij} \leq \hat{C}_F,\ \forall F \in \mathcal{F}\}$. We define

[3]Pathload is an adaptive algorithm that sends a train of probes at a time and tunes its rate to measure the residual capacity. In our simulation, the train length is set to 5000 probes, but the total number of trains is a variable in $[2, 15]$.

$\hat{\mathcal{P}}$ similarly for each of the benchmarks. We measure the consistency between these two regions by randomly sampling extreme points from one region and calculating the maximum constraint violation for the other region. We observe that the extreme points of $\mathcal{P}$ almost always satisfy the constraints of $\hat{\mathcal{P}}$ for all the solutions (omitted), but the extreme points of $\hat{\mathcal{P}}$ can violate the constraints of $\mathcal{P}$ (i.e., causing congestion), as shown in Fig. 4.7. We see from Fig. 4.7 that (i) the constraint violation of "Agnostic" is most severe, and (ii) our proposed solutions (COIN, R-COIN and T-COIN) can all notably reduce the constraint violation compared to both "Agnostic" and "LCC", where T-COIN achieves nearly zero constraint violation on the first two topologies due to its low miss rate in these cases. Despite the notable inference errors, *our solution can characterize the feasible region for overlay routing much more accurately than the existing solutions.*

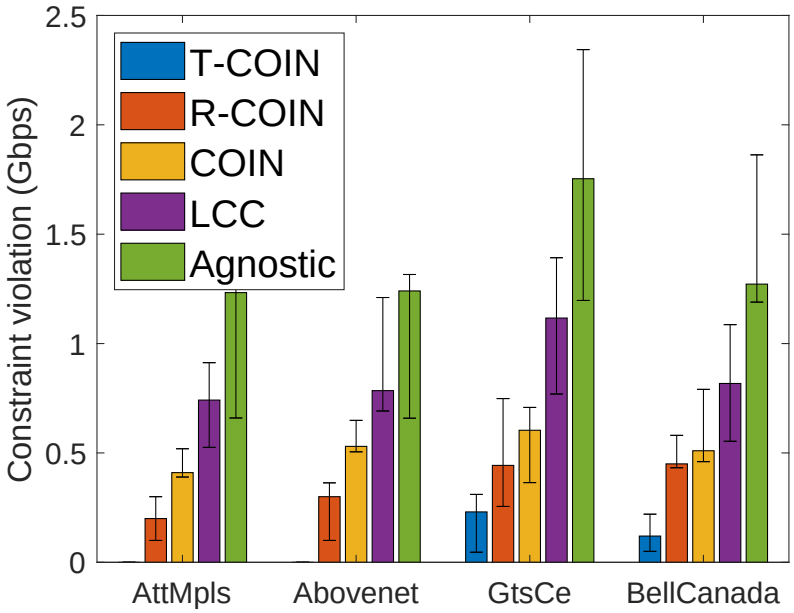


Figure 4.7. Constraint violation for randomly-sampled extreme points of the estimated feasible region.

4.5.3.4 Performance of Overlay Routing

When the demands are sufficiently light such that even "Agnostic" does not encounter any congestion, there is no need to consider link sharing among tunnels and all the overlay routing solutions achieve similar performance (omitted). We thus focus on scenarios where at least one link will be congested under one of the tested routing solutions, by scaling the demands to achieve (i) a maximum link utilization of 90% under the optimal

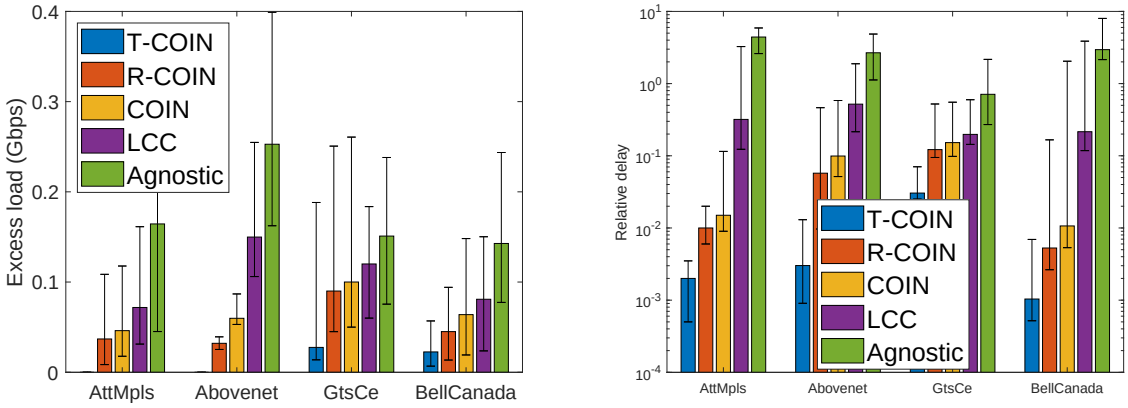
routing (with perfect knowledge about the underlay) and (ii) a maximum excess load of 10% under "Agnostic". We evaluate the performance of minimum cost overlay routing in terms of both congestion and routing cost. Here, we set the routing cost c_{ij} for each tunnel $(i, j) \in E$ to the sum (propagation) delay of the links traversed by this tunnel.

To measure congestion, we evaluate the maximum load on any underlay link in excess of its capacity. The result in Fig. 4.8(a) shows that "Agnostic" incurs the most congestion due to ignoring link sharing among the tunnels, followed by "LCC" that only considers a subset of the capacity constraints corresponding to the bottlenecks shared by tunnels with the same destination. Due to their better accuracy in approximating the feasible region (Fig. 4.7), our solutions can substantially outperform the state of the art ("LCC"), particularly with the knowledge of symmetric tree-based routing in the underlay ("T-COIN"). Among the underlay topologies, we observe more improvement for the topologies (AttMpls, Abovenet, BellCanada) supporting rich overlay routing choices; in contrast, the topology GtsCe yields less improvement as all the overlay nodes in this topology reside at degree-1 nodes and hence their incident links become the common bottleneck under all the overlay routing solutions. These observations also apply to the sum of excess loads.

To measure routing cost, we simulate overlay routing for 20,000 milliseconds and measure the average end-to-end delay over all the received packets. We then normalize the average delay to enable comparison across topologies. Given the average delay $\bar{\phi}$ obtained from simulation, we evaluate the relative delay $(\bar{\phi} - \bar{\phi}_0)/\bar{\phi}_0$, where $\bar{\phi}_0$ is the average delay under the optimal routing solution based on perfect knowledge about the underlay. The result in Fig. 4.8(b) confirms that (i) underlay-aware overlay routing (our solutions and "LCC") can notably outperform underlay-agnostic overlay routing ("Agnostic"), and (ii) by inferring the key information to characterize the feasible region, our solutions, especially T-COIN, can substantially improve the performance compared to the state of the art ("LCC").

4.6 Conclusion

We studied the problem of overlay routing over an uncooperative underlay with unknown topology and link capacities. We identified the minimum information needed to achieve congestion-free overlay routing, and then developed polynomial-complexity algorithms to infer this information with guaranteed accuracy. Our NS3 simulations based on realistic settings demonstrated the superior performance of our algorithms in characterizing the feasible region and improving the performance of overlay routing. Our solution paves the way for overlay-based communication optimization without cooperation from the



(a) max excess load (b) avg relative delay

Figure 4.8. Performance of overlay routing.

underlying network infrastructure.

4.7 Appendix

4.7.1 Supporting Proofs

Proof of Lemma 4.3.1. This is a direct consequence of the property that different categories are disjoint, as implied by Definition 4.3.1. □

Proof of Lemma 4.3.2. By Assumption 2, $w_F(E') = 0$ implies that $\Gamma_F(E') = \emptyset$. This means that either (i) no link is traversed by all the tunnels in F, or (ii) every link traversed by F is also traversed by at least another tunnel in $E' \setminus F$. In case (i), categories $\Gamma_F(E' \cup \{e\})$ and $\Gamma_{F\cup\{e\}}(E' \cup \{e\})$ must both be empty, as any link in either of these categories must be traversed by all the tunnels in F. The same conclusion holds in case (ii), as any link in either $\Gamma_F(E' \cup \{e\})$ or $\Gamma_{F\cup\{e\}}(E' \cup \{e\})$ must be traversed by all the tunnels in F but none of the tunnels in $E' \setminus F$. Thus, $w_F(E' \cup \{e\}) = w_{F\cup\{e\}}(E' \cup \{e\}) = 0$. □

Proof of Lemma 4.3.3. First, by the same argument as in the proof of Lemma 4.3.2, any $\underline{e} \notin \Gamma_F(E')$ must not be in either $\Gamma_F(E' \cup \{e\})$ or $\Gamma_{F\cup\{e\}}(E' \cup \{e\})$. Moreover, by the definition of category, any $\underline{e} \in \Gamma_F(E')$ must be in $\Gamma_F(E' \cup \{e\})$ if $\underline{e}$ is not traversed by tunnel e and in $\Gamma_{F\cup\{e\}}(E' \cup \{e\})$ if it is traversed by tunnel e. Thus, $\Gamma_F(E') = \Gamma_F(E' \cup \{e\}) \cup \Gamma_{F\cup\{e\}}(E' \cup \{e\})$, which implies the conclusion. □

Proof of Lemma 4.3.4. We prove by contradiction. Suppose that we have $w_F(E') > 0$ and $E_t := E' \cup \{e_1, \cdots, e_t\}$ where $E' \cap \{e_1, \cdots, e_t\} = \emptyset$. We will iteratively add

$e_i, i = 1, \cdots, t$ into E'. In the first iteration, according to Lemma 4.3.3, we have $w_F(E') = w_F(E_1) + w_{F\cup\{e_1\}}(E_1)$ where $E_1 := E' \cup \{e_1\}$. Then, at least one of $w_F(E_1)$ and $w_{F\cup\{e_1\}}(E_1)$ has non-zero value. In other words, there exists $F' \subseteq \text{supp}\,(\boldsymbol{w}(E_1))$ such that $F \subseteq F'$. Similarly, in adding e_2 to have $E_2 := E_1 \cup \{e_2\}$, at least one of $w_F(E_2)$, $w_{F\cup\{e_2\}}(E_2)$, $w_{F\cup\{e_1\}}(E_2)$ and $w_{F\cup\{e_1,e_2\}}(E_2)$ must have non-zero values by applying Lemma 4.3.3 to $w_F(E_1)$ and $w_{F\cup\{e_1\}}(E_1)$. By repeatedly applying Lemma 4.3.3 when adding e_j to $E_{j-1}, j = 2, \cdots, t$, there always exists $F' \subseteq \text{supp}\,(\boldsymbol{w}(E_j))$ such that $F \subseteq F'$, which introduces contradiction and completes the proof. □

Proof of Lemma 4.3.5. Suppose that contrary to the lemma, there exists $F_0 \subseteq E_0$ and $F_1 \subseteq E_1$ such that $w_{F_1}(E_1) = 0$ but $w_{F_0\cup F_1}(E_0 \cup E_1) > 0$. Then there must exist at least one underlay link $\underline{e}$ with non-zero metric that is traversed by all the tunnels in F_1 but none of the tunnels in $E_1 \setminus F_1$, because $E_0 \cap E_1 = \emptyset$. However, by Definition 4.3.1, $\underline{e}$ must belong to $\Gamma_{F_1}(E_1)$, and thus $w_{F_1}(E_1) > 0$, contradicting with the assumption. Similar contradiction can be derived if $w_{F_0}(E_0) = 0$ but $w_{F_0\cup F_1}(E_0 \cup E_1) > 0$. □

Proof of Theorem 4.3.2. We first notice that $E_t = \left(E_t \setminus E^{(s)}\right) \bigcup \left(E_t \cap E^{(s)}\right)$, where $\left(E_t \setminus E^{(s)}\right) \bigcap \left(E_t \cap E^{(s)}\right) = \emptyset$. Similarly, we have the following decomposition $F' = \left(F' \setminus E^{(s)}\right) \bigcup \left(F' \cap E^{(s)}\right)$. According to Lemma 4.3.4, we must have $w_{F'\cap E^{(s)}}(E_t \cap E^{(s)}) = 0$ if there is no $F'' \subseteq \text{supp}\Big(\boldsymbol{w}(E^{(s)})\Big)$ such that $\left(F' \cap E^{(s)}\right) \subseteq F''$. Consequently, according to Lemma 4.3.5, we must have $w_{F'}(E_t) = 0$ by setting E_0 and E_1 to $E_t \setminus E^{(s)}$ and $E_t \cap E^{(s)}$ respectively, which completes the proof. □

Proof of Lemma 4.3.6. We prove (4.18) by performing induction on $|E|$ and leveraging the relationships derived in Algorithm 9. First, for $|E| = 2$ (denoted by $E = \{e_1, e_2\}$), it is easy to see that $w_{\{e_i\}}(E) = \rho_E - \rho_{E\setminus\{e_i\}}$ $(i = 1, 2)$ and $w_E(E) = \rho_{e_1} + \rho_{e_2} - \rho_E$, satisfying (4.18). Now consider $|E| > 2$ and suppose that (4.18) is satisfied by $w_F(E')$ for any $|E'| \leq |E| - 1$. Let $E' \coloneqq E \setminus \{e\}$ for any $e \in E$. By line 4,

$$w_{\{e\}}(E) = \rho_E - \rho_{E'} = (-1)\rho_{E\setminus\{e\}} + (-1)^2\rho_{(E\setminus\{e\})\cup\{e\}}, \tag{4.32}$$

satisfying (4.18). Now consider $F \subseteq E'$ with $F \neq \emptyset$. Suppose that $w_{F'\cup\{e\}}(E)$ satisfies (4.18) for any $|F'| < |F|$. By line 6,

$$\begin{aligned} w_{F\cup\{e\}}(E) = {} & \rho_{(E'\setminus F)\cup\{e\}} - \rho_{E'\setminus F} - \rho_E + \rho_{E'} \\ & - \sum_{F'\subset F, F'\neq\emptyset} \sum_{F''\subseteq F'\cup\{e\}} (-1)^{|F''|+1}\rho_{(E'\setminus F')\cup F''}. \end{aligned} \tag{4.33}$$

The last term in (4.33) can be decomposed into the summation of the following two terms

$$\sum_{F'\subset F, F'\neq\emptyset} \sum_{F''\subseteq F'} (-1)^{|F''|+1} \rho_{(E'\setminus F)\cup((F\setminus F')\cup F'')}, \tag{4.34}$$

$$\sum_{F'\subset F, F'\neq\emptyset} \sum_{F''\subseteq F'} (-1)^{|F''|} \rho_{(E'\setminus F)\cup((F\setminus F')\cup F''\cup\{e\})}. \tag{4.35}$$

With $F_0 \coloneqq (F \setminus F') \cup F''$, we can rewrite (4.34) as

$$\begin{aligned}
&\sum_{F_0\subset F, F_0\neq\emptyset} \rho_{(E'\setminus F)\cup F_0} \sum_{F''\subset F_0} (-1)^{|F''|+1} \\
&\qquad\qquad + \rho_{(E'\setminus F)\cup F} \sum_{F''\subset F, F''\neq\emptyset} (-1)^{|F''|+1} \\
&= \sum_{F'\subseteq F, F'\neq\emptyset} (-1)^{|F'|} \rho_{(E'\setminus F)\cup F'} + \rho_{E'},
\end{aligned} \tag{4.36}$$

where we have plugged in $\sum_{F''\subset F_0}(-1)^{|F''|+1} = \sum_{i=0}^{|F_0|-1} \binom{|F_0|}{i}(-1)^{i+1} = (-1)^{|F_0|}$ and $\sum_{F''\subset F, F''\neq\emptyset}(-1)^{|F''|+1} = \sum_{i=1}^{|F|-1} \binom{|F|}{i}(-1)^{i+1} = (-1)^{|F|} + 1$. Similarly, we can rewrite (4.35) as

$$\begin{aligned}
&\sum_{F_0\subset F, F_0\neq\emptyset} \rho_{(E'\setminus F)\cup F_0\cup\{e\}} \sum_{F''\subset F_0} (-1)^{|F''|} \\
&\qquad\qquad + \rho_{(E'\setminus F)\cup F\cup\{e\}} \sum_{F''\subset F, F''\neq\emptyset} (-1)^{|F''|} \\
&= \sum_{F'\subseteq F, F'\neq\emptyset} (-1)^{|F'|+1} \rho_{(E'\setminus F)\cup F'\cup\{e\}} - \rho_E.
\end{aligned} \tag{4.37}$$

Plugging (4.36) and (4.37) into (4.33) yields

$$\begin{aligned}
w_{F\cup\{e\}}(E) &= \rho_{(E'\setminus F)\cup\{e\}} - \rho_{E'\setminus F} - \rho_E + \rho_{E'} \\
&\quad + \sum_{F'\subseteq F, F'\neq\emptyset} (-1)^{|F'|+1} \rho_{(E'\setminus F)\cup F'} - \rho_{E'} \\
&\quad + \sum_{F'\subseteq F, F'\neq\emptyset} (-1)^{|F'\cup\{e\}|+1} \rho_{(E'\setminus F)\cup F'\cup\{e\}} + \rho_E \\
&= \sum_{F'\subseteq F} (-1)^{|F'|+1} \rho_{(E'\setminus F)\cup F'} \\
&\quad + \sum_{F'\subseteq F} (-1)^{|F'\cup\{e\}|+1} \rho_{(E'\setminus F)\cup(F'\cup\{e\})},
\end{aligned} \tag{4.38}$$

which satisfies (4.18). Finally, by plugging in the expression for $w_F(E')$ (from the

induction assumption) and the expression (4.38) for $w_{F\cup\{e\}}(E)$ into the formula in line 7, we have

$$\begin{aligned} w_F(E) &= w_F(E') - w_{F\cup\{e\}}(E) \\ &= \sum_{F'\subseteq F} (-1)^{|F'|+1}\rho_{(E'\setminus F)\cup F'} - \sum_{F'\subseteq F} (-1)^{|F'|+1}\rho_{(E'\setminus F)\cup F'} - \sum_{F'\subseteq F} (-1)^{|F'|}\rho_{(E'\setminus F)\cup F'\cup\{e\}} \\ &= \sum_{F'\subseteq F} (-1)^{|F'|+1}\rho_{(E\setminus F)\cup F'} \end{aligned} \tag{4.39}$$

for any $F \subseteq E'$ and $F \neq \emptyset$, which also satisfies (4.18). Together, (4.32), (4.38), and (4.39) complete the induction. □

Proof of Lemma 4.3.7. By Taylor expansion of $-\log \overline{S}_F$ at s_F, we have

$$\hat{\rho}_F = -\log s_F - \frac{\overline{S}_F - s_F}{s_F} + \frac{(\overline{S}_F - s_F)^2}{2s_F^2} + O((\overline{S}_F - s_F)^3).$$

By ignoring the high-order terms $O((\overline{S}_F - s_F)^3)$, we have

$$\mathbb{E}[\hat{\rho}_F] \approx -\log s_F - \frac{\mathbb{E}[\overline{S}_F] - s_F}{s_F} + \frac{\mathrm{var}[\overline{S}_F]}{2s_F^2} = \rho_F + \frac{1-s_F}{2s_F T},$$

where we have plugged in $\rho_F = -\log s_F$, $\mathbb{E}[\overline{S}_F] = s_F$, and $\mathrm{var}[\overline{S}_F] = s_F(1-s_F)/T$. Similarly, we have

$$\begin{aligned} \mathrm{var}[\hat{\rho}_F] &\approx \mathbb{E}\left[\Big(-\frac{\overline{S}_F - s_F}{s_F} + \frac{(\overline{S}_F - s_F)^2}{2s_F^2} - \frac{1-s_F}{2s_F T}\Big)^2\right] \\ &\approx \frac{\mathrm{var}[\overline{S}_F]}{s_F^2} + \frac{(1-s_F)^2}{4s_F^2T^2} - \frac{(1-s_F)\mathrm{var}[\overline{S}_F]}{2s_F^3 T} \\ &= \frac{1-s_F}{s_F T} + O(\frac{1}{T^2}), \end{aligned} \tag{4.40}$$

where we have ignored terms of the order $O(\mathbb{E}[(\overline{S}_F - s_F)^i])$ with $i > 2$ in (4.40). □

Proof of Theorem 4.3.3. By the independent Gaussian assumption for $\{\hat{\rho}_F - \rho_F\}_{F\subseteq E, F\neq\emptyset}$ and (4.18), $\hat{w}_F(E) - w_F(E)$ has a Gaussian distribution with mean

$$\begin{aligned} &\sum_{F': E\setminus F\subseteq F'} (-1)^{|F'|-|E\setminus F|+1}(\mathbb{E}[\hat{\rho}_{F'}] - \rho_{F'}) \\ &= \sum_{F': E\setminus F\subseteq F'} (-1)^{|F'|-|E\setminus F|+1}\frac{(1-s_{F'})}{2s_{F'}T} =: \frac{\tilde{\delta}_F(E)}{T}, \end{aligned} \tag{4.41}$$

and variance

$$\sum_{F':E\setminus F\subseteq F'}((-1)^{|F'|-|E\setminus F|+1})^2\mathrm{var}[\hat{\rho}_{F'}]$$
$$=\sum_{F':E\setminus F\subseteq F'}\frac{1-s_{F'}}{s_{F'}T}=:\frac{\delta_F(E)^2}{T}. \tag{4.42}$$

Thus, if $w_F(E)=0$, we have

$$\begin{aligned}\Pr\{\hat{w}_F(E)>\eta\}&=\Pr\{\hat{w}_F(E)-w_F(E)>\eta\}\\&=1-\Phi\left(\frac{\eta-\tilde{\delta}_F(E)/T}{\sqrt{\delta_F(E)^2/T}}\right),\end{aligned} \tag{4.43}$$

which proves (4.22). Moreover, as $1-\Phi(x)\approx e^{-x^2/2}/(x\sqrt{2\pi})$ for $x\gg 1$, for $T\gg 1$, we have

$$(4.43)\approx 1-\Phi\left(\frac{\eta\sqrt{T}}{\delta_F(E)}\right)\approx\frac{\delta_F(E)}{\eta\sqrt{2\pi T}}\exp\left(-\frac{\eta^2}{2\delta_F(E)^2}T\right), \tag{4.44}$$

which proves (4.23). Similarly, if $w_F(E)>\eta$, we have

$$\begin{aligned}\Pr\{\hat{w}_F(E)\le\eta\}&=\Pr\{\hat{w}_F(E)-w_F(E)\le\eta-w_F(E)\}\\&=\Phi\left(\frac{\eta-w_F(E)-\tilde{\delta}_F(E)/T}{\sqrt{\delta_F(E)^2/T}}\right)\\&\approx\Phi\left(\frac{(\eta-w_F(E))\sqrt{T}}{\delta_F(E)}\right).\end{aligned} \tag{4.45}$$

Moreover, as $\Phi(x)=1-\Phi(-x)\approx -e^{-x^2/2}/(x\sqrt{2\pi})$ for $x\ll 0$, we have (for $T\gg 1$)

$$(4.45)\approx\frac{\delta_F(E)}{(w_F(E)-\eta)\sqrt{2\pi T}}\exp\left(-\frac{(\eta-w_F(E))^2}{2\delta_F(E)^2}T\right), \tag{4.46}$$

which proves (4.25). □

Proof of Theorem 4.3.4. It suffices to show that Algorithm 11 correctly detects all the nonempty categories on a given tunnel (s,t).

First, we argue that lines 4–12 correctly locate all the branching/joining points between (s,t) and the other tunnels. This is easy to see for the tunnels sharing the starting point (lines 4–5) or the ending point with (s,t) (lines 6–7). Any other tunnel

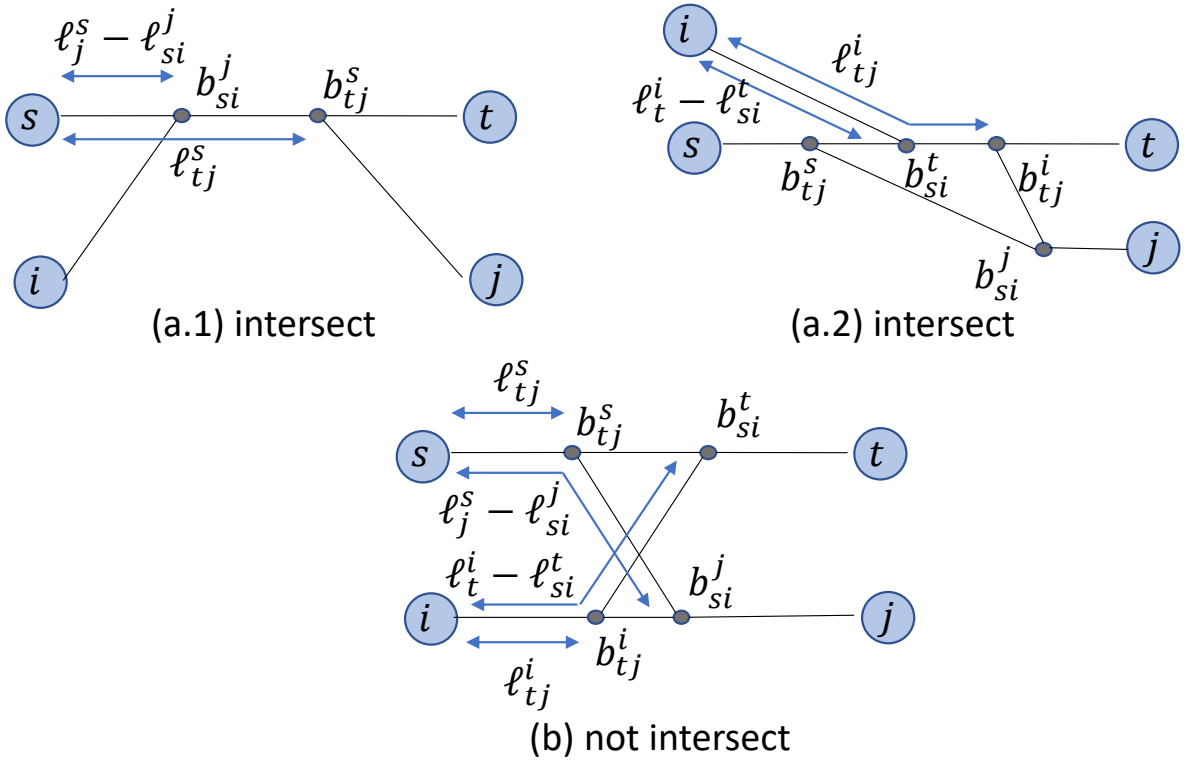


Figure 4.9. Cases for two tunnels without common endpoints.

(i, j) may or may not intersect with (s, t). Under the assumption (i) in the theorem, if they intersect, then the branching/joining points of their intersection must coincide with the branching points of some 1-by-2 components formed by $\{s, t, i, j\}$. As illustrated in Fig. 4.9 (a.1), if $\ell^s_{tj} > \ell^s_j - \ell^j_{si}$, then there must be a joining point at distance $\ell^s_j - \ell^j_{si}$ and a branching point at distance ℓ^s_{tj} from s on path $\underline{p}_{s,t}$, as recorded in line 10. If, instead, $\ell^i_{tj} > \ell^i_t - \ell^t_{si}$, then as illustrated in Fig. 4.9 (a.2), there will be a joining point at distance $\ell^s_t - \ell^t_{si}$ and a branching point at distance $\ell^s_t - (\ell^i_t - \ell^i_{tj})$ from s on path $\underline{p}_{s,t}$, as recorded in line 12. By symmetry of the tunnels (line 2), any intersection between (s, t) and (i, j) that satisfies $\ell^t_{si} > \ell^t_i - \ell^i_{tj}$ or $\ell^j_{si} > \ell^j_s - \ell^s_{tj}$ will also be detected when considering tunnels (t, s) in line 3 and (j, i) in line 8. If none of the above detects any intersection between tunnels (s, t) and (i, j), i.e., $\ell^s_{tj} \leq \ell^s_j - \ell^j_{si}$, $\ell^i_{tj} \leq \ell^i_t - \ell^t_{si}$, $\ell^t_{si} \leq \ell^t_i - \ell^i_{tj}$, and $\ell^j_{si} \leq \ell^j_s - \ell^s_{tj}$, then tunnels (i, j) and (s, t) must not intersect as illustrated in Fig. 4.9 (b). This is because: (i) the intersection cannot occur on the path segment $s \to b^s_{tj}$, as (i, j) must be disjoint from (s, j) before b^j_{si}, and b^s_{tj} must be before b^j_{si} (i.e., closer to s) due to $\ell^s_{tj} \leq \ell^s_j - \ell^j_{si}$, and similarly the intersection cannot occur on the path segments $b^t_{si} \to t$, $i \to b^i_{tj}$, or $b^j_{si} \to j$; (ii) the intersection cannot occur between the path segments $b^s_{tj} \to b^t_{si}$ and $b^i_{tj} \to b^j_{si}$ either, as otherwise the resulting branching/joining points will not coincide

with the branching point of any 1-by-2 component formed by $\{s, t, i, j\}$ as illustrated in Fig. 4.4, violating assumption (i).

Next, since all the links between consecutive branching/joining points belong to the same category, Algorithm 11 will discover all the categories appearing on (s, t) by going through these points from s to t, while updating the set F to track the set of tunnels traversing the links (if any) between the last and the current branching/joining points (lines 13–21).

Finally, as the distance between consecutive branching/joining points equals the sum metric of the involved links and every link metric $> \eta$, the condition in line 22 will be satisfied for every nonempty category on (s, t). This completes the proof. □

Proof of Theorem 4.3.5. We first prove the approximation ratio for a discretized version of (4.28). Suppose that we can only assign flow in integer multiples of a constant $\delta > 0$. This discretization converts (4.28) to a set function maximization problem as follows:

$$\max_{(X_e)_{e\in F}} \delta \sum_{e\in F} |X_e| \tag{4.47a}$$

$$\text{s.t. } \delta \sum_{e\in F\cap F'} |X_e| \le C_{F'},\ \forall F' \subseteq E, \Gamma_{F'} \neq \emptyset, F' \cap F \neq \emptyset, \tag{4.47b}$$

where X_e denotes the set of δ-flows assigned to tunnel e. In this conversion, we have used the simplification to (4.28) that restricts the nonzero variables to $(f_e)_{e\in F}$ (as f_e for $e \in E \setminus F$ does not contribute to the objective function), and ignores the constraints corresponding to F' with $F' \cap F = \emptyset$ (as $\sum_{e'\in F'} f_{e'} \equiv 0$ when $f_e \equiv 0$ for all $e \in E \setminus F$). According to the definitions in [143], the objective function (4.47a) is a *modular function*, as each item (i.e., a δ-flow) contributes a fixed value to the objective function regardless of the selection of other items. Moreover, the constraints (4.47b) form a q_F*-system* for q_F defined in (4.29), as to add an item into X_e for some $e \in F$, we need to remove at most q_F existing items, each representing a δ-flow on some tunnel e' for $e' \in F'$ to avoid violating the constraint (4.47b) associated with F'. For maximizing a modular function subject to a q_F-system constraint, the greedy algorithm that incrementally adds one item at a time to maximize the increase in the objective function within the constraint is guaranteed to achieve a $1/q_F$-approximation [143].

We then argue that Algorithm 12 essentially implements the greedy algorithm. In our problem, all feasible δ-flows yield the same increase in the objective function (4.47a), and thus it suffices for the greedy algorithm to go through $e \in F$ sequentially, assigning as many δ-flows to a tunnel as possible before moving to the next tunnel. Letting $\delta \to 0$

will reduce this sequential version of the greedy algorithm to lines 2–4 in Algorithm 12. The $1/q_F$-approximation remains valid as it does not depend on the value of δ. □

Proof of Corollary 4.3.5.1. First, as the subroutine only underestimates the residual capacities, we have $\tilde{C}_F \geq \hat{C}_F$ as argued before Theorem 4.3.5. Let $\hat{C}'_F$ denote the estimate by an ideal version of Algorithm 12, where the estimation of residual capacity in lines 2 and 4 is accurate. Then under the condition in Corollary 4.3.5.1, the actual estimate by Algorithm 12 satisfies $\hat{C}_F \geq \hat{C}'_F/q$. Meanwhile, by Theorem 4.3.5, $\hat{C}'_F \geq \tilde{C}_F/q_F$. Thus, $\hat{C}_F \geq \tilde{C}_F/(q \cdot q_F)$, completing the proof. □

Proof of Theorem 4.4.1. Under the conditions of the theorem, congestion can only occur if there exists a nonempty category that is missed by nonempty category detection. We can bound the probability of this event by

$$\Pr\{\exists F \in \mathcal{F}^*, \hat{w}_F \leq \eta\} \leq \sum_{F \in \mathcal{F}^*} \Pr\{\hat{w}_F \leq \eta\} \tag{4.48}$$

$$\leq |\mathcal{F}^*| \Pr\{\hat{w}_{F^*} \leq \eta\} \tag{4.49}$$

$$= |\mathcal{F}^*| \Phi\left(\frac{(\eta - w_{F^*})\sqrt{T} - \tilde{\delta}_{F^*}/\sqrt{T}}{\delta_{F^*}}\right) \tag{4.50}$$

$$\approx \frac{|\mathcal{F}^*|\delta_{F^*}}{(w_{F^*} - \eta)\sqrt{2\pi T}} \exp\left(-\frac{(w_{F^*} - \eta)^2}{2\delta_{F^*}^2} T\right), \tag{4.51}$$

where (4.48) is by union bound, and (4.50)–(4.51) are by Theorem 4.3.3. □

4.7.2 Supplementary Details

In this section, we will present the implementation details on estimating ρ_F as in (4.17). In the sequel, we denote the T end-to-end delay measurements on tunnel $e \in E$ as $\{\phi_{e_1}^t\}_{t=1}^T$.

The first issue is to determine whether a packet has experienced links with "bad status". We consider a packet as in "good status" on a tunnel if its end-to-end delay is below the mean of the 100 smallest delays on this tunnel plus 0.5 ms. Based on this detection criteria, the end-to-end delay measurements $\{\phi_{e_1}^t\}_{t=1}^T$ can be detected to be a binary sequence $\{q_e^t\}_{t=1}^T$, where $q_e^t = 0$ if the t-th measurement is detected to experience links with "bad status" and $q_{e_1}^t = 1$ otherwise.

With $\{q_e^t\}_{t=1}^T$, ρ_F for Alg. 11 (T-COIN) can be calculated as

$$\hat{\rho}_F = -\log\left(\frac{\sum_{t=1}^T q_{e_1}^t q_{e_2}^t}{T}\right), \tag{4.52}$$

since 1-by-2 structure contains only two tunnels sharing the same source.

For COIN and R-COIN, another issue is to calibrate measurements for tunnels in F with various sources so that the measurements can mimic the multicast probing. To begin with, we present the adopted heuristic for calibrating two tunnels. Suppose that we have $\{\phi_{e_1}^t\}_{t=1}^T$ and $\{q_{e_1}^t\}_{t=1}^T$ for $e_1 = (u_1, v_1)$. Similarly, we have measurements $\{\phi_{e_2}^t\}_{t=1}^T$ and the associated $\{q_{e_2}^t\}_{t=1}^T$ for tunnel $e_2 = (u_2, v_2)$. To calculate $\hat{\rho}_F$ as in (4.17) through mimicking multicast probes [113–115], we need to find an offset κ_{e_1,e_2} so that the i-th probe on e_1 and the $(i + \kappa_{e_1,e_2})$-th probe on e_2 traverse the shared links (if any) at approximately the same time. To this end, we maximize the correlation between $\{q_{e_1}^t\}_{t=1}^T$ and $\{q_{e_2}^t\}_{t=1}^T$ by solving

$$\kappa^* = \underset{1-T\le\kappa\le T-1}{\arg\max} \frac{1}{\min(T, T-\kappa) - \max(1, 1-\kappa) + 1} \sum_{i=\max(1,1-\kappa)}^{\min(T,T-\kappa)} q_{e_1}^i q_{e_2}^{i+\kappa}. \tag{4.53}$$

We then identify the i-th packet on e_1 and the $(i + \kappa^*)$-th packet on e_2 as a mimicked multicast.

Then, we will use an example to show that the method above is only applicable to the case with two tunnels. As shown in Fig. 4.10, suppose that we have three tunnels, which are $e_1 : u_1 \to u_2 \to u_3 \to u_4$, $e_2 : u_1 \to u_2 \to u_5 \to u_6$ and $e_3 : u_7 \to u_2 \to u_5 \to u_3 \to u_4$. We can observe that (i) each pair of tunnels have a shared link, and (ii) κ_{e_1,e_2}, κ_{e_1,e_3} and κ_{e_2,e_3} may not be the same. In other words, there may not exist a common κ for more than two tunnels.

Next, we will present the employed heuristic to obtain $\hat{\rho}_F$ based on the obtained $\kappa_{e,e'}, \forall e, e' \in E$ if $|F| \ge 3$. Denote $n_b(F)$ as the number of events that $S_F = 1$ transits to $S_F = 0$ out of T measurements on tunnels in F, we approximate ρ_F as $\hat{\rho}_F = \frac{n_b(F)}{T}$. The challenge is to find $n_b(F)$, which requires one and only one count of each event without repeating. In other words, a status transition event on link $\underline{e}$ will be counted only once even if it may be observed on all tunnels traversing $\underline{e}$.

Formally, q_e^t is detected to experience a status transition event on tunnel e if $\phi_e^{t-1} < \phi_e^t$ and $q_e^t = 0$. Then, two probes $q_{e_0}^{t_0}$ and $q_{e_1}^{t_1}$ are identified to experience the same status

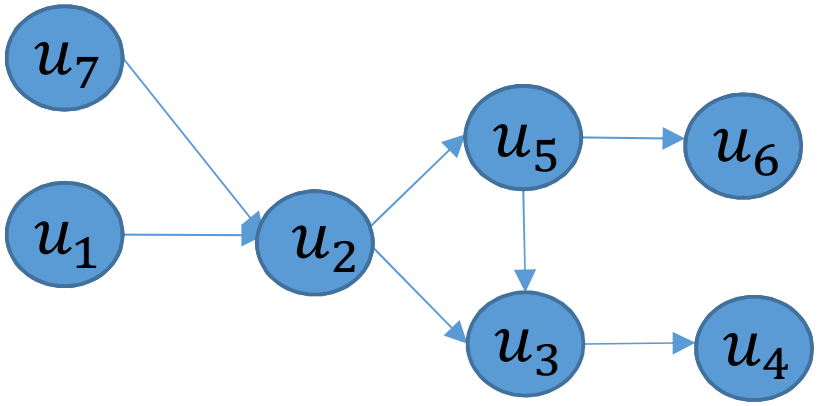


Figure 4.10. Cases for three tunnels with each pair having shared links.

transition event if there exists an $\epsilon \in [-\eta_\rho, \eta_\rho]$ such that $t_0 = t_1 + \kappa_{e_0,e_1} + \epsilon$, where η_ρ is a manually defined tolerance range.

Although the method discussed above is enough for computing $n_b(F)$, it is computationally inefficient if we need to compute from scratch for each F, especially when $|F|$ is large. Thanks to the iterative nature of the proposed category identification algorithm (COIN and R-COIN), we have the following observations:

Lemma 4.7.1. *We only need to compute $\rho_{(E_{t-1}\setminus F)\cup\{e\}}$ and $\rho_{E_{t-1}\setminus F}$ for all candidate category F in COIN.*

Proof. This can be proved by enumerating all the required ρ_F in Line 6 of Alg. 9. □

Lemma 4.7.2. *For any $\rho_{(E_{t-1}\setminus F)\cup\{e\}}$ and $\rho_{E_{t-1}\setminus F}$ required in iteration t, we must have calculated $\rho_{E_{t-1}\setminus F}$ in iteration $t-1$, where $E_t = E_{t-1} \cup \{e\}$.*

Proof. As can be seen in Line 6 of Alg. 9, for any $F_{t-1} \in \text{supp}\,(w(E_{t-1}))$, we must have calculated $\rho_{(E_{t-2}\setminus F_{t-1})\cup\{e'\}}$ and $\rho_{E_{t-2}\setminus F_{t-1}}$ where $E_{t-1} = E_{t-2} \cup \{e'\}$. By noticing that $\rho_{(E_{t-2}\setminus F_{t-1})\cup\{e'\}} = \rho_{E_{t-1}\setminus F}$, we complete the proof. □

The observations above imply that $\rho_{(E_{t-1}\setminus F)\cup\{e\}}$ can be calculated by counting the incremental events based on $n_b(E_{t-1} \setminus F)$. Based on these observations, we devised a mechanism to recursively compute the required $\rho_\cdot$ for COIN and R-COIN. As shown in Alg. 13, the proposed mechanism is based on a multi-level hash table Π, as shown in Fig. 4.11. Concretely, the keys of the first level table are categories $F \subseteq E$, while the value, i.e., $\Pi[F]$, is the the second level hash table. In each second level hash table $\Pi[F]$, the keys are the tunnels $e \in F$, while the value, i.e., $\Pi[F][e]$ is a set of time indices t satisfying $\phi_e^{t-1} < \phi_e^i$ and $q_e^t = 0$.

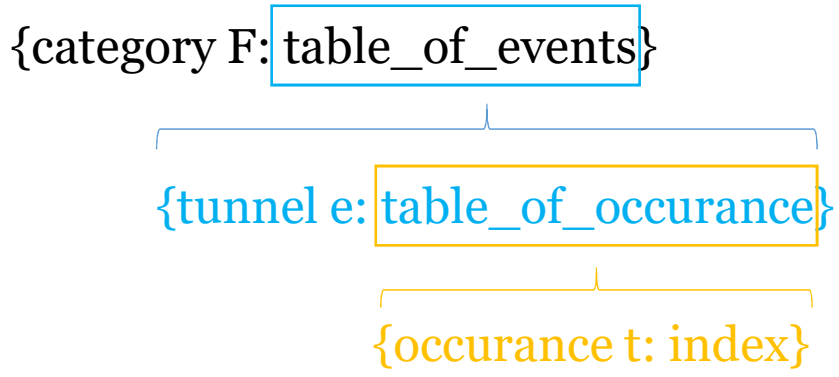


Figure 4.11. Illustration of the hash table for $\hat{\rho}_F$

Algorithm 13: Recursively Update Hash Table Π

input : Π containing the entry $\Pi(F)$, tolerance η_ρ

1 Initialize $\Pi[F \cup \{e\}]$ as $\Pi(F)$ and $\Pi[F \cup \{e\}][e]$ as $\emptyset$, where $\Pi[F \cup \{e\}][e]$ is the hash value of e in $\Pi[F \cup \{e\}]$;

2 **for** $t \in \{t : \phi_e^{t-1} < \phi_e^i, q_e^t = 1\}$ **do**

3 **for** $e' \in E$ **do**

4 **if** $\Pi[F]$ *does not contain key* e' **then**

5 Continue ;

6 **for** $\epsilon \in [-\eta_\rho, \eta_\rho]$ **do**

7 **if** $\Pi[F][e']$ *has* $t - \kappa_{e,e'} + \epsilon$ *as key* **then**

8 Go to Line 2;

9 Add t to $\Pi[F \cup \{e\}][e]$;

10 return $n_b(F \cup \{e\}) = \sum_{e' \in \Pi[F \cup \{e\}]} |\Pi[F \cup \{e\}][e']|$;

The basic idea of Alg. 13 for computing $\rho_{(E_{t-1} \setminus F) \cup \{e\}}$ is to test for each event $t \in \{t : \phi_e^{t-1} < \phi_e^i, q_e^t = 1\}$ whether it is already contained in $\Pi[(E_{t-1} \setminus F)]$ (Line 7). If not, add the event into $\Pi[(E_{t-1} \setminus F)][e]$ (Line 9). Now, we characterize the complexity of Alg. 13.

Lemma 4.7.3. *The complexity of computing* $\Pi[F \cup \{e\}]$ *in Alg. 13 is* $O(T\eta_\rho|V|^2)$.

Proof. In the loop traversing $\Pi[F]$ (Line 3), we have at most $2\eta_\rho$ hashing operations for each $e' \in E$. Thus, for each $t \in \{t : \phi_e^{t-1} < \phi_e^i, q_e^t = 1\}$ in Line 2, we have at most $2\eta_\rho|E|$ hashing operations. Since we have $O(T)$ elements to loop in Line 2, the complexity for computing $\Pi[F \cup \{e\}]$ is $O(T\eta_\rho|E|) = O(T\eta_\rho|V|^2)$ □

Since COIN and R-COIN have $O(|E|)$ iterations, in each of which we need to compute $\rho_\cdot$ for $O(|\underline{E}|)$ times. Thus, the total complexity in computing $\rho_\cdot$ during COIN and R-COIN is $O(|E||\underline{E}|T\eta_\rho|V|^2) = O(T\eta_\rho|\underline{E}||V|^4)$, which is still polynomial in the network size.

Chapter 5
Optimized Cross-Path Attacks via Adversarial Reconnaissance

5.1 Introduction

The trend of network softwarization and virtualization has fundamentally altered the way we build network systems. While the logically centralized control plane provides convenient ways to manage the network resources through various abstractions, such abstractions also hide the complex interactions within the network, which can cause unexpected security threats. In this work, we focus on a particular security threat due to the sharing of links between high-security paths and low-security paths, which enables a new type of *denial-of-service (DoS)* attacks called *cross-path attacks*.

Intuitively, cross-path attacks are indirect DoS attacks, where instead of directly attacking the paths of interest (*target paths*), the attacker sends attack traffic on some other paths (*attack paths*) sharing resources (e.g., link bandwidth) with the target paths, so as to degrade the performance of the target paths by consuming the shared resources. Such attacks are of interest when the target paths are difficult to attack directly, but share network resources with some low-security paths that are more susceptible to attacks.

One scenario for cross-path attacks is in the context of a *Software Defined Network (SDN)* [13], where the target paths are control-plane paths connecting switches to the controller and the attack paths are data-plane paths originating from attacker-controlled hosts that share links with some of the control-plane paths. Instead of directly triggering a flood of control messages to attack the control-plane paths as in earlier attacks [144], a cross-path attack only floods selected paths in the data plane, which makes it both stealthier and more resilient to state-of-the-art control plane defenses such as FloodGuard [145], FloodDefender [146], and SPHINX [147]. Another scenario for cross-path attacks is in

the context of network slicing [148], which is a technology in 5G networks that allows the network provider to set up multiple virtual networks over a shared infrastructure. To improve resource utilization and support elasticity, different slices may share network and computing resources [148]. Meanwhile, slices created for different applications can follow different security standards [149, 150], and some slices may even be managed by less trusted third parties [151]. These practices create opportunities for an attacker to attack paths in a high-security slice (target paths) by consuming resources shared with some paths the attacker can access in a low-security slice (attack paths), while remaining stealthy to intrusion detection systems in the high-security slice.

Despite the demonstration of feasibility in [13], there is little quantitative understanding about cross-path attacks. In this work, we will address this gap by designing an optimized attack strategy that can achieve the maximum impact with a constrained total attack rate. At a high level, our strategy works by (i) inferring the locations and parameters of network elements shared between the target paths and the attack paths, and then (ii) optimally allocating the total attack rate over the attack paths to maximize the performance degradation of the target paths. By analyzing the optimal attack strategy, we not only quantify the maximum damage due to cross-path attacks as a function of the attack rate, but also shed light on possible defenses.

5.1.1 Related Work

As a newly identified attack, cross-path attack has not been extensively studied previously. Therefore, we will provide background by reviewing some relevant security problems and solution techniques.

Security vulnerabilities in SDN: As the architectures and protocols of SDN are designed to facilitate performance and programmability, there are many security vulnerabilities, mainly due to the interdependency between the controller and the switches. In particular, the switch→controller dependency that arises due to the need for the data plane to obtain instructions from the control plane can create a communication bottleneck, which has been exploited in active attacks [144], adversarial reconnaissance [152], and joint reconnaissance and attack [13, 153]. The cross-path attack in this context [13] is a reconnaissance-based attack that exploits the switch→controller dependency to infer which attacker-controlled data-plane paths share links with at least one control-plane path, in order to identify the data-plane paths that can be used to launch a cross-path attack on the control plane. The reconnaissance strategy in [13] only infers whether there exists at least one shared link between a data-plane path and the targeted control-

plane paths, and thus can only support a non-optimized cross-path attack. In contrast, we will provide a way to design an optimized cross-path attack through fine-grained reconnaissance based on network tomography.

Security vulnerabilities in network slicing: Network slicing introduces both *content-level threats* such as unauthorized access, compromise of functions/devices, and side-channels across slices [154, 155], and *performance-level threats* due to malicious abuse of resource quotas [154, 156]. The cross-path attack in this context belongs to performance-level threats. While cross-path attack under network slicing can be defended by detecting the attack through the cooperation of its control plane [157] or completely isolating different slices [158], both approaches have severe limitations: the former will fail if the attack slice's control plane is compromised, and the latter will cause poor resource utilization. In this regard, our work helps to strike a balance between resource utilization and security in network slicing by quantifying the maximum impact of cross-path attacks.

Network interdiction: Traditional network interdiction refers to a problem where an interdictor tries to reduce the throughput of network users by removing selected links under a budget constraint [159]. A variation of this problem, recently proposed in [160], is conceptually similar to cross-path attack in that instead of removing links, the interdictor tries to reduce the available capacities of links traversed by target paths by injecting flows on selected paths. Despite the conceptual similarity, [160] addressed a fundamentally different problem of optimizing the (possibly multi-path) routing of injected traffic in a clairvoyant setting where the network topology and the links traversed by each target path are known to the interdictor, and the routing of injected traffic is controllable, which generally requires the cooperation of the network. In contrast, cross-path attack is based on a much weaker threat model where the attacker is an outsider of the network without internal support or information. Therefore, how to learn the internal information required for effective attack design is a critical question in cross-path attack, which is the focus of this work.

Network tomography: The optimal design of cross-path attacks requires the attacker to infer the network elements shared between two sets of paths from end-to-end measurements, which is similar to the problem addressed by network topology tomography/inference [112]. With few exceptions (e.g., [161, 162]), topology inference algorithms generally require active probing on all the paths; see [123] and references therein. This can be used to generate carefully crafted probes, such as "packet strings" [163] or "packet sandwiches" [164], which produce correlated measurements that can reveal the existence and parameters of the links shared by different paths. Our problem is different in that the attacker can only probe a subset of paths (i.e., the attack paths) but wants to infer

the elements they share with the other paths (i.e., the target paths).

5.1.2 Summary of Contributions

Our goal is to understand the strategy and impact of the optimal cross-path attack under a constrained total rate, with the following contributions:

1) We develop novel inference algorithms that can consistently estimate the locations and parameters of the links shared between attack paths and target paths via active probing on the attack paths and passive monitoring on the target paths.

2) Under the assumption that each shared link can be modeled as an M/M/1, M/D/1, or G/G/1 queue, we derive the optimal attack design that can maximally degrade the performance of the target paths under a bounded total attack rate.

3) We evaluate the proposed algorithms by high-fidelity packet-level simulations under various settings, which show that (i) our inference algorithms can estimate the (topological) locations of shared links with good accuracy but not their detailed parameters, but (ii) our attack strategy designed based on these estimates can still cause substantially more performance degradation than some intuitive ways of launching cross-path attacks, which signals the importance of considering such intelligent attack strategies in the design of defenses.

Roadmap. We formulate our problem in Section 5.2, present the algorithms to infer the locations and parameters of shared links in Section 5.3, and present the corresponding attack design in Section 5.4. We then evaluate our solutions in Section 5.5 and conclude the paper in Section 5.6. **All the proofs are provided in Appendix 5.7.1.**

5.2 Problem Formulation

5.2.1 Network and Threat Model

Consider two sets of paths in a network, referred to as the *attack paths* $P_A := \{s_{Ai} \to t_{Ai}\}_{i=1}^{N_A}$ and the *target paths* $P_B := \{s_{Bi} \to t_{Bi}\}_{i=1}^{N_B}$, where $s \to t$ denotes the routing path from source s to destination t and N_A/N_B the number of paths in P_A/P_B. Suppose that an attacker is interested in attacking the target paths in P_B, but can only passively monitor the end-to-end performance (e.g., delays) on these paths. Meanwhile, the attacker can actively send packets on the attack paths in P_A. We will focus on the important special case of $s_{Ai} \equiv s_A$ $(i = 1, \ldots, N_A)$, as it represents a most easily-deployable attack with only one active malicious node (i.e., s_A). Let $T_A := \{t_{Ai} | i = 1, \cdots, N_A\}$ denote

the set of destinations of the attack paths. We note that the multi-source case, where the attacker controls multiple active malicious nodes, is not a trivial extension of the single-source case. We leave the study of the multi-source case to future work.

The two sets of paths may share some network elements. For clarity, we will model all the shared elements as "shared links", which can represent any shared resources (e.g., communication links, network functions, and other services). Here, "shared" means shared by attack traffic and target traffic without isolation. While traffic isolation technologies exist, applying them will lower resource utilization and hence the revenue of the network provider [148, 165]. In this regard, our work aims at quantifying the risk due to lack of isolation to inform a proper tradeoff. We model such link sharing by a (logical) routing topology $G = (V, E)$, which is a graph formed by all the paths in $P_A \cup P_B$. According to [163, 166], V is a set of vertices representing sources, destinations, and branching/joining points between paths, and E is a set of edges representing the connections between the vertices, where a sequence of consecutive links without branching/joining points is represented by a single edge. We assume that the attacker does not have access to the control plane, i.e., he does not know the ground truth of G. Instead, the attacker can infer information about G from end-to-end measurements on P_A and P_B. We will use "link" to refer to a communication link in the underlying network and "edge" to refer to a point-to-point connection in the routing topology. Similarly, we will use "node" to refer to a physical node in the underlying network and "vertex" to refer to a logical node in the routing topology. As commonly assumed in the literature [163, 166], we assume that during the inference and attack, there is a fixed and unique routing path from each node to every other node.

Remark: While there may be other paths carrying co-existing flows in the network, it suffices to focus on the target and the attack paths for the purpose of modeling the cross-path attack. The impact of co-existing flows will be captured as background traffic on the links traversed by $P_A \cup P_B$. Our threat model depicts a *pure* cross-path attack where the attacker can only actively send packets on the attack paths and thus can only attack the target paths through "cross-path" influence. In some scenarios such as the cross-path attack between the data plane and the control plane in SDN [13], it is possible for the attacker to generate packets on the target paths (e.g., by triggering "packet-in" messages). However, to evade existing defenses against direct attacks, such attacker-triggered traffic on the target paths must resemble the normal traffic on these paths, which is usually insufficient to cause notable performance degradation. Intuitively, the ability to passively monitor the performance of the target paths is the minimum requirement for designing a nontrivial cross-path attack. We thus adopt this threat

model to maximize the applicability of our result.

We assume the following capabilities of the attacker. **First,** the attacker can observe packets on each target path $s_{Bi} \to t_{Bi}$ as soon as they are transmitted, even if s_{Bi} differs from s_A. For example, the attacker may intersect target traffic at locations $(s_{Bi})_{i=1}^{N_B}$ (i.e., s_{Bi} denotes the starting point of intersection for a path of interest). Instead of directly attacking target traffic at these points of intersection, the attacker only uses them to passively monitor the target traffic to launch a stealthier attack. **Second**, the attacker can measure the end-to-end one-way delays of packets on both the attack paths P_A and the target paths P_B. **Third**, the attacker will not be exposed by sending traffic on the attack paths. For example, the attacker may control many edge devices, which connect to the same ingress point s_A or egress point t_{Ai} (i.e., s_A and t_{Ai} may each represent a set of devices), and thus the attack flows can evade detection by the network operator even if the aggregate flow rate is high.

5.2.2 Problem Statement

While the sharing of links makes the paths in P_B vulnerable to cross-path attacks launched from the paths in P_A, the impact of such attacks greatly depends on the attack strategy. To understand the maximum impact of cross-path attacks, we develop an intelligent attack strategy by combining fine-grained adversarial reconnaissance with optimized attack design, by solving the following problems:

1) Adversarial Reconnaissance. We investigate to what extent the attacker can learn about the shared links based on active probing on P_A and passive monitoring on P_B.

2) Optimized Attack Design. Based on the inferred information, we investigate the optimal allocation of attack traffic over the attack paths to maximize the performance degradation (e.g., increase in average delay) inflicted on the target paths.

Remark: Our threat model requires the attacker to monitor end-to-end performance on the target paths. While this is arguably the minimum information needed for any nontrivial attack design, it does impose limitations on which paths can be set as the target. For example, in the context of SDN [13], only the control-plane paths for switches traversed by attacker-controlled data-plane paths can be the target paths. In the context of network slicing, the target paths can be the backhaul paths to cells containing attacker-controlled user equipments (UEs), which are likely to share the same paths with other UEs in the same cell and can thus be used as their proxies in collecting measurements. Instead of directly launching attacks from these attacker-controlled UEs, a cross-path attack only uses them to passively collect measurements so as to launch an effective

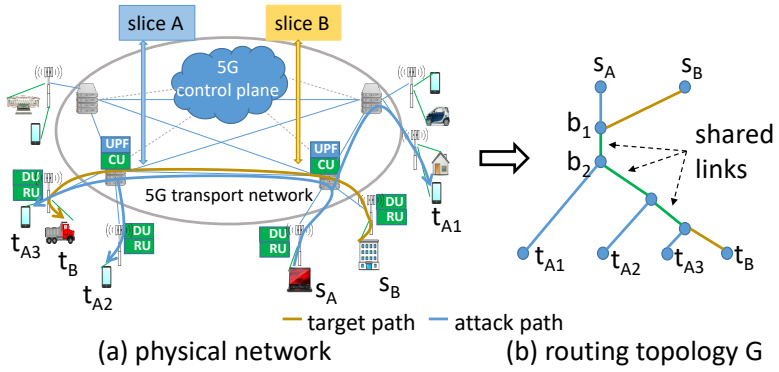


Figure 5.1. Cross-path attack in the context of network slicing.

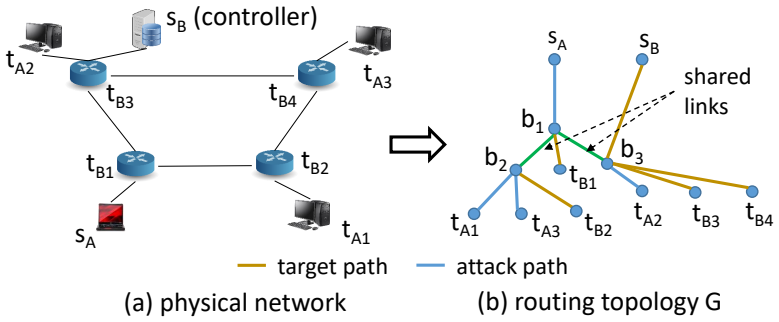


Figure 5.2. Cross-path attack in the context of SDN.

attack from elsewhere in the network, and is thus stealthier.

5.2.3 Illustrative Example

Example in network slicing: Consider the scenario in Fig. 5.1 (a), where the attacker controls a malicious node s_A that can send traffic on a set of paths $s_A \to t_{Ai}$ $(i = 1, 2, 3)$ in slice A but wants to attack another path $s_B \to t_B$ in slice B. The attack paths share the following network elements with the target path: $s_A \to t_{A1}$ only shares the source-side central unit (CU) and user plane function (UPF); $s_A \to t_{A2}$ also shares backhaul links between the cell sites and the destination-side CU and UPF; $s_A \to t_{A3}$ further shares midhaul links and the destination-side radio unit (RU) and distributed unit (DU). These relationships can be modeled by the routing topology in Fig. 5.1 (b).

Example in SDN: Consider the scenario in Fig. 5.2 (a), where the attacker wants to attack the control paths between the controller s_B and the switches t_{Bi} $(i = 1, \ldots, 4)$ by sending traffic on the data paths $s_A \to t_{Ai}$ $(i = 1, 2, 3)$. Assume shortest path routing for both data and control paths (assuming that t_{B2} connects to the controller via t_{B1} and s_A connects to t_{A3} via t_{B2}). Each data path shares some links with the control

paths: $s_A \to t_{A1}$ shares a link with $s_B \to t_{B2}$, $s_A \to t_{A2}$ shares a link with $s_B \to t_{B1}$ and $s_B \to t_{B2}$, and $s_A \to t_{A3}$ shares a link with $s_B \to t_{B2}$. Meanwhile, due to the separate processing of data and control packets within a switch, the shared nodes (i.e., switches) will not cause performance correlation between data and control paths and can thus be ignored. These relationships can be modeled by the routing topology in Fig. 5.2 (b), where we have inserted zero-delay edges (b_1, t_{B1}), (b_2, t_{B2}), and (b_3, t_{B3}) to make each source/destination have degree one.

While it is relatively easy to identify which attack paths share at least one link with the target paths (e.g., by measuring the target path delays with/without traffic on each attack path as in [13]), different attack paths can influence the target paths to different extents. Given the routing topology G, one can intuitively identify the most useful attack path, e.g., $s_A \to t_{A3}$ in Fig. 5.1, but the attacker cannot directly observe such internal information. Therefore, when the attacker has access to multiple attack paths but only resources to generate a limited amount of traffic, it is unclear how he can attack most effectively. Below, we will show that the attacker can actually infer sufficient information about the routing topology to design the optimal attack that causes the maximum performance degradation to the target paths, by only passively monitoring the target paths.

5.3 Adversarial Reconnaissance

We will show that under mild assumptions, the attacker can consistently infer both the locations and the parameters of the links shared between the attack paths and the target paths.

5.3.1 Preliminaries

The problem of inferring the relationship between paths from end-to-end measurements belongs to a branch of network tomography focusing on topology inference, for which many algorithms have been proposed (see Section 5.1.1). However, these algorithms typically require active probing on all the paths and hence are not applicable to our problem. Nevertheless, there are some results we can leverage, as summarized below.

The foundation of topology inference is using end-to-end measurements to infer the "lengths" of links defined by certain additive performance metrics. As a concrete example, we will adopt a canonical metric that can be inferred from delay measurements, but our reconnaissance algorithm can work with any additive metric for which the so-called

"category weight" (see Definition 5.3.1) can be inferred from end-to-end measurements.

The metric we adopt is called *utilization-based metric* [163, 167], which is a classical additive metric used in topology inference. Let γ_e denote the probability that a packet traversing link e does not experience any queueing delay. The utilization-based metric for link e is defined as $u_e := -log\gamma_e$, which is additive across independent links. By comparing the end-to-end delay of a packet with the minimum delay on the measured path, we can infer whether the packet incurs any queueing delay and hence estimate the utilization-based metric for the path. It was shown in [124] that with simultaneous measurements from multiple paths (obtained via multicast or back-to-back unicast), we can uniquely identify the utilization-based metric[1] at a certain granularity as follows.

Definition 5.3.1. *Given a set C of paths, we define the following:*

1. *the* cast weight ϕ_C *is the sum of metrics for all the links traversed by any of the paths in C;*

2. *a* category $\Gamma_{\frac{C'}{C}}$ *for $C' \subseteq C$ and $C' \neq \emptyset$ is the set of links traversed by every path in C' but none of the paths in $C \setminus C'$;*

3. *the* category weight *for a category $\Gamma_{\frac{C'}{C}}$, denoted by $w_{\frac{C'}{C}}$, is the sum of the metrics for all the links in $\Gamma_{\frac{C'}{C}}$.*

For example, consider the set of paths $C := \{s_A \to t_{A1}, s_A \to t_{A2}, s_A \to t_{A3}, s_B \to t_B\}$ in Fig. 5.1 (b). The cast weight $\phi_{s_A \to t_{A1},\ s_A \to t_{A2}, s_A \to t_{A3}}$ is the sum metric for all the blue and green links. Category $\Gamma_{\frac{s_A \to t_{A1}, s_A \to t_{A2}, s_A \to t_{A3}}{s_A \to t_{A1}, s_A \to t_{A2}, s_A \to t_{A3}, s_B \to t_B}}$ only contains link (s_A, b_1), and category $\Gamma_{\frac{s_A \to t_{A1}, s_A \to t_{A2}, s_A \to t_{A3}, s_B \to t_B}{s_A \to t_{A1}, s_A \to t_{A2}, s_A \to t_{A3}, s_B \to t_B}}$ only contains link (b_1, b_2).

Let $\mathcal{C} := 2^C \setminus \{\emptyset\}$ denote all the nonempty subsets of C and $U_{C'}$ ($C' \in \mathcal{C}$) denote a Bernoulli variable that equals 1 if and only if a multicast probe on C' does not incur any queueing delay. Under the assumption that different links have independent queue states as in [124, 163, 167] (which holds approximately under heavy independent cross-traffic), we have

$$-\log(\Pr\{U_{C'}=1\}) = -\log(\prod_{e \in \bigcup_{p \in C'} p} \gamma_e) = \sum_{e \in \bigcup_{p \in C'} p} u_e =: \phi_{C'}. \tag{5.1}$$

[1]The empirical evaluations in [124] were based on *loss-based metric* defined as $u'_e := -\log \gamma'_e$, where γ'_e denotes the no-loss probability at link e, but the theoretical result in Theorem III.1 in [124] held for any additive metric for which the cast weights can be estimated from end-to-end measurements.

This, together with Definition 5.3.1, implies the following relationship between the cast weights and the category weights:

$$\sum_{C_1 \in \mathcal{C}: C_1 \cap C_2 \neq \emptyset} w_{\frac{C_1}{C}} = \phi_{C_2}, \quad \forall C_2 \in \mathcal{C}. \tag{5.2}$$

Using (approximated) multicast on C, we can infer all the cast weights in $(\phi_{C'})_{C' \in \mathcal{C}}$, which can then be used to uniquely identify the category weights as shown below.

Theorem 5.3.1 (Theorem III.1 in [124]). *Given the cast weights $(\phi_{C'})_{C' \in \mathcal{C}}$, all the category weights $(w_{\frac{C'}{C}})_{C' \in \mathcal{C}}$ are uniquely determined by* (5.2).

Below, we will show how to use this existing result to detect the links shared between the target paths and the attack paths.

5.3.2 Shared Weight Inference

Category weights provide valuable information about the relationship between paths. Specifically, if $w_{\frac{C'}{C}} > 0$, then we know that $\Gamma_{\frac{C'}{C}} \neq \emptyset$, i.e., there is at least one link shared by the paths in C' but not those in $C \setminus C'$. Moreover, under the assumption that every link has a non-zero metric (i.e., non-zero queueing probability), $w_{\frac{C'}{C}} = 0$ implies that $\Gamma_{\frac{C'}{C}} = \emptyset$. However, applying this idea directly to the paths in $P_A \cup P_B$ will require active probing on all the paths. Nevertheless, with active probing only on P_A, we can still infer the relationship between each target path $p \in P_B$ and all the attack paths in P_A, which turns out to be sufficient for the design of the optimal cross-path attack as explained in Section 5.4. The idea is to mimic a multicast on $P_A \cup \{p\}$ by monitoring path p and sending a multicast probe (or back-to-back unicast probes) on P_A when a packet is transmitted on p. Because these packets are sent very close to each other, they will observe similar queue states at shared links [163], thus mimicking a multicast on $P_A \cup \{p\}$.

Using such mimicked multicast, one may try to apply existing topology inference algorithms. However, most of such algorithms are heuristics without guaranteed accuracy, and the existing algorithms with performance guarantee all address scenarios different from ours. For example, [163] requires all the paths to share a single source, [166, 168] require all the sources to share the same set of destinations, and [169] requires the ability to probe the path between any pair of boundary vertices (in our case, these are the endpoints of all the paths in $P_A \cup \{p\}$). These differences make the existing algorithms inapplicable to our problem. Below, we will show an algorithm that can infer the shared links between the attack paths in P_A and a given target path p with guaranteed accuracy,

which is then repeated for each $p \in P_B$. To our knowledge, this is the *first* algorithm that can infer the routing topology formed by a set of single-source paths and another path with arbitrary source and destination by only measuring these paths.

5.3.2.1 Algorithm

Under the assumptions in Section 5.2.1, the paths in P_A form a (logical) routing tree $\mathcal{T}$ with the source s_A as root and the destinations in T_A as leaves. Since the attacker can send active probes on P_A, he can infer $\mathcal{T}$ using existing topology inference algorithms such as Rooted Neighbor Joining (RNJ) [163]. Without loss of generality, we assume that $\mathcal{T}$ is a binary tree, as non-binary trees can be represented as binary trees by inserting zero-weight edges. Our focus is thus on inferring the relationship between $\mathcal{T}$ and a given target path $p := s_B \to t_B$. We model this relationship by a vector $\boldsymbol{W} := (W_e)_{e \in \mathcal{T}}$, where W_e denotes the sum metric of the links shared between edge $e \in \mathcal{T}$ and the target path ($W_e := 0$ if they do not share any link). We will refer to W_e as the *shared weight* on e for simplicity.

We define a few notations for the ease of presentation. Given a binary tree $\mathcal{T}$, $s_{\mathcal{T}}$ denotes the root, $b_{\mathcal{T}}$ denotes the first branching point from the root, $\delta_{l\mathcal{T}}$ is the set of leaves located in the left subtree of $\mathcal{T}$, and $\delta_{r\mathcal{T}}$ is the set of leaves in the right subtree of $\mathcal{T}$. If $\mathcal{T}$ only has one leaf t, then $\delta_{l\mathcal{T}} = \delta_{r\mathcal{T}} = \{t\}$. We denote the shared weight on a subpath $v_1 \to v_2$ in $\mathcal{T}$ by $W_{v_1 \to v_2} := \sum_{e \in v_1 \to v_2} W_e$.

The overall algorithm is given in Alg. 14, which prepares the routing tree $\mathcal{T}$ formed by the attack paths and then invokes Alg. 15. Alg. 15 is a recursive algorithm. Given a binary tree $\mathcal{T}'$ (initially $\mathcal{T}' = \mathcal{T}$), each recursion estimates the shared weight on the stem of $\mathcal{T}'$, i.e., edge $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$. To this end, the attacker mimics tri-cast by sending two back-to-back probes from s_A to two destinations τ_1, τ_2 from different subtrees of $\mathcal{T}'$ whenever observing a packet on the target path $s_B \to t_B$ (lines 1–2). The measured delays are used to estimate a subset of the category weights, stored in variables ρ_l, ρ_r, and ρ_s as in line 3. If we measure the category weights by the utilization-based metric, then the category weights can be inferred by first using the measured delays to estimate the no-queueing probability on each subset of $C := \{s_A \to \tau_1, s_A \to \tau_2, s_B \to t_B\}$ and compute the cast weights $(\phi_{C'})_{C' \subseteq C}$, which are then plugged into (5.2) to solve for the category weights. We can adopt other metrics by modifying the implementation of line 3, as long as the corresponding category weights can be inferred from end-to-end measurements. The shared weight on edge $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$ is estimated as ρ_s minus the shared weight on $s_A \to s_{\mathcal{T}'}$, which has been estimated in previous recursions (line 4). The recursion is then repeated

for each subtree of $\mathcal{T}'$. The recursion stops when either (i) $\mathcal{T}'$ has no subtree (line 5), or there is no overlap between the target path and either subtree (line 8).

Algorithm 14: Shared_Weight_Inference

input : s_A, $T_A := \{t_{Ai}\}_{i=1,\cdots,N_A}$, s_B, t_B
output : shared weight vector $\boldsymbol{W}$

1 $\mathcal{T} \leftarrow \bigcup_{i=1,\cdots,N_A}(s_A \rightarrow t_{Ai})$; `// inferred by RNJ`
2 $\boldsymbol{W} \leftarrow \mathbf{0}$;
3 $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}, s_A, T_A, s_B, t_B, \boldsymbol{W})$;

Algorithm 15: Recursive_Inference

input : $\mathcal{T}'$, s_A, T_A, s_B, t_B, previously inferred $\boldsymbol{W}$
output : updated shared weight vector $\boldsymbol{W}$

1 randomly pick τ_1 from $\delta_{l\mathcal{T}'}$ and τ_2 from $\delta_{r\mathcal{T}'}$;
2 send probes on $s_A \rightarrow \tau_1$, $s_A \rightarrow \tau_2$ concurrently[2] with packets monitored on $s_B \rightarrow t_B$;
3 use the measured delays to infer the category weights: $\rho_l \leftarrow w_{\frac{s_A\rightarrow\tau_1, s_B\rightarrow t_B}{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}}$, $\rho_r \leftarrow w_{\frac{s_A\rightarrow\tau_2, s_B\rightarrow t_B}{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}}$, $\rho_s \leftarrow w_{\frac{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}{s_A\rightarrow\tau_1, s_A\rightarrow\tau_2, s_B\rightarrow t_B}}$;
4 $W_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})} \leftarrow \rho_s - W_{s_A\rightarrow s_{\mathcal{T}'}}$;
5 **if** $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ **then**
6 | return;
7 **if** $\rho_s \neq 0$ **then**
8 | **if** $\rho_l = \rho_r = 0$ **then**
9 | | return;
10 | **if** $\rho_l > 0$ **then**
11 | | $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{l\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
12 | | $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;
13 | **if** $\rho_r > 0$ **then**
14 | | $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{r\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
15 | | $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;
16 **else**
17 | $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{l\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
18 | $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;
19 | $\mathcal{T}' \leftarrow \bigcup_{t_{Ai}\in\delta_{r\mathcal{T}'}}(b_{\mathcal{T}'} \rightarrow t_{Ai})$;
20 | $\boldsymbol{W} \leftarrow$ Recursive_Inference$(\mathcal{T}', s_A, T_A, s_B, t_B, \boldsymbol{W})$;

[2]For simplicity, here we assume that the attacker can observe packets on the target path as soon as they are transmitted, and s_A, s_B have similar distances to the shared links. This assumption will be relaxed by aligning the measurements via correlation maximization as discussed in Appendix 5.7.2.1.

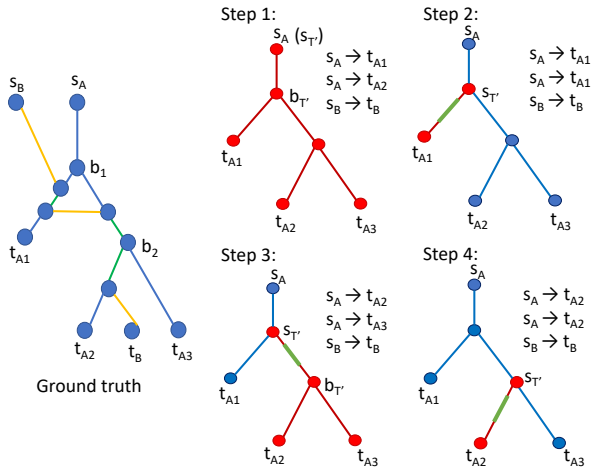


Figure 5.3. Illustration for Alg. 15 (shared links are marked in green).

5.3.2.2 Illustrative Example

Fig. 5.3 illustrates the steps of Alg. 15. On the left is the ground truth topology containing the attack paths from s_A to destinations t_{A1}, t_{A2}, t_{A3} and a target path $s_B \to t_B$, where the shared links are marked in green. Alg. 15 infers the locations and weights of these shared links in 4 steps. In each step, we mark the tree $\mathcal{T}'$ in red and label nodes $s_{\mathcal{T}'}$ and $b_{\mathcal{T}'}$ (if any). In step 1, we mimic tri-cast probes on $s_A \to t_{A1}$, $s_A \to t_{A2}$ (or t_{A3}), and $s_B \to t_B$. The results should show that $\rho_s = 0$, indicating that $s_B \to t_B$ has no overlap with $s_A \to b_{\mathcal{T}'}$. Then we search both subtrees. In the left subtree (step 2), we mimic bi-cast on $s_A \to t_{A1}$ and $s_B \to t_B$ to find out the shared weight between $s_B \to t_B$ and $s_{\mathcal{T}'} \to t_{A1}$. In the right subtree (steps 3–4), we first mimic tri-cast on $s_A \to t_{A2}$, $s_A \to t_{A3}$, and $s_B \to t_B$ to find out the shared weight on $s_{\mathcal{T}'} \to b_{\mathcal{T}'}$ (step 3), and then since $\rho_l > 0$, we will search the left subtree (step 4) to obtain all the shared weights.

5.3.2.3 Correctness

Alg. 14 gives consistent estimates of the shared weights in the following sense.

Theorem 5.3.2. *If all the shared links have non-zero metrics and the category weights are accurately inferred in line 3 of Alg. 15, then Alg. 14 will accurately infer the shared weight vector* $\boldsymbol{W}$.

As the number of probes increases, the estimated path-level statistics (i.e., no-queueing probabilities) will converge to their true values, so will the estimated category weights by Theorem 5.3.1. Thus, Alg. 14 provides consistent estimates of the shared weights.

5.3.2.4 Complexity

Each recursion of Alg. 15 takes $O(1)$ time (excluding probing time) as it only estimates a constant number of cast/category weights. For the number of recursions, the worst case is when all the non-zero shared weights are associated with the last edges to the destinations in $\mathcal{T}$, in which case Alg. 15 needs to perform a recursion for each edge. As a tree with N_A leaves (N_A: #attack paths) and no degree-2 vertices (implied by RNJ [163]) has at most $2N_A - 2$ edges, the complexity of Alg. 15 is $O(N_A)$. The overall complexity of Alg. 14 is $O(N_A^2 \log N_A)$, dominated by the complexity of RNJ [163].

5.3.3 Parameter Inference

For simplicity, we will refer to the shared portion between each edge $e \in \mathcal{T}$ and the target path as a *shared link* (although it can correspond to a sequence of links in the underlying network). Although the shared weight vector $\boldsymbol{W}$ provides both the locations and the metrics of the shared links, this information is not sufficient for optimal attack design. Specifically, by Alg. 15, each W_e is inferred under a probing rate that is only twice of the traffic rate on the target path, which is generally not enough to cause congestion. To design an effective attack, the attacker needs to predict the impact of higher attack rates on the shared links. Our idea for addressing this challenge is to model each shared link (detected by $W_e > 0$) as a queue with unknown parameters, and conduct further probing experiments with varying rates to infer these parameters.

Algorithm 16: Parameter_Inference

input : $\mathcal{T}$, $\boldsymbol{W}$, s_A, T_A, s_B, t_B
output : parameters $\boldsymbol{\xi} := (\xi_e)_{e \in \mathcal{T}}$ of shared links

1 $\boldsymbol{\xi} \leftarrow \boldsymbol{0}$;
2 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}, \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;

5.3.3.1 Algorithm

Let ξ_e denote the unknown parameter (or parameter vector) of the shared link on edge $e \in \mathcal{T}$. We infer $\boldsymbol{\xi} := (\xi_e)_{e \in \mathcal{T}}$ through a recursive procedure similar to Alg. 14–15, as shown in Alg. 16–17.

Specifically, given a binary tree $\mathcal{T}'$ (initially $\mathcal{T}' = \mathcal{T}$), each recursion of Alg. 17 estimates the parameter of the shared link on the stem of $\mathcal{T}'$, if any. If the shared link exists (i.e., $W_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})} > 0$), then either the left or the right subtree does not share any

Algorithm 17: Parameter_Update

input : $\mathcal{T}'$, $\boldsymbol{W}$, s_A, T_A,s_B,t_B, previous $\boldsymbol{\xi}$
output : updated $\boldsymbol{\xi}$

1 **if** $W_{(s_{\mathcal{T}'},b_{\mathcal{T}'})} > 0$ **then**
2 **if** $\delta_{l\mathcal{T}'} \neq \delta_{r\mathcal{T}'}$ **then**
3 randomly choose a destination τ^* from the subtree of $\mathcal{T}'$ not sharing any link with $s_B \to t_B$;
4 **else**
5 set τ^* to the only destination in $\mathcal{T}'$;
6 vary the probing rate on path $s_A \to \tau^*$ among $\bar{\lambda}_k$ $(k = 1, \ldots, K)$ and measure the corresponding average delay ψ_k of path $s_B \to t_B$;
7 $\xi_{(s_{\mathcal{T}'},b_{\mathcal{T}'})} \leftarrow \arg\min_{\xi_{(s_{\mathcal{T}'},b_{\mathcal{T}'})}} \sum_{k=1}^{K} \left(\psi_k - D_{\tau^*}(\boldsymbol{\xi}; \bar{\lambda}_k)\right)^2$;
8 **if** $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ **then**
9 return
10 **if** $\tau^* \in \delta_{l\mathcal{T}'}$ **then**
11 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{r\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
12 **if** $\tau^* \in \delta_{r\mathcal{T}'}$ **then**
13 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{l\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
14 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}', \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;
15 **else**
16 **if** $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ **then**
17 return
18 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{l\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
19 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}', \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;
20 $\mathcal{T}' \leftarrow \bigcup_{\tau \in \delta_{r\mathcal{T}'}} (b_{\mathcal{T}'} \to \tau)$;
21 $\boldsymbol{\xi} \leftarrow Parameter\_Update(\mathcal{T}', \boldsymbol{W}, s_A, T_A, s_B, t_B, \boldsymbol{\xi})$;

link with the target path as explained in the proof of Theorem 5.3.2 (e.g., if the stem $(b_{\mathcal{T}'}, b_{l\mathcal{T}'})$ of the left subtree has $W_{(b_{\mathcal{T}'},b_{l\mathcal{T}'})} = 0$, then the left subtree contains no shared link). Therefore, we can pick a destination τ^* from the subtree without any shared link (line 3). We then conduct a number of probing experiments on $s_A \to \tau^*$ with varying rates, while measuring the average delay of the target path (line 6). Under probing rate $\bar{\lambda}_k$, the true average delay of $s_B \to t_B$ is given by

$$D_{\tau^*}(\boldsymbol{\xi}; \bar{\lambda}_k) := c_{\tau^*} + \sum_{e \in s_A \to \tau^* : W_e > 0} d(\xi_e; \bar{\lambda}_k), \tag{5.3}$$

where c_{τ^*} denotes the average queueing and transmission delay on the links of $s_B \to t_B$ that are not shared with $s_A \to \tau^*$ plus the propagation delay on $s_B \to t_B$, and $d(\xi_e; \bar{\lambda}_k)$ denotes the average queueing and transmission delay of the shared link on edge e, which is

a function of the link parameter ξ_e and the probing rate $\bar{\lambda}_k$. Using (5.3) and the measured average delays, we can estimate the parameter for $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$ through least square fitting (line 7). The process is then repeated for other edges of $\mathcal{T}'$ through recursions. Note that the selection of τ^* and the top-down approach ensure that the parameters of other shared links on $s_A \to \tau^*$ would have been estimated, leaving $\xi_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})}$ (and possibly c_{τ^*}) as the only unknown parameter to estimate in line 7.

5.3.3.2 Queueing Models

As concrete examples, we consider the following queueing models ($\bar{\lambda}$ denotes probing rate):

• *M/M/1:* If each shared link is modeled as an M/M/1 queue with residual capacity $r_e - \bar{\lambda}$, its average delay equals [170]

$$d(r_e; \bar{\lambda}) = \frac{1}{r_e - \bar{\lambda}}, \tag{5.4}$$

where r_e is the residual capacity before probing, which is the unknown parameter to infer.

• *M/D/1:* If each shared link is modeled as an M/D/1 queue, the average delay depends on two unknown parameters [170]

$$d(\lambda_e, \mu_e; \bar{\lambda}) = \frac{2\mu_e - \lambda_e - \bar{\lambda}}{2\mu_e(\mu_e - \lambda_e - \bar{\lambda})}, \tag{5.5}$$

where λ_e is the background arrival rate (excluding probing traffic) and μ_e is the service rate.

• *G/G/1:* In general, we can model the shared link as a G/G/1 queue. By Kingman's formula [170], the average delay (including service time) can be approximated by

$$d(\lambda_e, \mu_e, \sigma_{ae}, \sigma_{se}; \bar{\lambda}) \approx \frac{1}{2\mu_e} \frac{\lambda_e + \bar{\lambda}}{\mu_e - \lambda_e - \bar{\lambda}} \Big(\sigma_{ae}^2 (\lambda_e + \bar{\lambda})^2 + \sigma_{se}^2 \mu_e^2 \Big) + \frac{1}{\mu_e}, \tag{5.6}$$

which requires four unknown parameters: the background arrival rate λ_e, the service rate μ_e, the variance of the interarrival time σ_{ae}^2, and the variance of the service time σ_{se}^2. Note that treating σ_{ae}^2 as a constant is an approximation as it generally depends on the probing traffic.

Discussion: It is known that the delay in traversing an IP network can be modeled as a deterministic propagation delay (incorporated into c_{τ^*}) plus random delays to traverse

a series of single-server FIFO queues [171]. The main restrictive assumptions here are that the background traffic is Poisson (for M/M/1 and M/D/1), and packet sizes are exponentially distributed (for M/M/1) or constant (for M/D/1). While these assumptions are not satisfied exactly in practice, studies have shown that when multiplexing a large number of independent flows as in the case of heavy background traffic, the packet arrivals tend to a Poisson process, and the queue length distribution tends to that of a M/D/1 queue [172]. In our evaluations (see Section 5.5.2), we will stress-test our algorithms derived from these queueing models in a realistic setting which does not follow these models exactly.

5.3.3.3 Correctness

Alg. 16 provides consistent estimates of the parameters of the shared links in the following sense.

Theorem 5.3.3. *Given an accurate estimate of the shared weight vector $\boldsymbol{W}$, if all the shared links have non-zero metrics, and the estimated average delay ψ_k in line 6 of Alg. 17 is accurate and consistent with the model in* (5.3)*, then Alg. 16 will accurately estimate the parameters of all the shared links as long as (i) $K > 2$ under the M/M/1 or M/D/1 model, and (ii) $K > 4$ under the G/G/1 model.*

5.3.3.4 Complexity

The number of recursions of Alg. 17 is $O(N_A)$ (N_A: #attack paths) as $\mathcal{T}$ has $O(N_A)$ edges, i.e., the parameter estimation (lines 2–7) is repeated for $O(N_A)$ times. For $K = O(1)$, solving the least square fitting problem (line 7) takes $O(1)$ time as it fits an $O(1)$-variable function at $O(1)$ points. Thus, excluding the measurement time (which is independent of N_A), the complexity of Alg. 16 is $O(N_A)$.

5.4 Optimized Attack Design

Given the locations and parameters of the shared links, the attacker can use this information to design optimized attacks. To quantify the potential impact of such attacks, we investigate attack strategies that can cause the maximum performance degradation on the target paths at a bounded cost.

5.4.1 Attacker's Optimization

As a concrete example, we consider the attacker's objective as maximally increasing the total average delay of the target paths. Our approach is extensible to other objectives, as demonstrated in Section 5.4.3.

Specifically, let W_{ie} denote the shared weight between target path $s_{Bi} \to t_{Bi}$ and edge $e \in \mathcal{T}$ (recall $\mathcal{T}$ denotes the routing tree formed by attack paths) and ξ_{ie} the corresponding queueing parameter (if $W_{ie} > 0$), both inferred as in Section 5.3. Let $h_{ek} \in \{0, 1\}$ indicate whether attack path $s_A \to t_{Ak}$ traverses edge e, $\beta_i > 0$ denote the importance of target path $s_{Bi} \to t_{Bi}$, and $\tilde{r}_e$ denote the minimum residual capacity of links from s_A to e (excluding e) before attack. Given a total attack rate λ, the attacker wants to find the rate allocation $\bar{\boldsymbol{\lambda}} := (\bar{\lambda}_k)_{k=1}^{N_A}$ that maximizes the weighted sum average delay of all the target paths, i.e.,

$$\max f(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e \in \mathcal{T}: W_{ie} > 0} d(\xi_{ie}; \sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k) \tag{5.7a}$$

$$\text{s.t.} \quad \sum_{k=1}^{N_A} \bar{\lambda}_k \leq \lambda, \tag{5.7b}$$

$$\sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k \leq \tilde{r}_e, \quad \forall e \in \mathcal{T}, \tag{5.7c}$$

$$\bar{\lambda}_k \geq 0, \quad k = 1, \ldots, N_A, \tag{5.7d}$$

where $d(\xi_{ie}; \bar{\lambda})$ represents the average queueing and transmission delay of the link shared between edge $e \in \mathcal{T}$ and target path $s_{Bi} \to t_{Bi}$.

Remark: First, (5.7a) excludes both the propagation delays and the queueing and transmission delays at links on the target paths that are not shared with any attack path, because these delays are not affected by the attack traffic and thus only contribute a constant shift. *Moreover*, the attacker does not need to know the exact locations of the shared links and their relationships. To explain this, let e_i denote the link shared between edge $e \in \mathcal{T}$ and path $s_{Bi} \to t_{Bi}$. We observe that: (i) e_i will experience the same load $\sum_{k=1}^{N_A} h_{ek} \bar{\lambda}_k$ from attack traffic regardless of its exact location on e, and (ii) even if e_i and e_j for $i \neq j$ have some overlap (i.e., sharing links in the underlying network), the load imposed by $s_{Bj} \to t_{Bj}$ on e_i is part of the background traffic that has been incorporated into the parameter ξ_{ie} and vice versa.

5.4.2 Attack Design

We now derive explicit solutions to (5.7) under each of the queueing models considered in Section 5.3.3.2. When the attacker can destabilize the queue at some shared link, i.e., $\exists i \in \{1, \ldots, N_B\}$ and $e \in \mathcal{T}$ with $W_{ie} > 0$ such that $\sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k \geq r_{ie}$ for some $\bar{\boldsymbol{\lambda}}$ satisfying (5.7b)–(5.7d) (r_{ie}: residual capacity of the shared link e_i excluding attack traffic), then the attacker should simply allocate sufficient traffic to the attack paths traversing e to congest the shared link e_i and drive the average delay of path $s_{Bi} \to t_{Bi}$ (and hence (5.7a)) to infinity. Thus, below we will focus on the nontrivial case when

$$\max_{\text{s.t. (5.7b)–(5.7d)}} \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k < \min_{i \in \{1,\ldots,N_B\}: W_{ie}>0} r_{ie}, \quad \forall e \in \mathcal{T}. \tag{5.8}$$

We will show that in this case, the optimal attack strategy is similar under all the considered queueing models.

5.4.2.1 Attack under M/M/1

When modeling each shared link as an M/M/1 queue, plugging (5.4) into (5.7a) yields

$$f_{\text{M/M/1}}(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e \in \mathcal{T}: W_{ie}>0} \frac{1}{r_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k}, \tag{5.9}$$

which has the following property:

Lemma 5.4.1. *Under* (5.8)*, $f_{M/M/1}(\bar{\boldsymbol{\lambda}})$ is convex in the feasible region of* (5.7)*.*

The convexity of the objective function implies the following property of the optimal solution:

Theorem 5.4.1. *Under* (5.8)*, the solution $\bar{\boldsymbol{\lambda}}^*$ that maximizes $f_{M/M/1}(\bar{\boldsymbol{\lambda}})$ s.t.* (5.7b)–(5.7d) *must achieve "=" for N_A of the constraints.*

Corollary 5.4.1.1. *Under $\lambda \leq \min_{e \in \mathcal{T}} \tilde{r}_e$ and* (5.8)*, the solution $\bar{\boldsymbol{\lambda}}^*$ that maximizes $f_{M/M/1}(\bar{\boldsymbol{\lambda}})$ s.t.* (5.7b)–(5.7d) *must satisfy $\bar{\lambda}_k^* = \lambda$ for some $k \in \{1, \ldots, N_A\}$ and $\bar{\lambda}_{k'}^* = 0$ for all $k' \in \{1, \ldots, N_A\} \setminus \{k\}$.*

For a resource-constrained attacker that faces the case in Corollary 5.4.1.1, our analysis shows that the optimal attack strategy is to enumerate all the N_A candidate solutions, each allocating all the attack rate onto a single attack path, and pick the solution maximizing $f_{\text{M/M/1}}(\bar{\boldsymbol{\lambda}})$.

5.4.2.2 Attack under M/D/1

When modeling each shared link as an M/D/1 queue, plugging (5.5) into (5.7a) yields

$$f_{\mathrm{M/D/1}}(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e\in\mathcal{T}:W_{ie}>0} \frac{2\mu_{ie} - \lambda_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k}{2\mu_{ie}(\mu_{ie} - \lambda_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k)}, \tag{5.10}$$

where μ_{ie} and λ_{ie} are the service/arrival rate at the link shared between $s_{Bi} \to t_{Bi}$ and $e \in \mathcal{T}$ before attack. This objective function has a property similar to $f_{\mathrm{M/M/1}}$:

Lemma 5.4.2. *Under* (5.8) *(where* $r_{ie} := \mu_{ie} - \lambda_{ie}$*),* $f_{M/D/1}(\bar{\boldsymbol{\lambda}})$ *is convex in the feasible region of* (5.7)*.*

The same argument as in the proofs of Theorem 5.4.1 and Corollary 5.4.1.1 leads to a similar attack design under M/D/1:

Theorem 5.4.2. *Under* (5.8)*, the solution* $\bar{\boldsymbol{\lambda}}^*$ *that maximizes* $f_{M/D/1}(\bar{\boldsymbol{\lambda}})$ *s.t.* (5.7b)–(5.7d) *must achieve "=" for* N_A *of the constraints. Furthermore, if* $\lambda \le \min_{e\in\mathcal{T}} \tilde{r}_e$*, then* $\bar{\boldsymbol{\lambda}}^*$ *must satisfy* $\bar{\lambda}_k^* = \lambda$ *for some* $k \in \{1,\dots,N_A\}$ *and* $\bar{\lambda}_{k'}^* = 0$ *for all* $k' \in \{1,\dots,N_A\} \setminus \{k\}$*.*

5.4.2.3 Attack under G/G/1

When each shared link is modeled as a G/G/1 queue, plugging (5.6) into (5.7a) yields

$$f_{\mathrm{G/G/1}}(\bar{\boldsymbol{\lambda}}) := \sum_{i=1}^{N_B} \beta_i \sum_{e\in\mathcal{T}:W_{ie}>0} \frac{\lambda_{ie} + \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k}{2\mu_{ie}(\mu_{ie} - \lambda_{ie} - \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k)} \cdot \left(\sigma_{aie}^2\Big(\lambda_{ie} + \sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k\Big)^2 + \sigma_{sie}^2\mu_{ie}^2\right), \tag{5.11}$$

where we have omitted the average service time (i.e., transmission delay) $1/\mu_{ie}$ as it does not depend on the attack traffic. This function is again convex as stated below:

Lemma 5.4.3. *Under* (5.8)*,* $f_{G/G/1}(\bar{\boldsymbol{\lambda}})$ *is convex in the feasible region of* (5.7)*.*

By the same argument as in Theorem 5.4.1 and Corollary 5.4.1.1, Lemma 5.4.3 implies the following attack design under G/G/1:

Theorem 5.4.3. *Under* (5.8)*, the solution* $\bar{\boldsymbol{\lambda}}^*$ *that maximizes* $f_{G/G/1}(\bar{\boldsymbol{\lambda}})$ *s.t.* (5.7b)–(5.7d) *must achieve "=" for* N_A *of the constraints. Furthermore, if* $\lambda \le \min_{e\in\mathcal{T}} \tilde{r}_e$*, then* $\bar{\boldsymbol{\lambda}}^*$ *must satisfy* $\bar{\lambda}_k^* = \lambda$ *for some* $k \in \{1,\dots,N_A\}$ *and* $\bar{\lambda}_{k'}^* = 0$ *for all* $k' \in \{1,\dots,N_A\} \setminus \{k\}$*.*

Remark: The above analysis shows that the optimal strategy for a resource-constrained attacker is to focus all the attack traffic on a single attack path, selected based on the locations and parameters of the shared links learned through the reconnaissance techniques presented in Section 5.3.

5.4.3 Other Attack Objectives

If the total attack rate violates (5.8), i.e., the attacker can destabilize the queue for at least one shared link, an objective that can differentiate the different ways of destabilizing queues is needed. As a concrete example, if the attacker wants to cause the worst congestion on any of the shared links, he can maximize the following objective function:

$$\max_{e\in\mathcal{T}:\exists W_{ie}>0}\left(\sum_{k=1}^{N_A} h_{ek}\bar{\lambda}_k - \min_{i\in\{1,\ldots,N_B\}:W_{ie}>0} r_{ie}\right), \tag{5.12}$$

subject to constraints (5.7b)–(5.7d), which will maximize the *maximum excess load* on a shared link. Maximizing (5.12) s.t. (5.7b)–(5.7d) is a maximization of a piece-wise linear convex function under linear constraints, for which the optimal solution must be achieved at an extreme point of the feasible region [173]. In our context, this will be a vertex of the polytope defined by (5.7b)–(5.7d), where "=" is achieved for N_A of the constraints. In the special case of $\lambda \leq \min_{e\in\mathcal{T}} \tilde{r}_e$, (5.7c) is redundant, and thus the optimal attack must allocate all the attack traffic onto a single path as in the case of optimizing the objective (5.7a). Note that the new objective (5.12) is invariant to the queueing model. The above result together with the results of Section 5.4.2 suggests the efficacy of the generic attack strategy that focuses resources on an attack path selected based on the information learned through reconnaissance. When the objectives in (5.7a) and (5.12) are both applicable, (5.7a) is usually a more meaningful objective for the attacker as it represents the end-to-end performance impact across all the target paths. Nevertheless, these are just concrete examples of the attacker's objectives to illustrate the impact of adversarial reconnaissance. What objective is most suitable will depend on the application scenario and is left to future work.

5.5 Performance Evaluation

In this study, we evaluate the performance of our algorithms under **two types** of networks using NS3 [174], a widely-used discrete-event network simulator. First, we conduct simulations in the context of an IP-based backbone network (Section 5.5.1).

Then, we validate our results by repeating the experiments in the context of a 5G Integrated Access and Backhaul (IAB) network (Section 5.5.2), leveraging the 5G-LENA module [175] for the radio access network (RAN).

5.5.1 NS3-based Simulation of Backbone Network

5.5.1.1 Simulation Setup

We simulate an IP-based backbone network based on GtsCe (GTS Central Europe) from the Internet Topology Zoo [138], which is a network with 149 nodes and 193 links. Following [176], we set the link capacities and delays using the dataset from [139], in which all link capacities are treated as 1 Gbps. In Appendix 5.7.2.4, we additionally study a case with higher link capacities, which yields similar results. We generate attack paths by randomly picking a source s_A and N_A destinations $\{t_{Ai}\}_{i=1}^{N_A}$ from the network and computing the shortest paths (in hop count). We generate target paths $\{s_B \to t_{Bi}\}_{i=1}^{N_B}$ similarly, while ensuring that each target path shares at least one link with the attack paths. Here each node in GtsCe represents a point of presence (PoP) so that multiple source/destination hosts can attach to the same node (through 'other links' outside the simulated network). Fig. 5.4 shows an example topology formed by the generated paths.

To evaluate the robustness of our approach, we have examined its performance under two types of background traffic. In the experiments presented here, we generate background traffic by a recently proposed methodology from [177], wherein the background traffic rate for each link is periodically sampled from a log-normal distribution characterized by parameters (μ, σ). In this study, we regenerate the rate every 0.5 ms and set σ as 1. Following [178], each background packet has a size randomly selected from 50, 576 and 1460 bytes with probabilities 0.4, 0.2 and 0.4, respectively. The μ-parameter of background traffic is designed to achieve a total utilization that is randomly distributed in [10%, 50%] prior to attack. In Appendix 5.7.2.2, we provide additional simulation results under background traffic generated according to ON-OFF processes as in Section 5.5.2. Here, we set $N_A = N_B = 10$, while in Appendix 5.7.2.3, we additionally study the case of $N_A = 20$. All the additional studies yield qualitatively similar results.

We configure each link to have a FIFO queue with a large buffer to guarantee no packet loss during the simulation. We set the rate on each target path to 50 Mbps with a constant packet size of 1000 bytes. The packet size on each attack path is 50 bytes for shared link detection and 1000 bytes for parameter inference and attack. We set the importance of target paths to $\beta_i = 1$ for $i = 1, \ldots, N_B$. All our results are based on 20

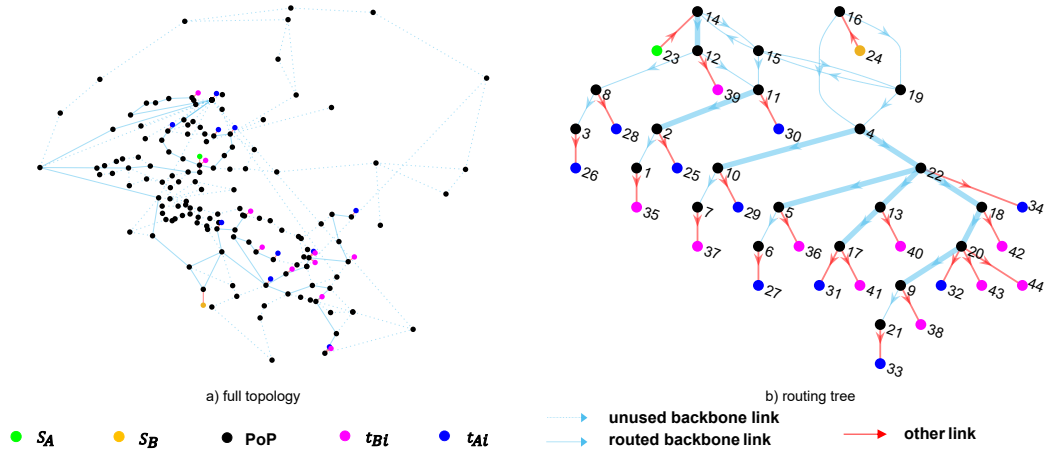


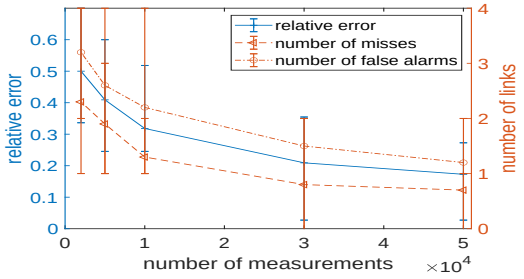
Figure 5.4. Sample topology in the simulation of backbone network ($N_A = N_B = 10$), with shared links highlighted as thick lines.

Monte Carlo runs.

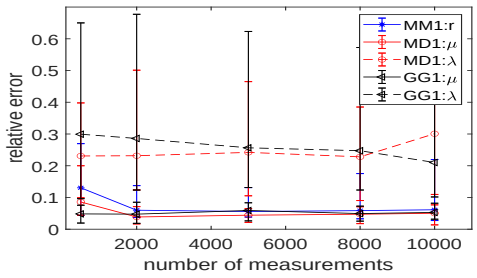
In shared weight inference (Alg. 14), we consider a packet as not incurring queueing on a path if its end-to-end delay is smaller than the mean of the 10 smallest delays on this path plus 0.1 ms. We detect a shared link exists between a target path $s_B \to t_{Bi}$ and an edge $e \in \mathcal{T}$ if the inferred value of W_{ie} exceeds 0.005. To reduce correlation across measurements, we maintain a spacing of at least 2 ms between consecutive measurements. Since the distances from s_A and s_B to the shared links may be different, we find an offset κ by correlation maximization to identify measurements forming a mimicked multicast, as detailed in Appendix 5.7.2.1. In parameter inference, we vary the probing rate among $K = 20$ values evenly distributed between 0 and 80% of the minimum residual capacity at shared links, and solve the least square fitting problem (line 7 in Alg. 17) by the trust-region-reflective least squares algorithm [179].

5.5.1.2 Results on Reconnaissance

Fig. 5.5 (a) shows the accuracy in detecting shared links, measured by the fraction of errors in inferring whether W_{ie} is non-zero for all $i = 1, \ldots, N_B$ and $e \in \mathcal{T}$. In addition, we also evaluate the number of false alarms (detected shared links that do not exist) and the number of misses (shared links that are not detected) averaged over all the target paths. Each measurement here corresponds to a mimicked tri-cast. The results show that our algorithms (Alg. 14–15) can detect the majority of the shared links with some errors (around 20% error if we collect 5×10^4 measurements for each tri-cast for both calibration and detection). Among the errors, there are more false alarms than misses.

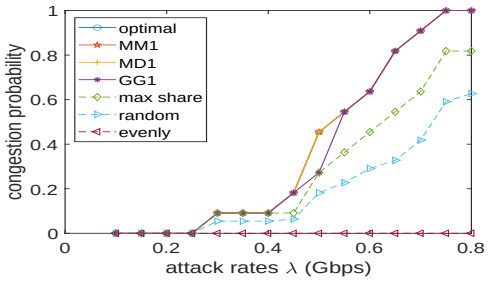


(a) Performance in detecting shared links

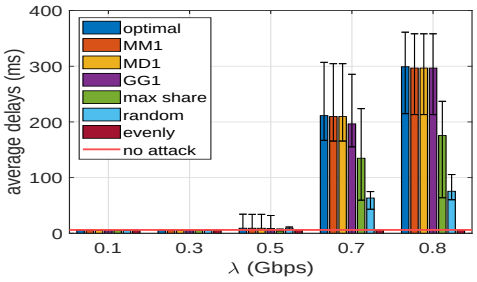


(b) Performance in inferring shared link parameters

Figure 5.5. Performance of reconnaissance in backbone network simulation ($N_A = N_B = 10$).



(a) Probability of congesting at least one shared link



(b) Average delay over all the target paths

Figure 5.6. Performance of attack design in backbone network simulation ($N_A = N_B = 10$).

Fig. 5.5 (b) shows the accuracy of inferring the parameters of the shared links, measured by the relative error $\|\hat{\boldsymbol{\xi}} - \boldsymbol{\xi}^*\|_1 / \|\boldsymbol{\xi}^*\|_1$ ($\hat{\boldsymbol{\xi}}$: estimate, $\boldsymbol{\xi}^*$: ground truth). Although the queues at the links do not exactly follow any of the assumed queueing models, we can still compare the estimated parameters with the best-fitting parameters based on per-link measurements. The results show that: (i) although the real queueing behavior does not exactly fit any of the assumed queueing models, the inference results based on these models are reasonable, (ii) while the link capacities ('μ') and the residual capacities ('r') can be inferred with good accuracy ($< 10\%$ of error), there is notable error in estimating the background traffic loads ('λ'), and (iii) G/G/1-based estimation performs slightly worse due to the difficulty of jointly estimating more parameters. There are two other parameters (variance of interarrival/service time) under G/G/1, for which the trend is similar. Although the inference process involves active probing, each probing experiment only lasts for a short period (e.g., 0.8 seconds for 5000 measurements, each corresponding to a packet on a target path).

5.5.1.3 Results on Attack Design

Since the original design of cross-path attack [13] only ensures to use some attack paths that share at least one link with the target paths, we compare the proposed attack design

with the following intuitive rate allocation strategies over such attack paths[3]:

1. 'Evenly': A natural strategy is to evenly split the total attack rate λ among all the attack paths that share at least one link with the target paths.

2. 'Random': Given that the optimal strategy is usually to focus on one path (see Section 5.4.2), the attacker may also allocate all the rate to a randomly selected attack path.

3. 'Max share': The attacker chooses the attack path traversing the maximum number of shared tree links, i.e., $t_A^* = \max_{t_{Ak} \in T_A} \{\sum_{e \in \mathcal{T}} h_{ek} \mathbb{I}(\sum_{i=1}^{N_B} W_{i,e} > 0)\}$ ($\mathbb{I}(\cdot)$: indicator function).

The results presented in Fig. 5.6 show that despite containing notable error, the information obtained by our reconnaissance algorithms is still useful for attack design. Here, 'optimal' denotes the optimal attack designed based on the true locations/parameters of the shared links, and 'MM1'/'MD1'/'GG1' denotes the proposed attack design based on the parameters inferred from 10,000 measurements under the model of M/M/1 or M/D/1 or G/G/1. Fig. 5.6 (a) shows the probability that the attack can congest (i.e., destabilize) at least one shared link. The results show that: (i) the proposed reconnaissance-based optimized attack design ('MM1', 'MD1', 'GG1') can achieve near-optimal impact despite the notable estimation errors, (ii) the non-optimized attack strategies based on [13] ('random', 'even') are much less effective, and (iii) knowing the locations of shared links ('max share') helps but is not enough. A closer examination shows that the estimated parameters can reveal which attack paths traverse the weakest shared link (the one with the minimum residual capacity), even if the estimated parameters are inaccurate. In Fig. 5.6 (b), we evaluate the impact of attacks on the delays of the target paths, computed over 10,000 measurements. As the objective of delay maximization is only meaningful at attack rates that are within the stability region, we combine multiple attack designs as follows: when $\lambda \leq \min_{e \in \mathcal{T}} \tilde{r}_e$ which satisfies the condition of Corollary 5.4.1.1, the attacker will send all the attack traffic on the attack path predicted to maximize the delay increase over all the target paths; when $\lambda > \min_{e \in \mathcal{T}} \tilde{r}_e$, the attacker will maximize the maximum excess load (5.12) as in Section 5.4.3. We observe that (i) the proposed attack designs produce near-optimal delay increase regardless of the assumed queueing model, and (ii) there is a wide variation among the impacts of different cross-path attacks,

[3]The set of attack paths sharing at least one link with the target paths can be inferred by a simple reconnaissance method proposed in [13]. Here, we use the true set for a conservative comparison.

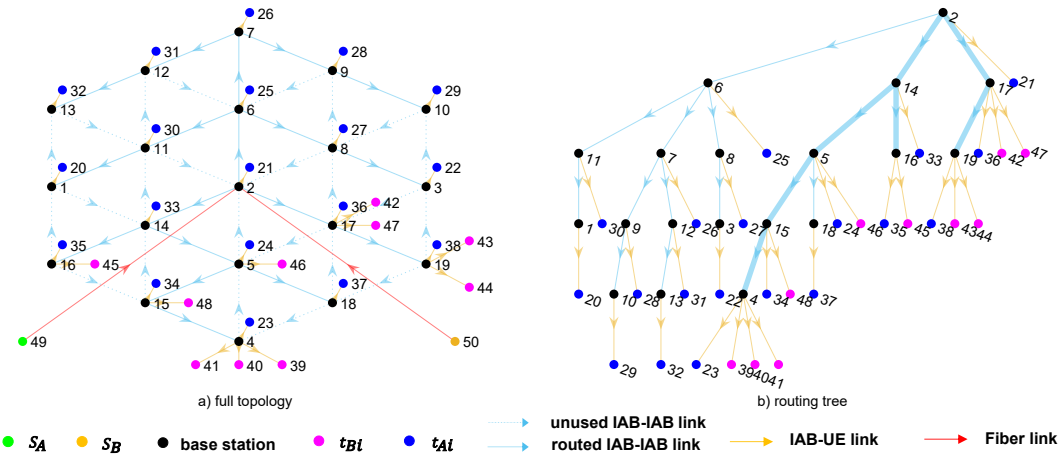


Figure 5.7. Sample topology in the simulation of IAB network ($N_A = 19, N_B = 10$), with shared links highlighted as thick lines.

where the carefully-designed attacks can generate a substantially higher performance impact than the straightforward attacks. These observations signal the importance of considering intelligent attack models in security analysis.

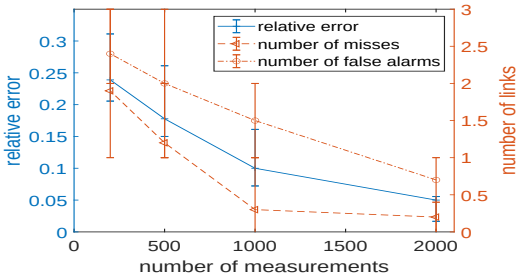
5.5.2 NS3-based Simulation of Integrated Access and Backhaul (IAB) Network

5.5.2.1 Simulation Setup

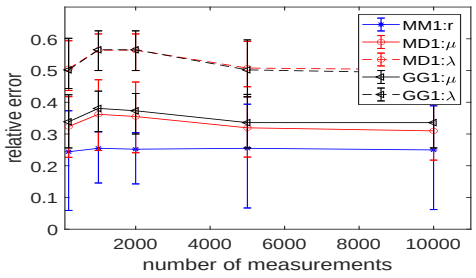
To test the generalizability of our observations, we repeat our experiments in the scenario of an IAB network with multiple slices. IAB network is a form of backhaul for 5G [180], where base stations (BS) are implemented as IAB nodes, among which only a subset of nodes (called IAB donors) are connected to the 5G core through fiber. An IAB node has both a DU and a mobile termination function. Thus, it can function not only as a traditional BS for UEs, but also as a relay for other IAB nodes through millimeter wave. In the process of downlink transmission, parent IAB nodes relay traffic to their child IAB nodes, and the process is reversed for uplink transmission. We simulate the IAB-UE links by 5G-LENA [175], which is a pluggable module in NS3 for simulating 5G RAN, and the rest of the links by point-to-point links[4].

Following [180], we consider an IAB network with 19 BSs in a hexagonal topology

[4]Although the IAB-IAB links are supposed to be through millimeter wave [180], this feature is not officially supported in NS3 to our knowledge, and hence we mimic them by point-to-point links with lower capacities than the fiber links.



(a) Performance in detecting shared links



(b) Performance in inferring shared link parameters

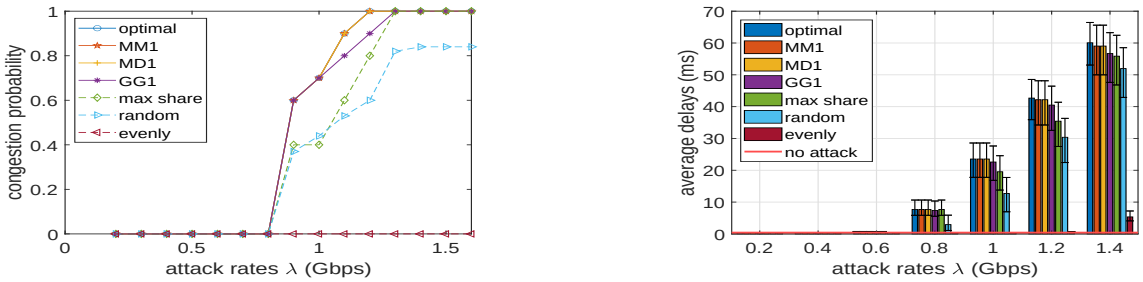
Figure 5.8. Performance of reconnaissance in IAB network simulation ($N_A = 19, N_B = 10$).

with one IAB donor at the center as illustrated in Fig. 5.7 (a). The network is shared by a slice A containing attack paths, a slice B containing target paths, and other slices treated as background traffic. We focus on downlink communication, where packets enter the IAB network through the IAB donor (node 2) and are then routed towards their destination UEs along a routing tree rooted at the donor. The links in the routing tree are highlighted as thick lines in Fig. 5.7 (a) and also depicted in Fig. 5.7 (b). We assume that there is at least one UE in slice A in each cell, and the UEs in slice B are randomly distributed among the cells. According to [180], we assign each slice a separate Bandwidth Part (BWP) for the IAB-UE links. We set the numerology in 5G-LENA to 5.

Following [180, 181], we set the capacity of IAB-IAB links to 2 Gbps, IAB-UE links to 0.5 Gbps, and fiber links to 100 Gbps. We limit the total flow rate to each cell in slice A to 1 Gbps. We set the flow rate for each UE in slice B to 0.1 Gbps to represent emerging applications like panoramic video streaming [182, 183]. Following [140, 184], we independently generate background traffic on each IAB-IAB link according to an ON-OFF process. The duration of each ON period follows the Pareto distribution with shape parameter set to 2.04 and scale parameter ζ_{ON} set to the average length of 13 packets. The duration of each OFF period follows the same distribution with a different scale parameter ζ_{OFF}, tuned to yield a link utilization randomly drawn from $[15\%, 35\%]$. To detect no queueing events for shared weight inference, we measure the delays during light traffic and set a threshold based on the 3σ rule. The rest of the setup is the same as that in Section 5.5.1. In the sequel, we will present our results in the case of $N_B = 10$. More results are given in Appendix 5.7.3.

5.5.2.2 Results on Reconnaissance

First, we evaluate the accuracy of shared link detection as in Fig. 5.5 (a). The error in shared link detection is shown in Fig. 5.8 (a). The results show similar observations as



(a) Probability of congesting at least one shared link

(b) Average delay over all the target paths

Figure 5.9. Performance of attack design in IAB network simulation ($N_A = 19, N_B = 10$).

Fig. 5.5 (a): the proposed algorithms (Alg. 14–15) can detect the shared links with good accuracy ($< 5\%$ error), and the errors are mostly due to false alarms.

In Fig. 5.8 (b), we evaluate the accuracy of parameter inference under each of the queueing models as in Fig. 5.5 (b). Similar to Fig. 5.5 (b), we observe that (i) the residual capacity ('r') and the capacity ('μ') can be estimated more accurately than the load ('λ'), and (ii) G/G/1-based estimation performs slightly worse. The main difference from Fig. 5.5 (b) is that the errors become larger. This is because the delays in the IAB network are affected by not only queueing in the backhaul but also MAC scheduling at the IAB-UE links. We also notice that the proposed parameter estimation method can help detect false alarms in shared link detection, as detailed in Appendix 5.7.4.

5.5.2.3 Results on Attack Design

We evaluate our attack design in comparison with the same benchmarks as in Section 5.5.1.3. The results, presented in Fig. 5.9, show similar observations as Fig. 5.6: (i) the attacks designed based on the results of our reconnaissance algorithms ('MM1', 'MD1', 'GG1') perform close to the optimal in terms of both the probability of congestion and the delay increase, and (ii) the proposed optimized attacks generate a higher performance impact than the straightforward attacks according to [13], especially under a limited total attack rate. Compared to Fig. 5.6, the gap between the optimized attacks and the baselines is smaller in Fig. 5.9. This is because on the average more links are shared between the attack paths and the target paths in the IAB network due to the single ingress point (the IAB donor), as shown in Fig. 5.7, making it easier to impact the target paths by launching attack on randomly selected attack paths.

5.6 Concluding Discussion

We studied a newly identified stealthy DoS attack called cross-path attack, with focus on quantifying the maximum impact of such attacks through optimized attack design. To this end, we developed a novel extension to network topology inference that allows the attacker to consistently estimate the locations and parameters of the links shared between the attack paths and the target paths by only passively monitoring the target paths, and provided an efficient method to compute the optimal attack rate allocation based on the estimated information. Our optimized attack achieved a much greater performance impact than its non-optimized counterparts in high-fidelity simulations. Besides quantifying the maximum impact of cross-path attacks, our work also sheds light on possible defenses. The root cause of such attacks is the sharing of network resources across flows of different levels of security. Although completely isolating flows (e.g., by assigning each flow a fixed share of bandwidth) can prevent cross-path attacks, it also sacrifices the benefits of resource sharing such as throughput elasticity and higher resource utilization. Meanwhile, allowing unlimited resource sharing will make the network vulnerable to malicious abuses of the shared resources as demonstrated in our work. Intuitively, an effective network design should strike a balance between the benefit of elastic resource allocation and the risk of abused elasticity. Determining the right balance will depend on a variety of factors, such as the capacity of the resource, the criticality of the supported application, and the perceived level of threat, which may vary over time. Due to the inherent ambiguity between attack traffic maliciously consuming resources and normal bursty traffic genuinely in need of more resources, the network will face an inevitable tradeoff between performance and security, the detailed investigation of which is left to future work.

5.7 Appendices

5.7.1 Appendix A: Proofs of Theorems

Proof of Theorem 5.3.2. First, we prove that given accurate estimates of the category weights, the shared weight on the stem of each tree considered by Alg. 15 will be accurately inferred. Since $s_A \to \tau_1$ and $s_A \to \tau_2$ branches at $b_{\mathcal{T}'}$, ρ_s is the shared weight on $s_A \to b_{\mathcal{T}'}$. Moreover, as Alg. 15 works in a top-down manner, the shared weights on edges above $s_{\mathcal{T}'}$ must have been inferred before considering $\mathcal{T}'$. Thus, $\rho_s - W_{s_A \to s_{\mathcal{T}'}}$ (line 4) must be the true shared weight on edge $(s_{\mathcal{T}'}, b_{\mathcal{T}'})$.

Moreover, we prove that every edge with non-zero shared weight will be the stem of a tree considered by Alg. 15. After inferring the shared weight on the stem of $\mathcal{T}'$, Alg. 15 will perform recursion for both subtrees of $\mathcal{T}'$ to consider the remaining edges except for two cases. The first case is when $\delta_{l\mathcal{T}'} = \delta_{r\mathcal{T}'}$ (line 5), in which case $\mathcal{T}'$ has no other edge. The second case is when $\rho_s \neq 0$ and $\rho_l = 0$ (or $\rho_r = 0$), in which case we can skip the left (or right) subtree of $\mathcal{T}'$ as all its edges have zero shared weight. To see this, suppose that $\rho_s \neq 0$ and $\rho_l = 0$, but $\exists$ edge e in the left subtree of $\mathcal{T}'$ with $W_e \neq 0$. Let e' be the stem of the left subtree. Suppose that $s_B \to t_B$ intersects with $s_A \to b_{\mathcal{T}'}$ at node v_1 (which exists because $\rho_s \neq 0$), and intersects with e at node v_2 (which exists because $W_e \neq 0$). Then there exist two routing paths between v_1 and v_2, one follows $\mathcal{T}$ and traverses e', and the other follows $s_B \to t_B$ without traversing e' (as $\rho_l = 0$), which contradicts with the unique route assumption in Section 5.2.1. Similar argument holds for $\rho_s \neq 0$ and $\rho_r = 0$. □

Proof of Theorem 5.3.3. First, we argue that all the shared inks will be considered in parameter estimation. As the given weight vector $\boldsymbol{W}$ is accurate and all the shared links have non-zero metrics, every edge e containing a shared link will have $W_e > 0$, and thus will be considered in the parameter estimation when e is the stem of the tree $\mathcal{T}'$ under consideration. As the recursion examines the edges of $\mathcal{T}$ in a top-down manner, it remains to show that when $W_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})} > 0$, we can safely skip one of the subtrees of $\mathcal{T}'$ as long as $\delta_{l\mathcal{T}'} \neq \delta_{r\mathcal{T}'}$ (otherwise $\mathcal{T}'$ only has one edge, i.e., the stem). This is because conditioned on $W_{(s_{\mathcal{T}'}, b_{\mathcal{T}'})} > 0$, $s_B \to t_B$ cannot intersect with both of the subtrees of $\mathcal{T}'$, or there will be a contradiction with the unique route assumption in Section 5.2.1.

Next, we argue that the parameter for each shared link considered in lines 2–7 of Alg. 17 will be estimated accurately. We start by considering the top-most shared link, assumed to reside on an edge $e \in \mathcal{T}$. Our selection of the probing destination τ^* ensures that it is the only shared link between $s_B \to t_B$ and $s_A \to \tau^*$, for which the objective of the least square fitting in line 7 is reduced to $\sum_{k=1}^{K} \left(\psi_k - c_{\tau^*} - d(\xi_e; \bar{\lambda}_k)\right)^2$. Let $(c^*_{\tau^*}, \xi^*_e)$ denote the ground truth parameters. By our assumption, $(c^*_{\tau^*}, \xi^*_e)$ achieves a zero fitting error. Suppose that the estimated parameters $(\hat{c}_{\tau^*}, \hat{\xi}_e) \neq (c^*_{\tau^*}, \xi^*_e)$. Then $(\hat{c}_{\tau^*}, \hat{\xi}_e)$ must also achieve a zero fitting error, i.e.,

$$\hat{c}_{\tau^*} + d(\hat{\xi}_e; \bar{\lambda}_k) = c^*_{\tau^*} + d(\xi^*_e; \bar{\lambda}_k), \quad k = 1, \ldots, K. \tag{5.13}$$

Under M/M/1, plugging (5.4) into (5.13) implies that $\bar{\lambda}_k$ $(k = 1, \ldots, K)$ must all satisfy

$$\hat{c}_{\tau^*} + \frac{1}{\hat{r}_e - \bar{\lambda}} = c^*_{\tau^*} + \frac{1}{r^*_e - \bar{\lambda}}. \tag{5.14}$$

For $K > 2$, this leads to a contradiction as (5.14) is a quadratic equation in $\bar{\lambda}$ with at most two distinct solutions. Similarly, under M/D/1, plugging (5.5) into (5.13) gives a quadratic equation of $\bar{\lambda}$ with at most two distinct solutions, contradicting with $K > 2$; under G/G/1, plugging (5.6) into (5.13) gives a quartic equation of $\bar{\lambda}$ with at most four distinct solutions, contradicting with $K > 4$. The same argument applies to every other shared link, as our selection of the probing destination ensures that when estimating ξ_e, all the other shared links between the target path and the probing path are above e, whose parameters should already be accurately inferred by induction. □

Proof of Lemma 5.4.1. As $f_{\mathrm{M/M/1}}$ is a non-negative linear combination of functions of the form $g_1(\bar{\boldsymbol{\lambda}}) := \frac{1}{p - \sum_{i=1}^{N_A} q_i \bar{\lambda}_i}$, where $p - \sum\limits_{i=1}^{N_A} q_i \bar{\lambda}_i > 0$, it suffices to prove that $g_1(\bar{\boldsymbol{\lambda}})$ is convex.

To this end, it suffices to show that for any $\bar{\boldsymbol{\lambda}}_j$ $(j = 1, 2)$ satisfying $p - \sum\limits_{i=1}^{N_A} q_i \bar{\lambda}_{ji} > 0$, $g_1(\bar{\boldsymbol{\lambda}}_1) + g_1(\bar{\boldsymbol{\lambda}}_2) \geq 2g_1(\frac{\bar{\boldsymbol{\lambda}}_1 + \bar{\boldsymbol{\lambda}}_2}{2})$, since a continuous function that is midpoint convex must be convex [185]. The proof completes by ($\Leftrightarrow$ means equivalence):

$$\begin{aligned}
&\frac{1}{p - \sum\limits_{i=1}^{N_A} q_i \bar{\lambda}_{1i}} + \frac{1}{p - \sum\limits_{i=1}^{N_A} q_i \bar{\lambda}_{2i}} \geq \frac{2}{p - \sum\limits_{i=1}^{N_A} q_i \frac{\bar{\lambda}_{1i} + \bar{\lambda}_{2i}}{2}} \\
&\Leftrightarrow (\sum_{i=1}^{N_A} q_i \bar{\lambda}_{1i})(\sum_{i=1}^{N_A} q_i \frac{\bar{\lambda}_{1i} + \bar{\lambda}_{2i}}{2}) \\
&+ (\sum_{i=1}^{N_A} q_i \bar{\lambda}_{2i})(\sum_{i=1}^{N_A} q_i \frac{\bar{\lambda}_{1i} + \bar{\lambda}_{2i}}{2}) \geq 2(\sum_{i=1}^{N_A} q_i \bar{\lambda}_{1i})(\sum_{i=1}^{N_A} q_i \bar{\lambda}_{2i}) \\
&\Leftrightarrow (\sum_{i=1}^{N_A} q_i \bar{\lambda}_{1i})^2 + (\sum_{i=1}^{N_A} q_i \bar{\lambda}_{2i})^2 \geq 2(\sum_{i=1}^{N_A} q_i \bar{\lambda}_{1i})(\sum_{i=1}^{N_A} q_i \bar{\lambda}_{2i}) \\
&\Leftrightarrow \left(\sum_{i=1}^{N_A} q_i \bar{\lambda}_{1i} - \sum_{i=1}^{N_A} q_i \bar{\lambda}_{2i}\right)^2 \geq 0.
\end{aligned} \tag{5.15}$$

□

Proof of Theorem 5.4.1. By Lemma 5.4.1, the attacker's optimization is a maximization of a convex function over a polytope defined by (5.7b)–(5.7d), for which the optimal solution must be achieved at an extreme point of the feasible region [173]. In our context,

this will be a vertex of the polytope, which achieves "=" for N_A of the constraints in (5.7b)–(5.7d). □

Proof of Corollary 5.4.1.1. Under $\lambda \leq \min_{e\in\mathcal{T}} \tilde{r}_e$, the constraint in (5.7c) can be ignored. The remaining constraints define a polytope with only N_A non-zero vertices, each in the form of $\bar{\lambda}^*_k = \lambda$ and $\bar{\lambda}^*_{k'} = 0$ for all $k' \in \{1, \ldots, N_A\} \setminus \{k\}$. The optimal solution must be one of them by Theorem 5.4.1. □

Proof of Lemma 5.4.2. As $f_{\mathrm{M/D/1}}$ is a non-negative linear combination of functions of the form

$$g_2(\bar{\boldsymbol{\lambda}}) := \frac{2\mu - \lambda - \sum_{i=1}^{N_A} q_i \bar{\lambda}_i}{2\mu(\mu - \lambda - \sum_{i=1}^{N_A} q_i \bar{\lambda}_i)}, \tag{5.16}$$

where $\mu - \lambda - \sum_{i=1}^{N_A} q_i \bar{\lambda}_i > 0$, it suffices to prove that $g_2(\bar{\boldsymbol{\lambda}})$ is convex. To this end, note that

$$g_2(\bar{\boldsymbol{\lambda}}) = \frac{1}{2\mu} + \frac{1}{2(\mu - \lambda - \sum_{i=1}^{N_A} q_i \bar{\lambda}_i)} = \frac{1}{2\mu} + \frac{1}{2} g_1(\bar{\boldsymbol{\lambda}}), \tag{5.17}$$

where $g_1(\bar{\boldsymbol{\lambda}})$ is defined as in the proof of Lemma 5.4.1 with $p := \mu - \lambda$. Since $g_1(\bar{\boldsymbol{\lambda}})$ is convex, $g_2(\bar{\boldsymbol{\lambda}})$ is convex. □

Proof of Lemma 5.4.3. Function $f_{\mathrm{G/G/1}}$ is a non-negative linear combination of functions of the form

$$g_3(\bar{\boldsymbol{\lambda}}) := \frac{\theta}{t - \theta}(\theta^2 + s) \tag{5.18}$$

with $\theta := p + \sum_{i=1}^{N_A} q_i \bar{\lambda}_i$, where $\theta \geq 0$, $t - \theta > 0$, and $s > 0$. Thus, it suffices to prove that $g_3(\bar{\boldsymbol{\lambda}})$ is convex.

To this end, note that

$$\frac{\partial g_3}{\partial \theta} = \frac{3\theta^2(t-\theta) + \theta^3}{(t-\theta)^2} + \frac{st}{(t-\theta)^2} > 0, \tag{5.19}$$

$$\frac{\partial^2 g_3}{\partial \theta^2} = \frac{6t\theta}{(t-\theta)^2} + \frac{2\theta^3}{(t-\theta)^3} + \frac{2st}{(t-\theta)^3} > 0, \tag{5.20}$$

i.e., g_3 is an increasing convex function of θ. Since θ is a linear function of $\bar{\boldsymbol{\lambda}}$, g_3 is a convex function of $\bar{\boldsymbol{\lambda}}$. □

5.7.2 Appendix B: Supplementary Evaluation Results for Backbone Network

5.7.2.1 Measurement Calibration in NS3 Simulation of Backbone Network

As discussed in Section 5.5.1.1, during shared weight inference, we need to estimate an offset κ between measurements on a pair of probed attack paths (p_{A1}, p_{A2}) and measurements on a target path p_B to mimic tri-cast, as the delays from s_A and s_B to the links shared by all these paths (if any) may be different. We use the following heuristic to estimate κ.

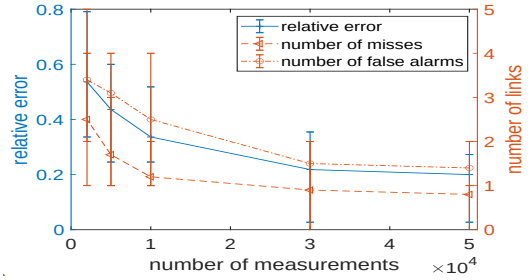
We send a flow on each probed attack path to collect a sequence of end-to-end delay measurements. We also collect end-to-end delays on the target path in the meanwhile. For the target path, we directly transform the delay measurements into a binary sequence of queueing indicators using the threshold given in Section 5.5.1.1, denoted as $\{q_B^t\}_{t=1}^T$, where $q_B^t = 1$ if the t-th measurement is detected to experience queueing and $q_B^t = 0$ otherwise. Since p_{A1}, p_{A2} share the same source s_A, we combine the delay measurements on p_{A1}, p_{A2} by adding the delays of the i-th packets from both paths, and then transform the combined delay measurements into a binary sequence $\{q_A^t\}_{t=1}^T$ as for $\{q_B^t\}_{t=1}^T$. To find κ so that the i-th packet on p_B and the $(i+\kappa)$-th packet pair on (p_{A1}, p_{A2}) traverse the shared links (if any) at approximately the same time, we maximize the correlation between $\{q_B^t\}_{t=1}^T$ and $\{q_A^t\}_{t=1}^T$ by solving

$$\kappa^* = \underset{1-T\leq\kappa\leq T-1}{\arg\max} \frac{1}{\min(T, T-\kappa) - \max(1, 1-\kappa) + 1} \sum_{i=\max(1,1-\kappa)}^{\min(T,T-\kappa)} q_B^i q_A^{i+\kappa}. \tag{5.21}$$

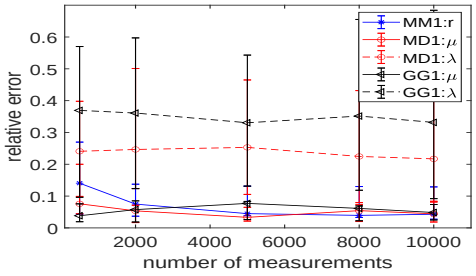
We then identify the i-th packet on p_B and the $(i+\kappa^*)$-th packet pair on (p_{A1}, p_{A2}) as a mimicked tri-cast.

5.7.2.2 NS3 Simulation of Backbone Network under an Alternative Background Traffic Model

In Section 5.5.1, we showed the results in the scenario where the background traffic follows log-normal distribution. In this section, we validate our algorithms under background traffic generated according to ON-OFF process [140, 186, 187]. More specifically, the

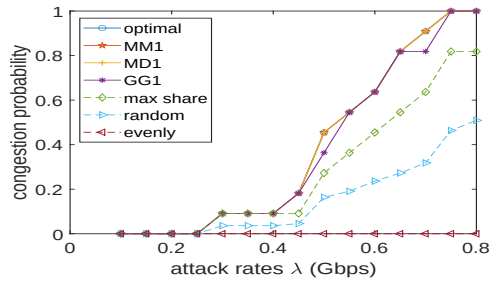


(a) Performance in detecting shared links

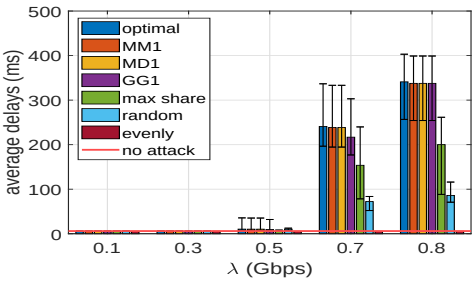


(b) Performance in inferring shared link parameters

Figure 5.10. Performance of reconnaissance in backbone network simulation under ON-OFF background traffic ($N_A = N_B = 10$).



(a) Probability of congesting at least one shared link



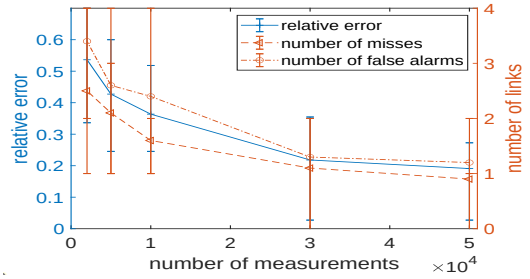
(b) Average delay over all the target paths

Figure 5.11. Performance of attack design in backbone network simulation under ON-OFF background traffic ($N_A = N_B = 10$).

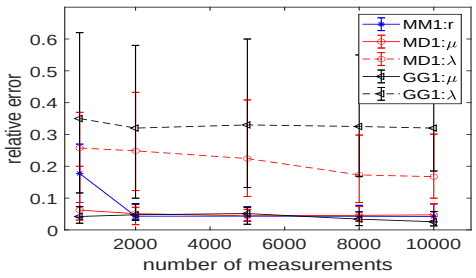
duration of each ON period is sampled from a Pareto distribution with the shape parameter as 2.04 and the scale parameter as the average length of 13 packets. The duration of each OFF period is sampled from the same distribution with a different scale parameter, configured to result in the same utilization of each link as the values used in Section 5.5.1 for log-normally distributed background traffic. The results for reconnaissance are shown in Fig. 5.10 as the counterpart of Fig. 5.5, while the results for attack design are given in Fig. 5.11 as the counterpart of Fig. 5.6. We observe that the results under ON-OFF background traffic are similar to those in Section 5.5.1, which confirms the robustness of the proposed methods under different background traffic patterns.

5.7.2.3 Evaluation Results for NS3 Simulation of Backbone Network with $N_A = 20$

In Section 5.5.1, we evaluate our algorithms with $N_A = 10$. A larger N_A will result in fewer links on each edge in the routing tree $\mathcal{T}$, which makes it harder for Alg. 14 to accurately detect the shared links. To test its impact, we evaluate our algorithms with

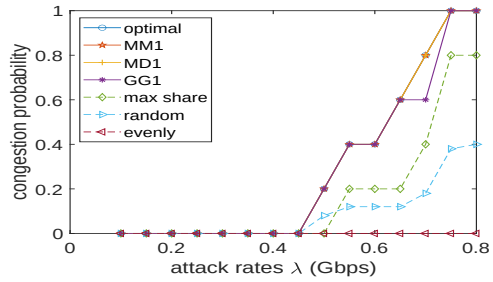


(a) Performance in detecting shared links

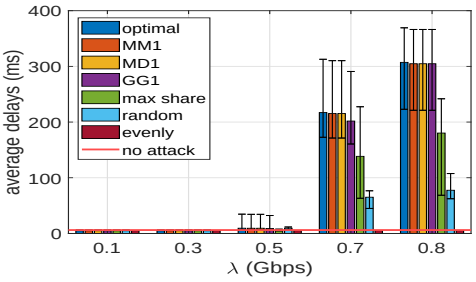


(b) Performance in inferring shared link parameters

Figure 5.12. Performance of reconnaissance in backbone network simulation ($N_A = 20, N_B = 10$).



(a) Probability of congesting at least one shared link



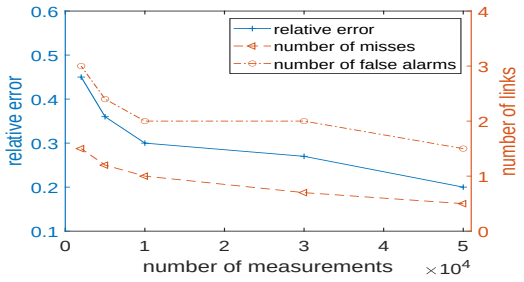
(b) Average delay over all the target paths

Figure 5.13. Performance of attack design in backbone network simulation ($N_A = 20, N_B = 10$).

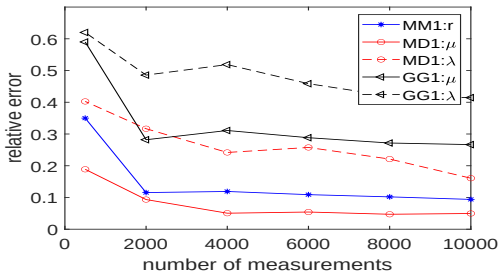
$N_A = 20$ in the same scenario as in Section 5.5. The results are given in Fig. 5.12-5.13, as the counterparts of Fig. 5.5-5.6. We observe that (i) the performance of reconnaissance slightly degraded, but (ii) the attack design still achieved significantly better performance than the baselines (i.e., "max share", "random", and "evenly"). This result demonstrates the robustness of our methods to the number of attack paths. We have also verified that the performance of our methods is not sensitive to the number of target paths.

5.7.2.4 Evaluation Results for NS3 Simulation of Backbone Network with 50 Gbps Link Capacity

Building on Section 5.5.1.1, where the link capacity is normalized to 1 Gbps, this section validates those results under increased link capacities. Specifically, we repeat the NS3 simulation for the backbone network GtsCe with a link capacity of 50 Gbps. To accommodate this, the rate for background traffic is regenerated every 0.05 ms, compared to the previous 0.5 ms in Section 5.5.1.1. Moreover, the flow rate on the target paths has been adjusted from 50 Mbps to 2500 Mbps, and the background traffic rates have been



(a) Performance in detecting shared links



(b) Performance in inferring shared link parameters

Figure 5.14. Performance of reconnaissance in backbone network simulation ($N_A = N_B = 10$).

increased by 50 times too, while all other settings remain the same as Section 5.5.1.1. Results from a single Monte Carlo run are presented below.

The reconnaissance results as the counterpart of Fig. 5.5 are given in Fig. 5.14, in which we observe similar trends as in Fig. 5.5. We then assess the rate at which each attack method induces congestion on at least one shared link, as a counterpart to Fig. 5.6 (a). For this specific Monte Carlo run, the benchmarks "optimal" and "max share", and all the proposed methods (i.e., "MM1"/"MD1"/"GG1") begin to induce congestion when the total attack rate exceeds 48.7% of the link capacity. At this rate, "random" starts exhibiting a non-zero (0.3) probability of causing congestion. Moreover, "random" only reaches a 0.5 congestion probability even when the attack rate surpasses 70% of the link capacity. In contrast, "evenly" fails to induce congestion even when the attack rate reaches 80% of the link capacity.

As the counterpart of Fig. 5.6 (b), we analyze the average delay induced by various attack designs in Fig. 5.15, computed over 30,000 packets on the target paths. We observe that "max share" and all the proposed methods (i.e., "MM1"/"MD1"/"GG1") achieve the same performance as "optimal" since they all correctly identify the attack path traversing the weakest shared link[5]. Notably, the proposed methods markedly outperform the non-optimized benchmarks "random" and "evenly". These findings, which are consistent with Fig. 5.6 (b), underscore the efficacy of our proposed methods.

[5]The absolute delays in Fig. 5.15 are smaller than those in Fig. 5.6 (b) due to the increased link capacity.

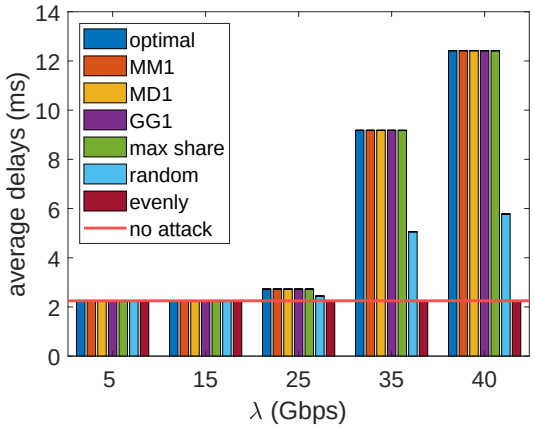
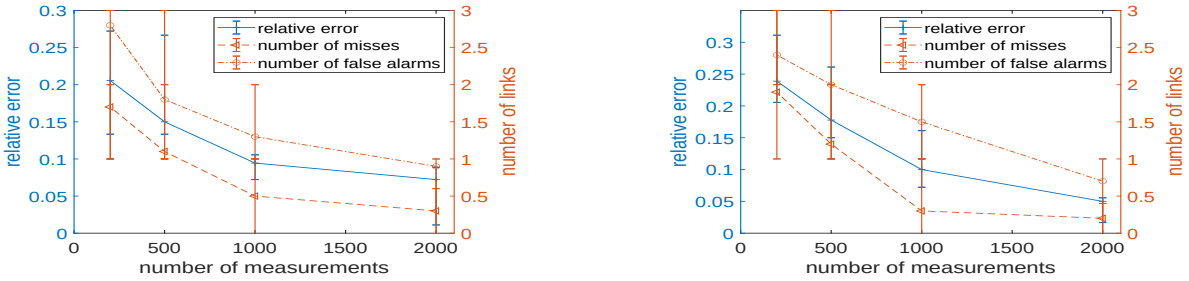


Figure 5.15. Average delay over all the target paths ($N_A = N_B = 10$)).



(a) $N_B = 5$ (b) $N_B = 10$

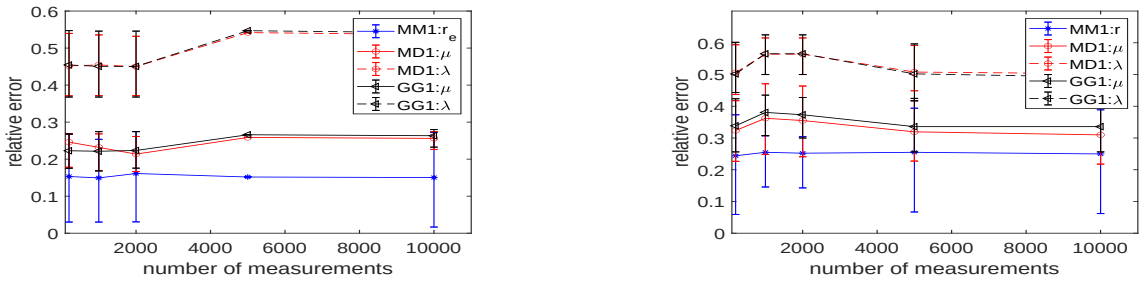
Figure 5.16. Performance in detecting shared links in IAB network simulation.

5.7.3 Appendix C: Supplementary Evaluation Results for Integrated Access and Backhaul (IAB) Network

In this section, we will present the supplementary experimental results for Section 5.5.2 in the case of $N_B = 5$. The previously presented results under $N_B = 10$ are also shown here for comparison.

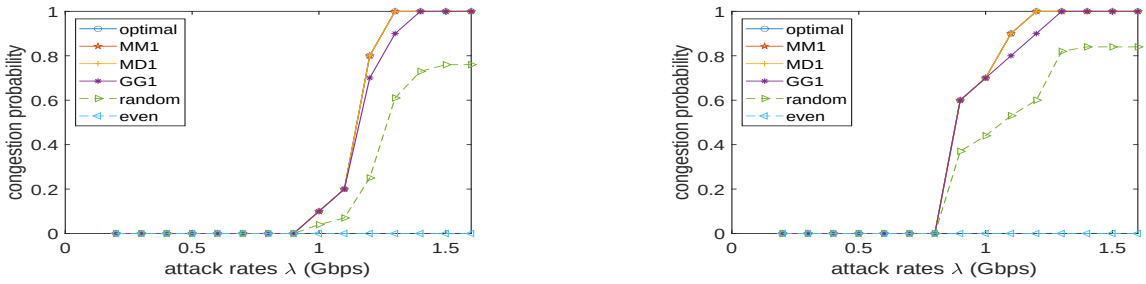
5.7.3.1 Results on Reconnaissance

In Fig. 5.16, we present the performance of shared link detection for different numbers of target paths. We observe that the results are insensitive to the number of target paths N_B. Next, we show the results of parameter estimation for the detected shared links, as given in Fig. 5.17. Again, the observations under different values of N_B are qualitatively similar.



(a) $N_B = 5$ (b) $N_B = 10$

Figure 5.17. Performance in inferring parameters of shared links in IAB network simulation.



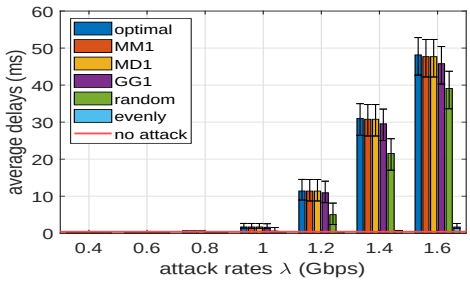
(a) $N_B = 5$ (b) $N_B = 10$

Figure 5.18. Probability that the attack can destabilize the queue for at least one shared link in IAB network simulation.

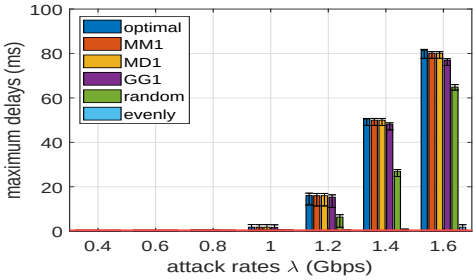
5.7.3.2 Results on Attack Design

We first evaluate the probability that the proposed design with objective (5.12) can destabilize at least one queue, as shown in Fig. 5.18. As before, the results for $N_B = 5$ and $N_B = 10$ show the same trend.

Finally, we compare the delays of target paths under various attack designs in Fig. 5.19-5.20, where Fig. 5.19 (a) and Fig. 5.20 (a) show the overall average delay (averaged over all the target paths), while Fig. 5.19 (b) and Fig. 5.20 (b) show the maximum average delay (maximized over all the target paths). Similar to the results discussed in Section 5.5.2.3, we observe that the proposed attack designs generate higher impacts than the benchmarks, regardless of the number of target paths and the performance metric (either the average delay over all the target paths or the average delay of the worst-performing target path).

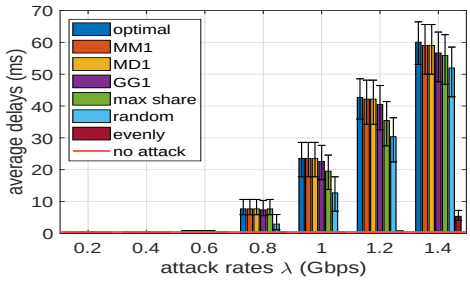


(a) average delays

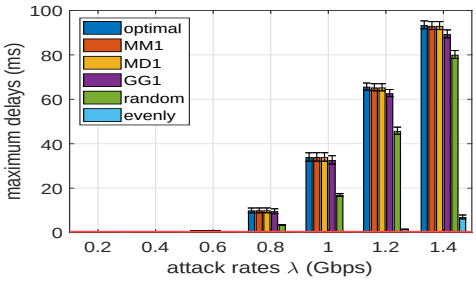


(b) maximum delays

Figure 5.19. Delay increase under different λ in IAB network simulation ($N_B = 5$).



(a) average delays



(b) maximum delays

Figure 5.20. Delay increase under different λ in IAB network simulation ($N_B = 10$).

5.7.4 Appendix D: Discussion on detecting false alarms through parameter estimation

In this section, we will discuss an observation that the proposed parameter estimation method (Alg. 16–17) can help detect the false alarms in shared link detection (based on Alg. 14–15).

In the case of a false alarm, the "shared link" under consideration does not actually exist, and thus varying probing rate (line 6 in Alg. 17) will not impact the average delay of the target path under consideration as expected. This will manifest as an abnormally large estimated link capacity, which can then be used to detect that this "shared link" does not exist. To see the reason, let us consider the example in Fig. 5.7. If Alg. 14 falsely detects $(2, 6)$ to be a shared link for the target path $2 \to 43$ and Alg. 17 tries to estimate its capacity by varying probing rate on the path $2 \to 25$, then the best-fitting capacity will be infinity as the average delay on $2 \to 43$ will not increase with the probing rate on $2 \to 25$. Even if the probing path and the target path have shared links, false alarms may still be detected. For example, suppose that link $(5, 18)$ in Fig. 5.7 is falsely detected as a shared link for the target path $2 \to 46$, and $2 \to 37$ is selected as the probing path

for estimating its parameters, then the delay increase on $2 \rightarrow 46$ caused by the probing on $2 \rightarrow 37$ will be captured by the delay increase on the truly shared links $(2, 14)$ and $(14, 5)$ (if they are detected), still making the best-fitting capacity of link $(5, 18)$ infinity. This observation together with the fact that there are fewer misses than false alarms (see Fig. 5.8 (a)) allows our solution to detect the shared links with high accuracy.

Chapter 6
Conclusion and Future Work

In this section, we first outline the limitations of the studies presented in this dissertation, followed by a discussion on potential future directions for each. Subsequently, we will delve deeper into one of these topics, using it as an illustrative example to explore further.

6.1 Future Work

In Chapter 2, we addressed the issue of power line state estimation following joint cyber-physical attacks. It was assumed that the phase angles for the DC power flow model, as outlined in (2.8), or the voltages for the AC power flow model, as detailed in (2.25), could be restored using the methods described in Section 2.8.3. However, the methods in Section 2.8.3 are applicable if certain topological conditions are satisfied, which may not be the case in practice. Our findings have illustrated that line states can be estimated with high accuracy provided that the post-attack phase angles/voltages are successfully restored. This underscores the critical need for precise recovery of phase angles/voltages across diverse power grid configurations. While initial studies, such as [188], have shown promising outcomes, they lack guaranteed performance. Comprehensive solutions for attack recovery thus require further research.

In Chapter 3, we examined the optimal secure PMU placement strategy to prevent overload-induced line tripping. Our modeling includes some limiting assumptions. Firstly, we presuppose that the load remains constant during line tripping incidents. However, due to preventive control measures [65], the load, particularly the reactive power, might adjust dynamically. Incorporating this factor could introduce more potent attack vectors and present new challenges for developing defensive mechanisms. Additionally, we assume that the power grid invariably achieves a steady state. This quasi-steady-state modeling assumption overlooks the effects of angle stability, frequency stability, and dynamic

voltage stability. Considering the transient phase of line tripping [64, 66] in modeling adversarial behavior necessitates further research. For additional details, we direct the reader to [189] and references therein.

In Chapter 4, we explored congestion-free overlay routing, focusing on the categorization of underlay links and the inference of the associated capacities. While the proposed algorithms are designed to be applicable under any underlay topology, accurately estimating overlay metrics remains a significant challenge that warrants further study. Additionally, the current algorithms operate in an offline mode. Adapting these algorithms for use in a dynamic underlay network by developing an online version would enhance their applicability. Moreover, minimizing routing delay, although crucial, may not suffice for several overlay applications that depend on routing, such as decentralized learning [190]. Therefore, optimizing application-specific quality-of-service, while balancing the trade-off with routing delay, presents an interesting avenue for future research.

In Chapter 5, we examined the optimization of cross-path attacks to maximize their impact on the target path. The proposed methods depend on the ability to estimate the end-to-end delays of target paths to infer the locations of shared network components. This assumption constrains the practical deployment of the proposed approaches. Furthermore, our optimization of attack impact is confined to scenarios where the adversary controls only a single source. The question of how to maximize attack impact when multiple adversarial sources are involved remains an unresolved challenge.

6.2 Illustrative Example: Overlay-Based Decentralized Learning

In this section, we will explore in greater detail the design of overlay networks to accelerate decentralized learning [190], serving as an illustrative example to extend the work presented in this dissertation.

As an emerging machine learning paradigm, *federated learning* allows multiple learning agents to collaboratively learn a shared model from the union of their local data without directly sharing the data [190]. To achieve this goal, the agents repeatedly exchange model updates, through a centralized parameter server [190], a hierarchy of parameter servers [191], or peer-to-peer links between neighboring agents [192], which are then

aggregated to update the shared model. Due to its promise in protecting data privacy, this learning paradigm has found many applications, such as improving mobile apps [193, 194] and browsers [195, 196]. In particular, federated learning via decentralized optimization [197] has attracted significant attention. Instead of forming a star [190] or hierarchical topology [191], agents in *decentralized federated learning (DFL)* can communicate along an arbitrary topology, where parameter exchanges only occur between neighbors. This framework also avoids a single point of failure or hot spot at the central aggregator, and is known to reduce the communication complexity at the busiest node without increasing the computational complexity [197].

Meanwhile, federated learning still faces significant performance challenges due to the extensive data transfer. Although the training data stay local, the agents still need to communicate frequently to exchange local model updates, which incurs a nontrivial communication cost for training deep learning models due to the large model size. Such communication cost can dominate the total cost of the learning task, e.g., up to 90% of time in cloud-based learning is spent on communications [198], and the problem is exacerbated when the agents are distributed across a bandwidth-limited network. This issue has attracted tremendous interests in reducing the communication cost, including compression-based methods for reducing the amount of data in each communication [199–201] and optimizations for reducing the number of communications through hyperparameter optimization [202–206] or adaptive communications [207–210].

However, most existing works make simplistic assumptions about the connection between agents, where each pair of logically adjacent agents is assumed to be connected by a link that incurs a fixed cost when used in communication, regardless of the communications between other agents. This is not true when the agents are connected through an underlying communication network (referred to as an *underlay*), as the connections between different agents may map to multi-hop paths that share links. For example, consider a set of learning agents with the physical connectivity in Fig. 6.1a and the logical connectivity in Fig. 6.1b. Although connections (A, B) and (C, D) appear disjoint to the learning agents, they actually map to paths sharing link (h_1, h_2) in the underlay, and thus concurrent communications over these connections can take longer than stand-alone communication on each of them. Moreover, links in the underlay may have heterogeneous capacities and various loads of background traffic. Most existing works on communication optimization in federated learning have ignored such complications by assuming the communication time to be proportional to the maximum number of neighbors an agent communicates with [205, 206, 211–213]. Such simplistic assumption will lead to incorrect

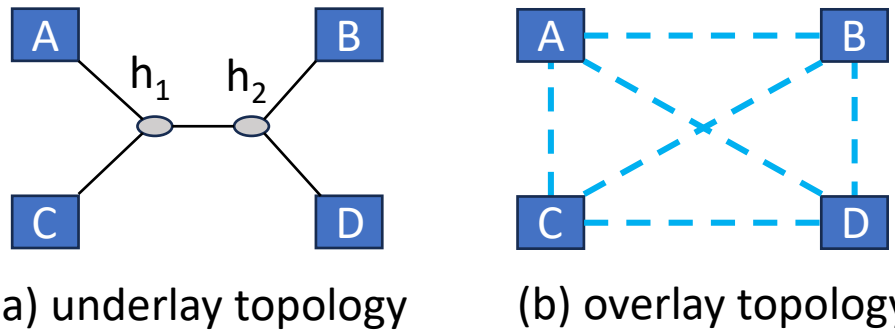


Figure 6.1. Overlay-underlay structure for learning over a communication network (learning agents: $\{A, B, C, D\}$; underlay nodes: $\{h_1, h_2\}$).

prediction of the communication time and suboptimal designs in the case of overlay-based DFL.

We want to address this gap in the context of overlay-based DFL *without requiring explicit cooperation from the underlay network*, i.e., the agents can neither directly observe the internal state of the underlay (e.g., routing topology, link capacities) nor control its internal behavior. Such scenarios arise naturally when the agents are interconnected by a public network controlled by a third party. In particular, we are interested in running DFL over bandwidth-limited underlay networks. In contrast to high-bandwidth underlays such as inter-datacenter networks as considered in [214], bandwidth-limited underlays are more sensitive to communication demands generated by the learning task and can thus benefit more from underlay-aware designs. Examples of such bandwidth-limited underlays include but are not limited to: cellular edge network [215], power line communication [216, 217], device-to-device communication [218], underwater communication networks [219], and multi-hop IoT networks [220].

To this end, we propose an overlay-based framework to jointly design the *communication demands* between learning agents and the *communication schedule* for serving these demands, without explicit cooperation from the underlay. Building upon recent advances in network tomography [221] and mixing matrix design [206], we cast the problem into a set of optimizations that collectively minimize the completion time of the learning task in achieving a given level of convergence.

6.2.0.1 Related Work

Decentralized federated learning. Initially proposed under a parameter server architecture [190], learning from decentralized data was later extended to a fully decentralized architecture [197], which was shown to achieve the same computational complexity but

a lower communication complexity than training via a central server. Since then a number of improvements have been developed, e.g., [222] improved the robustness to data variance, and [223] provided a lower bound on the iteration complexity and an algorithm that achieves the bound. These works only focused on reducing the number of iterations.

Communication cost reduction. There are two general approaches for reducing the communication cost. One approach is to reduce the amount of data per communication through model compression, e.g., [199–201]. The other approach is to reduce the number of communications, e.g., by reducing the communication frequency [202–204]. It was shown that model compression and infrequent communications can be combined to further improve the communication efficiency [209, 210]. Instead of either activating all the links or activating none, it has been recognized that better tradeoffs can be achieved by activating subsets of links. To this end, [209, 210] proposed an event-triggered mechanism where a node sends its local model to neighbors only if the model has changed sufficiently, and [205, 206, 211] proposed to activate subsets of links with predetermined probabilities. In this regard, our work designs predetermined link activation as in [205, 206, 211], which provides more predictable performance than event-triggered mechanisms, but *we consider a cost model that is more practical in overlay-based DFL*: instead of measuring the communication time by the number of matchings as in [205, 206, 211] or the maximum degree as in [212, 213], we use the minimum time to complete all the activated agent-to-agent communications over a bandwidth-limited underlay, while taking into account heterogeneous residual capacities and possibly shared links.

Topology design in DFL. The logical topology connecting learning agents is an important design parameter in DFL that controls the communication demands during training, as only neighboring agents will communicate. Much has been done on characterizing the impact of this topology on the convergence rate of DFL. Most convergence analysis captures this impact through the spectral gap of the mixing matrix [197, 224–227] or equivalent parameters [205, 211]. Recent works have identified other parameters through which the topology can impact the convergence rate, such as the effective number of neighbors [228] and the neighborhood heterogeneity [213]. Yet, these results did not invalidate the impact of spectral gap; they just pointed out additional factors that also matter in some cases. Based on the convergence parameters identified in such analysis, several solutions have been proposed to design the logical topology to balance the convergence rate and the cost per communication round [205, 206, 211, 213], and some solutions combined topology design with other optimizations (e.g., bandwidth allocation [229], model pruning [227]) for further improvement. In this regard, part of our work also

addresses the topology design problem based on a parameter related to the spectral gap, but *we explicitly design the communication schedule to serve the demands triggered by the designed topology through a bandwidth-limited underlay* to optimize the wall-clock time of overlay-based DFL.

To our knowledge, the only existing work addressing topology design in overlay-based DFL is [214]. However, it assumed a special underlay where the paths connecting learning agents only share links at the first and the last hops, and the internal links are effectively not shared. While this model may suit high-bandwidth underlays such as inter-datacenter networks where dedicated capacity shares can be reserved for each overlay link, it fails to capture the impact of overlay topology on the time to complete the corresponding communications in a bandwidth-limited underlay, as addressed in our work.

Network-aware distributed computing. Broadly speaking, it was known that awareness to the state of the communication underlay is important for data-intensive distributed computing tasks [198]. Several works attempted to solve this problem for a black-box cloud network based on simple heuristics (e.g., clustering nodes based on pairwise performance metrics [198, 230]) or limiting assumptions about the network (e.g., multi-rooted tree [231]), and another work [232] proposed a white-box solution by asking the cloud provider to provide the required network information. In this regard, our work assumes a black-box underlay as in [198, 230, 231], but unlike the simple heuristics in these works, we leverage state-of-the-art techniques from network tomography to estimate the necessary parameters about the underlay with guaranteed accuracy.

Network tomography. Network tomography can provide critical information for overlay-based DFL by enabling the learning agents to infer the topology and parameters of the underlay from end-to-end measurements between themselves [112]. Many solutions for topology inference have been developed (see [123] and references therein), but most are based on the limiting assumption of tree-based routing. Recently, it was discovered that the existence of links shared by subsets of paths can be reliably detected under arbitrary routing [123–125]. While this information is not enough for identifying the underlay topology, it is useful for optimizing communications in the overlay, as the knowledge of shared links together with the ability to estimate available path capacities [126] allows the overlay to estimate the capacity region for communications between the overlay nodes [221]. In this work, we leverage this capability to optimize how the learning agents communicate over an uncooperative underlay.

In summary, we proposed to jointly design the communication demands and the communication schedule for overlay-based DFL over a bandwidth-limited uncooperative

underlay.

6.2.1 Background and Problem Formulation

6.2.1.1 Notations

Let $\boldsymbol{a} \in \mathbb{R}^m$ denote a vector and $\boldsymbol{A} \in \mathbb{R}^{m \times m}$ a matrix. We use $\|\boldsymbol{a}\|$ to denote the ℓ-2 norm, $\|\boldsymbol{A}\|$ to denote the spectral norm, and $\|\boldsymbol{A}\|_F$ to denote the Frobenius norm. We use $\text{diag}(\boldsymbol{a})$ to denote a diagonal matrix with the entries in $\boldsymbol{a}$ on the main diagonal, and $\text{diag}(\boldsymbol{A})$ to denote a vector formed by the diagonal entries of $\boldsymbol{A}$. We use $\lambda_i(\boldsymbol{A})$ $(i = 1, \ldots, m)$ to denote the i-th smallest eigenvalue of $\boldsymbol{A}$.

6.2.1.2 Network Model

Consider a network of m learning agents connected through a logical *base topology* $G = (V, E)$ $(|V| = m)$, that forms an overlay on top of a communication underlay $\underline{G} = (\underline{V}, \underline{E})$. Unless otherwise stated, both overlay and underlay links are considered directed. Each underlay link $\underline{e} \in \underline{E}$ has a finite capacity $C_{\underline{e}}$. Each overlay link $e = (i, j) \in E$ indicates that agent i is *allowed* to communicate to agent j during learning, and is implemented via a routing path $\underline{p}_{i,j}$ from the node running agent i to the node running agent j in the underlay. We assume that if $(i, j) \in E$, then $(j, i) \in E$ (agents i and j are allowed to exchange results). The routing paths are determined by the topology and the routing protocol in the underlay. Let $l_{i,j}$ denote the propagation delay on $\underline{p}_{i,j}$. We assume that neither the routing paths nor the link capacities in the underlay are observable by the overlay, but the propagation delays between overlay nodes (e.g., $l_{i,j}$) are observable[1].

6.2.1.3 Decentralized Federated Learning (DFL)

Consider a DFL task, where each agent $i \in V$ has a possibly non-convex objective function $F_i(\boldsymbol{x})$ that depends on the parameter vector $\boldsymbol{x} \in \mathbb{R}^d$ and the local dataset $\mathcal{D}_i$, and the goal is to find the parameter vector $\boldsymbol{x}$ that minimizes the global objective function $F(\boldsymbol{x})$, defined as

$$F(\boldsymbol{x}) := \frac{1}{m} \sum_{i=1}^{m} F_i(\boldsymbol{x}). \tag{6.1}$$

[1]This can be obtained by measuring the delays of small probing packets.

For example, we can model the objective of empirical risk minimization by defining the local objective as $F_i(\boldsymbol{x}) := \sum_{\boldsymbol{s}\in\mathcal{D}_i} \ell(\boldsymbol{x}, \boldsymbol{s})$, where $\ell(\boldsymbol{x}, \boldsymbol{s})$ is the loss function for sample $\boldsymbol{s}$ under model $\boldsymbol{x}$, and the corresponding global objective is proportional to the empirical risk over all the samples.

We consider a standard decentralized training algorithm called D-PSGD [197], where each agent repeatedly updates its own parameter vector and aggregates it with the parameter vectors of its neighbors to minimize the global objective function. Specifically, let $\boldsymbol{x}_i^{(k)}$ $(k \geq 1)$ denote the parameter vector at agent i after $k-1$ iterations and $g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)})$ the stochastic gradient computed by agent i in iteration k (where $\xi_i^{(k)}$ is the mini-batch). In iteration k, agent i updates its parameter vector by

$$\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}^{(k)} \boldsymbol{x}_j^{(k)} - \eta g(\boldsymbol{x}_i^{(k)}; \xi_i^{(k)}), \tag{6.2}$$

where $\boldsymbol{W}^{(k)} = (W_{ij}^{(k)})_{i,j=1}^{m}$ is the $m \times m$ *mixing matrix* in iteration k, and $\eta > 0$ is the learning rate. To be consistent with the base topology, $W_{ij}^{(k)} \neq 0$ only if $(i, j) \in E$. The update rule in (6.2) has the same convergence performance as $\boldsymbol{x}_i^{(k+1)} = \sum_{j=1}^{m} W_{ij}^{(k)}(\boldsymbol{x}_j^{(k)} - \eta g(\boldsymbol{x}_j^{(k)}; \xi_j^{(k)}))$ [197, 205], but (6.2) allows each agent to parallelize the parameter exchange with neighbors and the gradient computation.

The mixing matrix $\boldsymbol{W}^{(k)}$ plays an important role in controlling the communication cost, as agent j needs to send its parameter vector to agent i in iteration k if and only if $W_{ij}^{(k)} \neq 0$. According to [197], the mixing matrix should be *symmetric with each row/column summing up to one*[2] in order to ensure convergence for D-PSGD. The symmetry implies a one-one correspondence between distinct (possibly) non-zero entries in $\boldsymbol{W}^{(k)}$ and the *undirected overlay links*, denoted by $\tilde{E}$ (i.e., each $(i, j) \in \tilde{E}$ represents a pair of directed links $\{(i, j), (j, i)\} \in E$), and thus $W_{ij}^{(k)}$ can be interpreted as the *link weight* of the undirected overlay link $(i, j) \in \tilde{E}$. The requirement of each row summing to one further implies that $W_{ii}^{(k)} = 1 - \sum_{j=1}^{m} W_{ij}^{(k)}$. In the vector form, the above implies the following decomposition of the mixing matrix

$$\boldsymbol{W}^{(k)} := \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}^{(k)})\boldsymbol{B}^{\top}, \tag{6.3}$$

[2]In [197], the mixing matrix was assumed to be symmetric and *doubly stochastic* with entries constrained to $[0, 1]$, but we find this requirement unnecessary for the convergence bound we use from [233, Theorem 2], which only requires the mixing matrix to be symmetric with each row/column summing up to one.

where $\boldsymbol{I}$ is the $m \times m$ identity matrix, $\boldsymbol{B}$ is the $|V| \times |\tilde{E}|$ incidence matrix[3] for the base topology G, and $\boldsymbol{\alpha}^{(k)} := (\alpha_{ij}^{(k)})_{(i,j)\in\tilde{E}}$ is the vector of link weights. It is easy to verify that $W_{ij}^{(k)} = \alpha_{ij}^{(k)}$. This decomposition reduces the design of mixing matrix to the design of link weights $\boldsymbol{\alpha}^{(k)}$ in the overlay, where agents i and j need to exchange parameter vectors in iteration k *if and only if* $\alpha_{ij}^{(k)} \neq 0$. Thus, we say that *the (undirected) overlay link* (i, j) *is activated* in iteration k (i.e., both (i, j) and (j, i) are activated) if $\alpha_{ij}^{(k)} \neq 0$.

6.2.1.4 Communication Optimization for Overlay-based DFL

Our goal is to *jointly design the communication demands between the agents and the communication schedule about how to service these demands* so as to minimize the total (wall-clock) time for the learning task to reach a given level of convergence. The challenges are two-fold: (i) the design of communication demands faces the tradeoff between communicating more per iteration and converging in fewer iterations versus communicating less per iteration and converging in more iterations, and (ii) the design of communication schedule faces the lack of observability and controllability within the underlay network. Below, we will tackle these challenges by combining techniques from network tomography and mixing matrix design.

6.2.2 Proposed Solution

Our approach is to first characterize the total training time as an explicit function of the set of activated links in the overlay, and then optimize this set. We will focus on a deterministic design that can give a predictable training time, and thus the iteration index k will be omitted. For ease of presentation, we will consider the set of activated overlay links, denoted by $E_a \subseteq \tilde{E}$, as *undirected links*, as the pair of links between two agents must be activated at the same time.

6.2.2.1 Overlay-based Communication Schedule Optimization

Given a set of overlay links $E_a \subseteq \tilde{E}$ activated in an iteration, each $(i, j) \in E_a$ triggers two communications, one for agent i to send its parameter vector to agent j and the other for agent j to send its parameter vector to agent i. However, directly sending the parameter vectors along the underlay routing paths can lead to suboptimal performance. For example, consider Fig. 6.2. If $E_a = \{(A, B), (A, D)\}$ but both $\underline{p}_{A,B}$ and $\underline{p}_{A,D}$ traverse

[3]This is defined under an arbitrary orientation of each link $e_j \in \tilde{E}$ as $B_{ij} = +1$ if e_j starts from i, -1 if e_j ends at i, and 0 otherwise.

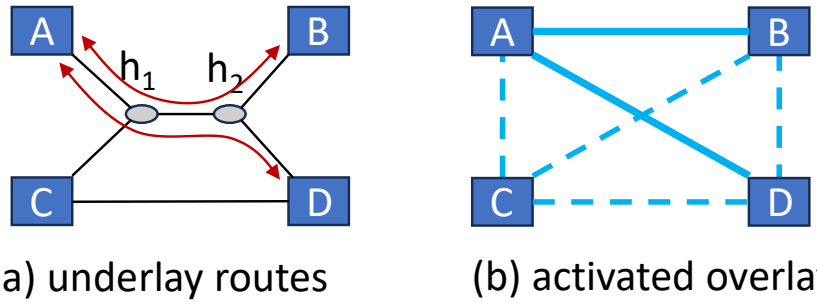


Figure 6.2. Underlay-aware communication schedule optimization (learning agents: $\{A, B, C, D\}$; underlay nodes: $\{h_1, h_2\}$).

the same underlay link (h_1, h_2), directly communicating between the activated agent pairs can take longer than redirecting part of the traffic through other agents (e.g., redirecting $A \to D$ traffic through the overlay path $A \to C \to D$). The same holds if the capacity of the direct path is low, but the capacity through other agents is higher (e.g., if (h_2, D) is a slow link, then redirecting $A \to D$ traffic through C can bypass it to achieve a higher rate). This observation motivates the need of optimizing how to serve the demands triggered by the activated links by routing within the overlay.

Demand Model: Let κ_i denote the size of the parameter vector (or its compressed version if model compression is used) at agent i. A straightforward way to model the communication demands triggered by a set of activated links E_a is to generate two unicast flows for each activated link $(i, j) \in E_a$, one in each direction. However, this model will lead to a suboptimal communication schedule as it ignores the fact that some flows carry identical content. Specifically, all flows originating from the same agent will carry the latest parameter vector at this agent. Thus, the actual communication demands is a set of multicast flows, each for distributing the parameter vector of an activated agent (incident to at least one activated link) to the agents it needs to share parameters with. Let $N_{E_a}(i) := \{j \in V : (i, j) \in E_a\}$. We can express the demands triggered by the activated links E_a as

$$H = \{(i, N_{E_a}(i), \kappa_i) : \forall i \in V \text{ with } N_{E_a}(i) \neq \emptyset\}, \tag{6.4}$$

where each $h = (s_h, T_h, \kappa_h) \in H$ represents a multicast flow with source s_h, destinations T_h, and data size κ_h.

Baseline Formulation: To help towards minimizing the total training time, the communication schedule should minimize the time for completing all the communication demands triggered by the activated links, within the control of the overlay. To this

end, we *jointly optimize the routing and the flow rate within the overlay.* The former is represented by decision variables $z_{ij}^h \in \{0,1\}$ that indicates whether overlay link (i,j) is traversed by the multicast flow h and $r_{ij}^{h,k} \in \{0,1\}$ that indicates whether (i,j) is traversed by the flow from s_h to $k \in T_h$, both in the direction of $i \to j$. The latter is represented by decision variables $d_h \geq 0$ that denotes the rate of flow h and $f_{ij}^h \geq 0$ that denotes the rate of flow h on overlay link (i,j) in the direction of $i \to j$. Define constant $b_i^{h,k}$ as 1 if $i = s_h$, -1 if $i = k$, and 0 otherwise. We can formulate the objective of serving all the multicast flows in H (6.4) within the minimum amount of time as the following optimization:

$$\min_{\boldsymbol{z},\boldsymbol{r},\boldsymbol{d},\boldsymbol{f}} \quad \tau \tag{6.5a}$$

$$\text{s.t.} \quad \tau \geq \frac{\kappa_h}{d_h} + \sum_{(i,j)\in E} l_{i,j} r_{ij}^{h,k}, \quad \forall h \in H, k \in T_h, \tag{6.5b}$$

$$\sum_{(i,j)\in E:\underline{e}\in \underline{p}_{i,j}} \sum_{h\in H} f_{ij}^h \leq C_{\underline{e}}, \quad \forall \underline{e} \in \underline{E}, \tag{6.5c}$$

$$\sum_{j\in V} r_{ij}^{h,k} = \sum_{j\in V} r_{ji}^{h,k} + b_i^{h,k}, \quad \forall h \in H, k \in T_h, i \in V, \tag{6.5d}$$

$$r_{ij}^{h,k} \leq z_{ij}^h, \quad \forall h \in H, k \in T_h, (i,j) \in E, \tag{6.5e}$$

$$d_h - M(1 - z_{ij}^h) \leq f_{ij}^h \leq d_h, \ \forall h \in H, (i,j) \in E, \tag{6.5f}$$

$$f_{ij}^h \leq M z_{ij}^h, \ \forall h \in H, (i,j) \in E, \tag{6.5g}$$

$$r_{ij}^{h,k}, z_{ij}^h \in \{0,1\}, \ d_h \in [0,M], \ f_{ij}^h \geq 0, \quad \forall h \in H, k \in T_h, (i,j) \in E, \tag{6.5h}$$

where M is an upper bound on d_h ($\forall h \in H$). Constraint (6.5b) makes τ an upper bound on the completion time of the slowest flow; (6.5c) ensures that the total traffic rate imposed by the overlay on any underlay link is within its capacity; (6.5d)–(6.5e) are the *Steiner arborescence* constraints [234] that guarantee the set of overlay links with $z_{ij}^h = 1$ will form a Steiner arborescence (i.e., a directed Steiner tree) that is the union of paths from s_h to each $k \in T_h$ (where each path is formed by the links with $r_{ij}^{h,k} = 1$); (6.5f) implies that $f_{ij}^h = d_h$ if $z_{ij}^h = 1$ and (6.5g) together with (6.5h) implies that $f_{ij}^h = 0$ if $z_{ij}^h = 0$, which allows the capacity constraint to be formulated as a linear inequality (6.5c) instead of a bilinear inequality $\sum_{(i,j)\in E:\underline{e}\in \underline{p}_{i,j}} \sum_{h\in H} d_h z_{ij}^h \leq C_{\underline{e}}$. The optimal solution $(\boldsymbol{z}^*, \boldsymbol{r}^*, \boldsymbol{d}^*, \boldsymbol{f}^*)$ to (6.5) provides an overlay communication schedule that minimizes the communication time in a given iteration when the set of activated links is E_a.

Complexity: As $|H| \leq |V|$, the optimization (6.5) contains $O(|V|^2|E|)$ variables

(dominated by $\boldsymbol{r}$), and $O(|\underline{E}| + |V|^2(|V| + |E|))$ constraints. Since constraints (6.5c)–(6.5f) are linear and constraint (6.5b) is convex, the optimization (6.5) is a mixed integer convex programming (MICP) problem and thus can be solved by existing MICP solvers such as Pajarito [235] at a super-polynomial complexity or approximate MICP algorithms such as convex relaxation plus randomized rounding at a polynomial complexity.

Handling Uncooperative Underlay: When learning over an uncooperative underlay as considered in this work, the overlay cannot directly solve (6.5), because the capacity constraint (6.5c) requires the knowledge of the routing in the underlay and the capacities of the underlay links. In absence of such knowledge, we leverage a recent result from [221] to convert this constraint into an equivalent form that can be consistently estimated by the overlay. To this end, we introduce the following notion from [221], adapted to our problem setting.

Definition 6.2.1 ([221])**.** *A* ***category of underlay links*** Γ_F *for a set of overlay links* F *(*$F \subseteq E$*) is the set of underlay links traversed* by and only by *the underlay routing paths for the overlay links in* F *out of all the paths for* E*, i.e,*[4]

$$\Gamma_F \coloneqq \Big(\bigcap_{(i,j)\in F} \underline{p}_{i,j} \Big) \setminus \Big(\bigcup_{(i,j)\in E\setminus F} \underline{p}_{i,j} \Big). \tag{6.6}$$

The key observation is that since all the underlay links in the same category are traversed by the same set of overlay links, they must carry the same traffic load from the overlay. Therefore, we can reduce the per-link capacity constraint (6.5c) into the following *per-category capacity constraint*:

$$\sum_{(i,j)\in F} \sum_{h\in H} f_{ij}^h \le C_F, \quad \forall F \subseteq E \text{ with } \Gamma_F \neq \emptyset, \tag{6.7}$$

where $C_F \coloneqq \min_{\underline{e}\in\Gamma_F} C_{\underline{e}}$, referred to as the *category capacity*, is the minimum capacity of all the links in category Γ_F. The new constraint (6.7) is equivalent to the original constraint (6.5c), as an overlay communication schedule satisfies one of these constraints if and only if it satisfies the other. However, instead of requiring detailed internal information about the underlay (i.e., $(\underline{p}_{i,j})_{(i,j)\in E}$ and $(C_{\underline{e}})_{\underline{e}\in\underline{E}}$), constraint (6.7) only requires the knowledge of the *nonempty categories* and the corresponding *category capacities*.

Under the mild assumption that every underlay link introduces a nontrivial performance impact (e.g., non-zero loss/queueing probability), [221] provided an algorithm that

[4]Here $\underline{p}$ is interpreted as the set of underlay links traversed by path $\underline{p}$.

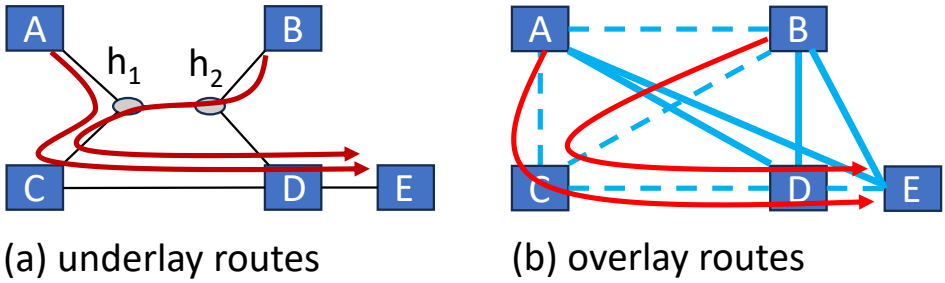


Figure 6.3. Challenge for in-overlay aggregation (learning agents: $\{A, B, C, D, E\}$; underlay nodes: $\{h_1, h_2\}$).

can consistently infer the nonempty categories from losses/delays of packets sent concurrently through the overlay links, under the assumption that concurrently sent packets will experience the same performance when traversing a shared underlay link. Moreover, by leveraging state-of-the-art single-path residual capacity estimation methods, [221] gave a simple algorithm that can accurately estimate the *effective category capacity* $\tilde{C}_F$ for each detected nonempty category, that can be used in place of C_F in (6.7) without changing the feasible region. Given the indices of inferred nonempty categories $\hat{\mathcal{F}}$ and their inferred effective capacities $(\hat{C}_F)_{F\in\hat{\mathcal{F}}}$, we can construct the per-category capacity constraint as

$$\sum_{(i,j)\in F}\sum_{h\in H} f_{ij}^h \le \hat{C}_F, \quad \forall F \in \hat{\mathcal{F}}, \tag{6.8}$$

which can then be used in place of (6.5c) in (6.5) to compute an optimized overlay communication schedule.

Remark: First, the traversal of overlay paths through the overlay links is directional, and the traversal of underlay routing paths through the underlay links is also directional. Correspondingly, the overlay links in a category index F should be treated as directed links (i.e., $(i,j) \in F$ only implies that the underlay links in Γ_F are traversed by the path $\underline{p}_{i,j}$). This is not to be confused with treating the activated links in E_a as undirected links, because each $(i,j) \in E_a$ stands for a parameter exchange between agents i and j. Moreover, we only use the activated links E_a to determine the flow demands H for the overlay, but any link $(i,j) \in E$ within the control of the overlay can be used in serving these flows.

Additional Optimization Opportunities and Challenges: The formulation (6.5) treats each overlay node that is neither the source nor one of the destinations of a multicast flow as a pure relay, but this node is actually a learning agent capable of aggregating the parameter vectors. This observation raises two questions: (i) Can an

agent include parameter vectors relayed through it in its own parameter aggregation? (ii) If an agent relays multiple parameter vectors for different sources, can it forward the aggregated vector instead of the individual vectors?

To answer the first question, consider the case in Fig. 6.2 when A sends its parameter vector $\boldsymbol{x}_A$ to D through the overlay path $A \to C \to D$. If C includes $\boldsymbol{x}_A$ in its own parameter aggregation with a non-zero weight W_{CA}, then by the symmetry of the mixing matrix, A must also include $\boldsymbol{x}_C$ in its parameter aggregation with weight $W_{AC} = W_{CA}$, which is equivalent to activating the overlay link (A, C). As we have left the optimization of the activated links E_a to another subproblem (Section 6.2.2.3), there is no need to include relayed parameter vectors in parameter aggregation when optimizing the communication schedule for a given set of activated links.

To answer the second question, consider the case in Fig. 6.3 when the overlay routes the multicast from A to $\{D, E\}$ (for disseminating $\boldsymbol{x}_A$) over $A \to C \to D \to E$, and the multicast from B to $\{D, E\}$ (for disseminating $\boldsymbol{x}_B$) over $B \to C \to D \to E$. Although instead of separately forwarding $\boldsymbol{x}_A$ and $\boldsymbol{x}_B$, C could aggregate them before forwarding, the aggregation will not save bandwidth for C, as D needs $W_{DA}\boldsymbol{x}_A + W_{DB}\boldsymbol{x}_B$ but E needs $W_{EA}\boldsymbol{x}_A + W_{EB}\boldsymbol{x}_B$, which are generally not the same. Another issue with in-network aggregation (within the overlay) is the synchronization delay introduced at the point of aggregation, and thus in-network aggregation may not reduce the completion time even when it can save bandwidth, e.g., at D. We thus choose not to consider in-network aggregation in our formulation (6.5). Further optimizations exploiting such capabilities are left to future work.

6.2.2.2 Conditional Link Weight Optimization

Given the set of activated links $E_a \subseteq \tilde{E}$, the communication time per iteration has been determined as explained in Section 6.2.2.1, but the number of iterations has not, and is heavily affected by the weights of the activated links. This leads to the question of how to minimize the number of iterations for achieving a desired level of convergence, under the constraint that only the activated links can have non-zero weights.

To answer this question, we leverage a state-of-the-art convergence bound for D-PSGD under the following assumptions:

(1) Each local objective function $F_i(\boldsymbol{x})$ is l-Lipschitz smooth, i.e., $\|\nabla F_i(\boldsymbol{x}) - \nabla F_i(\boldsymbol{x}')\| \leq l\|\boldsymbol{x} - \boldsymbol{x}'\|$, $\forall i \in V$.

(2) There exist constants $M_1, \hat{\sigma}$ such that $\frac{1}{m}\sum_{i \in V} \mathbf{E}[\|g(\boldsymbol{x}_i; \xi_i) - \nabla F_i(\boldsymbol{x}_i)\|^2] \leq \hat{\sigma}^2 +$

$\frac{M_1}{m}\sum_{i\in V}\|\nabla F(\boldsymbol{x}_i)\|^2$, $\forall \boldsymbol{x}_1,\ldots,\boldsymbol{x}_m\in\mathbb{R}^d$.

(3) There exist constants $M_2,\hat{\zeta}$ such that $\frac{1}{m}\sum_{i\in V}\|\nabla F_i(\boldsymbol{x})\|^2\leq\hat{\zeta}^2+M_2\|\nabla F(\boldsymbol{x})\|^2,\forall\boldsymbol{x}\in\mathbb{R}^d$.

Let $\boldsymbol{J}:=\frac{1}{m}\mathbf{1}\mathbf{1}^\top$ denote an ideal $m\times m$ mixing matrix with all entries being $\frac{1}{m}$.

Theorem 6.2.1. *[233, Theorem 2] Under assumptions (1)–(3), if there exist constants $p\in(0,1]$ and integer $t\geq 1$ such that the mixing matrices $\{\boldsymbol{W}^{(k)}\}_{k=1}^K$, each being symmetric with each row/column summing to one*[5]*, satisfy*

$$\mathbf{E}[\|\boldsymbol{X}\prod_{k=k't+1}^{(k'+1)t}\boldsymbol{W}^{(k)}-\boldsymbol{X}\boldsymbol{J}\|_F^2]\leq(1-p)\|\boldsymbol{X}-\boldsymbol{X}\boldsymbol{J}\|_F^2 \tag{6.9}$$

for all $\boldsymbol{X}:=[\boldsymbol{x}_1,\ldots,\boldsymbol{x}_m]$ and integer $k'\geq 0$, then D-PSGD can achieve $\frac{1}{K}\sum_{k=1}^K\mathbb{E}[\|\nabla F(\overline{\boldsymbol{x}}^k)\|^2]\leq\epsilon_0$ for any given $\epsilon_0>0$ ($\overline{\boldsymbol{x}}^{(k)}:=\frac{1}{m}\sum_{i=1}^m\boldsymbol{x}_i^{(k)}$) when the number of iterations reaches

$$K(p,t):=l(F(\overline{\boldsymbol{x}}^{(1)})-F_{\inf})\cdot O\left(\frac{\hat{\sigma}^2}{m\epsilon_0^2}+\frac{\hat{\zeta}t\sqrt{M_1+1}+\hat{\sigma}\sqrt{pt}}{p\epsilon_0^{3/2}}+\frac{t\sqrt{(M_2+1)(M_1+1)}}{p\epsilon_0}\right), \tag{6.10}$$

where $\overline{\boldsymbol{x}}^{(1)}$ is the initial parameter vector, and $F_{\inf}$ is a lower bound on $F(\cdot)$.

Remark: While there exist other convergence bounds for D-PSGD such as [205, 213, 226, 228], we choose Theorem 6.2.1 as the theoretical foundation of our design due to the generality of its assumptions. For example, assumption (2) generalizes the assumption of uniformly bounded variance of stochastic gradients in [205,213], assumption (3) generalizes the assumption of bounded data heterogeneity in [205, 226, 228], and assumptions (2-3) are easily implied by the bounded gradient assumption in [226] (see explanations in [233]).

For tractability, we will focus on the case of i.i.d. mixing matrices. In this case, to achieve ϵ_0-convergence, it suffices for the number of iterations to reach $K(p,t)$ as in (6.10) for $t=1$. We note that $K(p,1)$ depends on the mixing matrix only through the parameter p: the larger p, the smaller $K(p,1)$.

Recall that as explained in Section 6.2.1.3, the mixing matrix $\boldsymbol{W}$ is related to the link weights $\boldsymbol{\alpha}$ as $\boldsymbol{W}=\boldsymbol{I}-\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top$. To restrict the activated links to E_a, we set $\alpha_{ij}=0$ for all $(i,j)\notin E_a$. Below, we will show that the following optimization gives a

[5]Originally, [233, Theorem 2] had a stronger assumption that each mixing matrix is doubly stochastic, but we have verified that it suffices to have each row/column summing to one.

good design of the link weights:

$$\min_{\boldsymbol{\alpha}} \ \tilde{\rho} \tag{6.11a}$$

$$\text{s.t.} \ \ -\tilde{\rho}\boldsymbol{I} \preceq \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J} \preceq \tilde{\rho}\boldsymbol{I}, \tag{6.11b}$$

$$\alpha_{ij} = 0, \ \ \forall (i,j) \notin E_a. \tag{6.11c}$$

Corollary 6.2.1.1. *Under assumptions (1)–(3) and i.i.d. mixing matrices $\boldsymbol{W}^{(k)} \overset{d}{=} \boldsymbol{W}$ that is symmetric with each row/column summing to one, D-PSGD achieves ϵ_0-convergence as in Theorem 6.2.1 when the number of iterations reaches*

$$K(1 - \mathbf{E}[\|\boldsymbol{W} - \boldsymbol{J}\|^2], 1). \tag{6.12}$$

Moreover, conditioned on the set of activated links being E_a, (6.12) $\geq K(1-\tilde{\rho}^{*2}, 1)$, where $\tilde{\rho}^*$ is the optimal value of (6.11), with "=" achieved at $\boldsymbol{W}^* = \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}^*)\boldsymbol{B}^\top$ for $\boldsymbol{\alpha}^*$ being the optimal solution to (6.11).

Proof of Corollary 6.2.1.1. As $K(p,1)$ decreases with p, its minimum is achieved at the maximum value of p that satisfies (6.9) for $t=1$ and any value of $\boldsymbol{X}$, i.e.,

$$p := \min_{\boldsymbol{X} \neq \boldsymbol{0}} \left(1 - \frac{\mathbf{E}[\|\boldsymbol{X}(\boldsymbol{W} - \boldsymbol{J})\|_F^2]}{\|\boldsymbol{X}(\boldsymbol{I} - \boldsymbol{J})\|_F^2}\right). \tag{6.13}$$

By [236, Lemma 3.1], p defined in (6.13) satisfies $p = 1 - \rho$ for $\rho := \|\mathbf{E}[\boldsymbol{W}^\top \boldsymbol{W}] - \boldsymbol{J}\|$. By Jensen's inequality and the convexity of $\|\cdot\|$, $\rho \leq \mathbf{E}[\|\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{J}\|]$. For every realization of $\boldsymbol{W}$ that is symmetric with rows/columns summing to one, we have $\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{J} = (\boldsymbol{W} - \boldsymbol{J})^2$. Based on the eigendecomposition $\boldsymbol{W} - \boldsymbol{J} = \boldsymbol{Q}\,\mathrm{diag}(\lambda_1, \ldots, \lambda_m)\boldsymbol{Q}^\top$, we have

$$\|\boldsymbol{W}^\top \boldsymbol{W} - \boldsymbol{J}\| = \|\boldsymbol{Q}\,\mathrm{diag}(\lambda_1^2, \ldots, \lambda_m^2)\boldsymbol{Q}^\top\| = \max_{i=1,\ldots,m} \lambda_i^2 = \|\boldsymbol{W} - \boldsymbol{J}\|^2, \tag{6.14}$$

where we have used the fact that $\|\boldsymbol{W} - \boldsymbol{J}\| = \max_{i=1,\ldots,m} |\lambda_i|$. Thus, $K(p,1)$ for p defined in (6.13) is upper-bounded by $K(1 - \mathbf{E}[\|\boldsymbol{W} - \boldsymbol{J}\|^2], 1)$, which is a sufficient number of iterations for D-PSGD to achieve ϵ_0-convergence by Theorem 6.2.1.

The matrix inequality (6.11b) implies that $\tilde{\rho} \geq |\lambda_i|$ for all $i = 1, \ldots, m$, and thus the optimal value of (6.11) must satisfy $\tilde{\rho} = \max_{i=1,\ldots,m} |\lambda_i| = \|\boldsymbol{W} - \boldsymbol{J}\|$. Hence, the optimal value $\tilde{\rho}^*$ of (6.11) is the minimum value of $\|\boldsymbol{W} - \boldsymbol{J}\|$ for any realization of $\boldsymbol{W}$ that only activates the links in E_a. Therefore, $1 - \mathbf{E}[\|\boldsymbol{W} - \boldsymbol{J}\|^2] \leq 1 - \tilde{\rho}^{*2}$ and (6.12) $\geq K(1 - \tilde{\rho}^{*2}, 1)$, with "=" achieved at $\boldsymbol{W}^* = \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha}^*)\boldsymbol{B}^\top$. □

Corollary 6.2.1.1 implies that given the set of activated links, we can design the corresponding link weights by solving (6.11), which will minimize an upper bound (6.12) on the number of iterations to achieve ϵ_0-convergence. Optimization (6.11) is a semi-definite programming (SDP) problem that can be solved in polynomial time by existing algorithms [237].

Remark: When $\boldsymbol{W}$ satisfies the additional property of $\boldsymbol{I} \succeq \boldsymbol{W} \succeq -\boldsymbol{I}$, the largest singular value of $\boldsymbol{W}$ is 1 [212], and thus $\|\boldsymbol{W} - \boldsymbol{J}\|$ is the second largest singular value of $\boldsymbol{W}$. In this case, minimizing $\tilde{\rho}$ in (6.11) (which equals $\|\boldsymbol{W} - \boldsymbol{J}\|$ under the optimal solution) is equivalent to maximizing $\gamma(\boldsymbol{W}) := 1 - \|\boldsymbol{W} - \boldsymbol{J}\|$, which is the *spectral gap* of the mixing matrix $\boldsymbol{W}$ [226]. The spectral gap is the most widely-used parameter to capture the impact of the mixing matrix on the convergence rate [197, 224–227]. In this sense, our Corollary 6.2.1.1 extends the relationship between the spectral gap and the number of iterations to the case of random mixing matrices. As $\gamma(\boldsymbol{W}) \to 0$ (in probability), the number of iterations according to (6.12) grows at

$$K\left(1 - \mathbf{E}[(1-\gamma(\boldsymbol{W}))^2], 1\right) = O\left(\frac{1}{1 - \mathbf{E}[(1-\gamma(\boldsymbol{W}))^2]}\right) = O\left(\frac{1}{\mathbf{E}[\gamma(\boldsymbol{W})]}\right), \quad (6.15)$$

which is consistent with the existing result of $O(1/\gamma(\boldsymbol{W}))$ in the case of deterministic mixing matrix [225]. While other parameters affecting the convergence rate have been identified, e.g., the effective number of neighbors [228] and the neighborhood heterogeneity [213], these parameters are just additional factors instead of replacements of the spectral gap. We thus leave the optimization of these other objectives to future work.

6.2.2.3 Link Activation Optimization

Given how to optimize the communication schedule and the link weights for a given set E_a of activated links as explained in Sections 6.2.2.1–6.2.2.2, what remains is to optimize E_a itself, which affects both the communication demands (and hence the time per iteration) and the sparsity pattern of the mixing matrix (and hence the number of iterations required). As mentioned in Section 6.2.1.4, our goal is to minimize the total training time. For network-distributed learning, it is known that the training time is dominated by the communication time [198]. We thus model the total training time by

$$\tau(E_a) \cdot K(E_a), \quad (6.16)$$

where we have used $\tau(E_a)$ to denote the communication time per iteration according to (6.5), with (6.5c) replaced by (6.8), and $K(E_a)$ to denote the number of iterations $K(1-\tilde{\rho}^{*2},1)$ to achieve a given level of convergence according to Corollary 6.2.1.1. Our goal is to minimize (6.16) over all the candidate values of $E_a \subseteq \tilde{E}$.

Directly solving this optimization is intractable because the objective function (6.16) is not given explicitly. To address this challenge, we will relax $\tau(E_a)$ and $K(E_a)$ into upper bounds that are explicit functions of E_a.

Relaxed Objective Function: We first upper-bound $\tau(E_a)$ by considering a suboptimal but analyzable communication schedule. Consider a special solution to (6.5), where $z_{ij}^h = 1$ if $i = s_h, j \in T_h$ and 0 otherwise, and $r_{ij}^{h,k} = 1$ if $i = s_h, j = k$ and 0 otherwise, i.e., the parameter exchange triggered by each activated link $(i,j) \in E_a$ is performed directly along the underlay routing paths $\underline{p}_{i,j}$ and $\underline{p}_{j,i}$. To achieve a per-iteration communication time of $\overline{\tau}$, the rate d_i of sending the parameter vector of agent i to its activated neighbors must satisfy

$$d_i \geq \frac{\kappa_i}{\overline{\tau} - l_{i,j}}, \quad \forall j \text{ with } (i,j) \in E_a. \tag{6.17}$$

Let $l_i := \max_{(i,j)\in E} l_{i,j}$ denote the maximum propagation delay from agent i to its neighbors in the base topology. Then to satisfy (6.17), it suffices to have $d_i = \kappa_i/(\overline{\tau} - l_i)$. This communication schedule is feasible for (6.5) (with (6.5c) replaced by (6.8)) if and only if

$$\sum_{(i,j)\in F \cap E_a} \frac{\kappa_i}{\overline{\tau} - l_i} \leq \hat{C}_F, \quad \forall F \in \hat{\mathcal{F}}, \tag{6.18}$$

where "$F \cap E_a$" denotes the set of directed links in F for which the undirected counterparts belong to E_a. The minimum value of $\overline{\tau}$ satisfying (6.18), denoted by $\overline{\tau}(E_a)$, thus provides an upper bound on the minimum per-iteration time $\tau(E_a)$ under the activated links in E_a.

We then upper-bound $K(E_a)$ by upper-bounding the optimal value $\tilde{\rho}^*$ of (6.11). To this end, consider a specific solution to (6.11), where

$$\alpha_{ij} = \begin{cases} \frac{1}{\max(|N_{E_a}(i)|, |N_{E_a}(j)|)} & \text{if } (i,j) \in E_a, \\ 0 & \text{o.w.}, \end{cases} \tag{6.19}$$

for $N_{E_a}(i)$ (the set of activated neighbors of agent i) defined as in (6.4). Let $\boldsymbol{L}_{E_a} :=$

$\boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top$ for $\boldsymbol{\alpha}$ in (6.19). Under this solution, the objective value of (6.11) is

$$\tilde{\rho} = \|\boldsymbol{I} - \boldsymbol{L}_{E_a} - \boldsymbol{J}\| \tag{6.20}$$

$$= \max(1 - \lambda_2(\boldsymbol{L}_{E_a}),\ \lambda_m(\boldsymbol{L}_{E_a}) - 1), \tag{6.21}$$

where (6.20) is proved in the proof of Corollary 6.2.1.1, and (6.21) is by [206, Lemma IV.2] (here $\lambda_i(\boldsymbol{L}_{E_a})$ denotes the i-th smallest eigenvalue of $\boldsymbol{L}_{E_a}$). Since $K(1-\tilde{\rho}^{*2}, 1)$ is an increasing function of $\tilde{\rho}^*$, we have

$$K(E_a) \le K\left(1 - (\max(1 - \lambda_2(\boldsymbol{L}_{E_a}),\ \lambda_m(\boldsymbol{L}_{E_a}) - 1))^2, 1\right) =: \overline{K}(E_a). \tag{6.22}$$

Bi-level Optimization: Relaxing (6.16) into its upper bound $\overline{\tau}(E_a)\cdot\overline{K}(E_a)$ provides an objective function that can be easily evaluated for each candidate value of E_a without solving any optimization. However, we still face the exponentially large solution space of $E_a \subseteq \tilde{E}$. To address the complexity challenge, we decompose the relaxed optimization into a bi-level optimization as follows.

Lemma 6.2.1. *Let β be the maximum time per iteration. Then*

$$\min_{E_a \subseteq \tilde{E}} \overline{\tau}(E_a)\cdot\overline{K}(E_a) = \min_{\beta \ge 0} \beta \cdot \left(\min_{\overline{\tau}(E_a)\le\beta} \overline{K}(E_a)\right), \tag{6.23}$$

and the optimal solution E_a^* to the RHS of (6.23) *is also optimal for the LHS of* (6.23).

Proof of Lemma 6.2.1. Let (β^*, E_a^*) be the optimal solution to the RHS of (6.23), and E_a^o be the optimal solution to the LHS of (6.23). Let $\beta^o := \overline{\tau}(E_a^o)$. Then

$$\min_{\overline{\tau}(E_a)\le\beta^o} \overline{K}(E_a) \le \overline{K}(E_a^o) \tag{6.24}$$

$$\Rightarrow \beta^o \cdot \left(\min_{\overline{\tau}(E_a)\le\beta^o} \overline{K}(E_a)\right) \le \overline{\tau}(E_a^o)\cdot\overline{K}(E_a^o) \tag{6.25}$$

$$\Rightarrow \min_{\beta\ge 0} \beta \cdot \left(\min_{\overline{\tau}(E_a)\le\beta} \overline{K}(E_a)\right) \le \overline{\tau}(E_a^o)\cdot\overline{K}(E_a^o). \tag{6.26}$$

Meanwhile, β^* must equal $\overline{\tau}(E_a^*)$, as otherwise we can reduce β^* to further reduce the value of $\beta \cdot \left(\min_{\overline{\tau}(E_a)\le\beta} \overline{K}(E_a)\right)$, contradicting with the assumption that (β^*, E_a^*) is optimal. Therefore, by the definition of E_a^o,

$$\min_{\beta\ge 0} \beta \cdot \left(\min_{\overline{\tau}(E_a)\le\beta} \overline{K}(E_a)\right) = \overline{\tau}(E_a^*)\cdot\overline{K}(E_a^*) \ge \overline{\tau}(E_a^o)\cdot\overline{K}(E_a^o), \tag{6.27}$$

which together with (6.26) proves (6.23).

Moreover, (6.23) implies that "=" must hold for (6.27), i.e., E_a^* is also optimal for the LHS of (6.23). □

The bi-level decomposition in (6.23) allows us to focus on the constrained optimization

$$\min_{\overline{\tau}(E_a)\leq\beta} \overline{K}(E_a), \tag{6.28}$$

as the upper-level optimization only has one scalar variable β that can be optimized numerically once we have an efficient solution to (6.28).

6.2.3 Future Directions

In this section, we summarize the possible approaches for solving (6.16), which will be studied in the future.

6.2.3.1 Lower-Level Optimization Approach

The first approach focuses on solving (6.28). The feasible space of (6.28) has a special structure. By (6.18), we see that E_a is feasible for (6.28) if and only if

$$\sum_{(i,j)\in F\cap E_a} \frac{\kappa_i}{\beta - l_i} \leq \hat{C}_F, \quad \forall F \in \hat{\mathcal{F}}. \tag{6.29}$$

We further noticed that for the relaxed objective of $\min_{E_a\subseteq\tilde{E}} \overline{\tau}(E_a)\cdot K(E_a)$, we have

$$\min_{\overline{\tau}(E_a)\leq\beta} K(E_a) \Leftrightarrow \min_{\overline{\tau}(E_a)\leq\beta} \rho(E_a), \tag{6.30}$$

where $\rho(E_a)$ denotes the optimal objective value of (6.11) when the set of activated links is E_a.

Let $y_{ij}\in\{0,1\}$ $(\forall (i,j)\in\tilde{E})$ indicate whether link (i,j) is activated. We can integrate (6.30) and (6.11) into a joint optimization of whether each link will be activated ($\boldsymbol{y}$) and its weight if activated ($\boldsymbol{\alpha}$):

$$\min_{\boldsymbol{y},\boldsymbol{\alpha}} \quad \tilde{\rho} \tag{6.31a}$$

$$\text{s.t.} \sum_{(i,j)\in F} y_{ij}\cdot\frac{\kappa_i}{\beta - l_{ij}} \leq \hat{C}_F, \quad \forall F\in\hat{\mathcal{F}}, \tag{6.31b}$$

$$-\tilde{\rho}\boldsymbol{I} \preceq \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J} \preceq \tilde{\rho}\boldsymbol{I}, \tag{6.31c}$$

$$0 \leq \alpha_{ij} \leq y_{ij}, \quad \forall (i,j) \in \tilde{E}, \tag{6.31d}$$

$$y_{ij} \in \{0,1\}, \quad \forall (i,j) \in \tilde{E}, \tag{6.31e}$$

where (6.31a), (6.31c), and (6.31d) are inherited from (6.11), (6.31b) enforces the constraint of (6.30) as explained in (6.29), and (6.31e) imposes the integer constraint on $\boldsymbol{y}$. Optimization (6.31) is again a MICP and thus can be solved by existing MICP solvers [235].

In summary, this approach is transformed into developing algorithms to solve (6.31).

6.2.3.2 Monolithic Approach

Joint minimization of τ and $\tilde{\rho}$ can be formulated as an monolithic optimization problem. More specifically, we can formulate a monolithic optimization as (6.32). Thus, we can directly invoke solvers to solve this problem.

$$\min_{\boldsymbol{f},\boldsymbol{d},\boldsymbol{z},\tau} \ \tau + \lambda_\rho \tilde{\rho} \tag{6.32a}$$

$$\text{s.t. } \tau \geq \frac{k}{d_e} + l_e - M(1 - z_e), \forall e \in E \tag{6.32b}$$

$$d_e - M(1 - z_e) \leq f_e \leq d_e + M(1 - z_e), \forall e \in E \tag{6.32c}$$

$$0 \leq f_e \leq M z_e, \forall e \in E \tag{6.32d}$$

$$\sum_{e \in F} f_e \leq C_F, \forall F \in \mathcal{F}, \tag{6.32e}$$

$$-\tilde{\rho}\boldsymbol{I} \preceq \boldsymbol{I} - \boldsymbol{B}\,\mathrm{diag}(\boldsymbol{\alpha})\boldsymbol{B}^\top - \boldsymbol{J} \preceq \tilde{\rho}\boldsymbol{I}, \tag{6.32f}$$

$$0 \leq \alpha_e \leq M z_e, \quad \forall e \ \in E, \tag{6.32g}$$

$$z_e \in \{0,1\}, \forall e \in E. \tag{6.32h}$$

In summary, this approach is transformed into developing algorithms to solve (6.32).

6.3 Concluding Remarks

In this dissertation, we studied the secure and efficient operation of cyber-physical systems (CPSs) through the lens of estimation, optimization and control. Under this context, we studied several problems in both physical and cyber spaces as well as their interactions.

In Chapter 2, we studied the line state estimation problem for post-attack recovery from the joint cyber-physical attacks in the smart grid. Under DC power flow model, we first extended the existing results and algorithms to the unexplored cases that the

attack may disconnect the power grid. Then, we developed the first known verification algorithms to verify the line state estimation results in the granularity of each single line. Finally, both estimation and verification algorithms are extended for AC power flow model. Evaluations over Polish grid and IEEE 300-bus system demonstrate that most of the lines can be correctly identified and verified.

In Chapter 3, we considered the optimal PMU placement problem to prevent the overload-induced line tripping. Under DC power flow model, we formulated a tri-level optimization problem and transformed it into a bi-level mixed integer linear programming (MILP). Then, we proposed a alternating optimization algorithm framework with two constraint generation algorithms to solve the proposed formulation optimally. Furthermore, for large power grid, we developed a polynomial-complexity heuristic to trade the solution quality for speed. Evaluation results over IEEE test systems showed that the proposed solution can significantly reduce the number of PMUs compared to the one to achieve full observability.

In Chapter 4, we revisited the overlay routing problem over an uncooperative underlay network. We first identified the minimum required information to achieve congestion-free overlay routing. Then, for general underlay networks, we developed polynomial-complexity algorithms to infer such information with theoretical guarantees. Furthermore, we proposed an additional algorithm for the special underlay network with symmetric tree-based routing. Extensive evaluations have demonstrated that the proposed approaches are capable of reducing the overlay routing delay by better avoiding congestion.

In Chapter 5, we investigated the optimal attack design problem for the cross-path attack, which is a particular DoS attack paradigm. We first developed novel reconnaissance algorithms to infer the locations and capacities of the shared network components between the attack paths and the target paths. Then, we formulated two attack budget allocation problems, for each of which we have demonstrated the optimal allocation policy. The proposed solution has been shown to cause severer performance degradation than its non-optimized counterparts in extensive evaluations, confirming the necessity of defending against such attacks during operation.

In Chapter 6, we examined the limitations of the research presented in this dissertation and proposed potential extensions. Following that, as an illustrative example, we investigated the joint optimization of overlay routing and the mixing matrix to minimize the total wall-clock time of decentralized learning over Cyber-Physical Systems (CPSs). We formulated this challenge as a bi-level optimization problem and outlined potential approaches for addressing it.

Bibliography

[1] Rehmani, M. H., A. Davy, B. Jennings, and C. Assi (2019) "Software defined networks-based smart grid communication: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, **21**(3), pp. 2637–2670.

[2] Czentye, J., J. Dóka, Á. Nagy, L. Toka, B. Sonkoly, and R. Szabó (2018) "Controlling drones from 5G networks," in *ACM SIGCOMM on Posters and Demos*, pp. 120–122.

[3] Liu, K., J. K. Ng, V. C. Lee, S. H. Son, and I. Stojmenovic (2015) "Cooperative data scheduling in hybrid vehicular ad hoc networks: VANET as a software defined network," *IEEE/ACM transactions on networking*, **24**(3), pp. 1759–1773.

[4] Aujla, G. S., R. Chaudhary, K. Kaur, S. Garg, N. Kumar, and R. Ranjan (2018) "SAFE: SDN-assisted framework for edge–cloud interplay in secure healthcare ecosystem," *IEEE Transactions on Industrial Informatics*, **15**(1), pp. 469–480.

[5] Sztipanovits, J., X. Koutsoukos, G. Karsai, N. Kottenstette, P. Antsaklis, V. Gupta, B. Goodwine, J. Baras, and S. Wang (2011) "Toward a science of cyber–physical system integration," *Proceedings of the IEEE*, **100**(1), pp. 29–44.

[6] Huang, H., H. Xu, Y. Cai, R. S. Khalid, and H. Yu (2018) "Distributed machine learning on smart-gateway network toward real-time smart-grid energy management with behavior cognition," *ACM Transactions on Design Automation of Electronic Systems*, **23**(5), pp. 1–26.

[7] Li, S., Q. Ni, Y. Sun, G. Min, and S. Al-Rubaye (2018) "Energy-efficient resource allocation for industrial cyber-physical IoT systems in 5G era," *IEEE Transactions on Industrial Informatics*, **14**(6), pp. 2618–2628.

[8] Liu, Y., Y. Peng, B. Wang, S. Yao, and Z. Liu (2017) "Review on cyber-physical systems," *IEEE/CAA Journal of Automatica Sinica*, **4**(1), pp. 27–40.

[9] Humayed, A., J. Lin, F. Li, and B. Luo (2017) "Cyber-physical systems security—A survey," *IEEE Internet of Things Journal*, **4**(6), pp. 1802–1831.

[10] Pasqualetti, F., F. Dörfler, and F. Bullo (2013) "Attack detection and identification in cyber-physical systems," *IEEE transactions on automatic control*, **58**(11), pp. 2715–2729.

[11] Dibaji, S. M., M. Pirani, D. B. Flamholz, A. M. Annaswamy, K. H. Johansson, and A. Chakrabortty (2019) "A systems and control perspective of CPS security," *Annual Reviews in Control*, **47**, pp. 394–411.

[12] Burg, A., A. Chattopadhyay, and K.-Y. Lam (2017) "Wireless communication and security issues for cyber–physical systems and the Internet-of-Things," *Proceedings of the IEEE*, **106**(1), pp. 38–60.

[13] Cao, J., Q. Li, R. Xie, K. Sun, G. Gu, M. Xu, and Y. Yang (2019) "The {CrossPath} Attack: Disrupting the {SDN} Control Channel via Shared Links," in *28th USENIX Security Symposium (USENIX Security 19)*, pp. 19–36.

[14] Andersen, D., H. Balakrishnan, F. Kaashoek, and R. Morris (2001) "Resilient overlay networks," in *SOSP*, pp. 131–145.

[15] Olowononi, F. O., D. B. Rawat, and C. Liu (2020) "Resilient machine learning for networked cyber physical systems: A survey for machine learning security to securing machine learning for CPS," *IEEE Communications Surveys & Tutorials*, **23**(1), pp. 524–552.

[16] Zhang, Q., F. Li, Q. Shi, K. Tomsovic, J. Sun, and L. Ren (2020) "Profit-Oriented False Data Injection on Energy Market: Reviews, Analyses and Insights," *IEEE Transactions on Industrial Informatics*.

[17] Che, L., X. Liu, Z. Li, and Y. Wen (2018) "False data injection attacks induced sequential outages in power systems," *IEEE Transactions on Power Systems*, **34**(2), pp. 1513–1523.

[18] (2016), "Analysis of the cyber attack on the Ukrainian power grid," `https://ics.sans.org/media/E-ISAC\_SANS\_Ukraine\_DUC\_5.pdf`.

[19] Soltan, S., M. Yannakakis, and G. Zussman (2018) "Power grid state estimation following a joint cyber and physical attack," *IEEE Transactions on Control of Network Systems*, **5**(1), pp. 499–512.

[20] Liu, Y., P. Ning, and M. K. Reiter (2011) "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security*, **14**(1), pp. 1–33.

[21] Li, Z., M. Shahidehpour, F. Aminifar, A. Alabdulwahab, and Y. Al-Turki (2017) "Networked microgrids for enhancing the power system resilience," *Proceedings of the IEEE*, **105**(7), pp. 1289–1310.

[22] Hossain, M. B., S. R. Pokhrel, and J. Choi (2023) "Smart Grid Meets URLLC: A Federated Orchestration With Improved Communication for Efficient Energy Resources Management," *IEEE Internet of Things Journal*.

[23] Fairley, P. (2016) "Cybersecurity at U.S. Utilities due for an Upgrade: Tech to Detect Intrusions into Industrial Control Systems will be Mandatory," *IEEE Spectrum*, **53**(5), pp. 11–13.

[24] Zhu, H. and G. B. Giannakis (2012) "Sparse Overcomplete Representations for Efficient Identification of Power Line Outages," *IEEE Transactions on Power Systems*, **27**(4), pp. 2215–2224.

[25] Chen, J.-C., W.-T. Li, C.-K. Wen, J.-H. Teng, and P. Ting (2014) "Efficient identification method for power line outages in the smart power grid," *IEEE Transactions on Power Systems*, **29**(4), pp. 1788–1800.

[26] Soltan, S., M. Yannakakis, and G. Zussman (2018) "REACT to cyber attacks on power grids," *IEEE Transactions on Network Science and Engineering*, **6**(3), pp. 459–473.

[27] Soltan, S., P. Mittal, and H. V. Poor (2019) "Line failure detection after a cyber-physical attack on the grid using Bayesian regression," *IEEE Transactions on Power Systems*, **34**(5), pp. 3758–3768.

[28] Tate, J. E. and T. J. Overbye (2008) "Line outage detection using phasor angle measurements," *IEEE Transactions on Power Systems*, **23**(4), pp. 1644–1652.

[29] ——— (2009) "Double line outage detection using phasor angle measurements," in *2009 IEEE Power & Energy Society General Meeting*, IEEE, pp. 1–5.

[30] Huang, Y.-F., S. Werner, J. Huang, N. Kashyap, and V. Gupta (2012) "State estimation in electric power grids: Meeting new challenges presented by the requirements of the future grid," *IEEE Signal Processing Magazine*, **29**(5), pp. 33–43.

[31] Monticelli, A. (2000) "Electric power system state estimation," *Proceedings of the IEEE*, **88**(2), pp. 262–282.

[32] Shoukry, Y., P. Nuzzo, A. Puggelli, A. L. Sangiovanni-Vincentelli, S. A. Seshia, and P. Tabuada (2017) "Secure state estimation for cyber-physical systems under sensor attacks: A satisfiability modulo theory approach," *IEEE Transactions on Automatic Control*, **62**(10), pp. 4917–4932.

[33] Deng, R., P. Zhuang, and H. Liang (2017) "CCPA: Coordinated cyber-physical attacks and countermeasures in smart grid," *IEEE Transactions on Smart Grid*, **8**(5), pp. 2420–2430.

[34] Vuković, O., K. C. Sou, G. Dán, and H. Sandberg (2011) "Network-layer protection schemes against stealth attacks on state estimators in power systems," in *2011 IEEE International Conference on Smart Grid Communications (SmartGridComm)*, IEEE, pp. 184–189.

[35] Kim, J. and L. Tong (2013) "On topology attack of a smart grid: Undetectable attacks and countermeasures," *IEEE Journal on Selected Areas in Communications*, **31**(7), pp. 1294–1305.

[36] Kaviani, R. and K. W. Hedman (2020) "A detection mechanism against load-redistribution attacks in smart grids," *IEEE Transactions on Smart Grid*.

[37] Garcia, M., T. Catanach, S. Vander Wiel, R. Bent, and E. Lawrence (2016) "Line outage localization using phasor measurement data in transient state," *IEEE Transactions on Power Systems*, **31**(4), pp. 3019–3027.

[38] Soltan, S. and G. Zussman (2017) "Power grid state estimation after a cyber-physical attack under the AC power flow model," in *2017 IEEE Power & Energy Society General Meeting*, IEEE, pp. 1–5.

[39] ——— (2018) "EXPOSE the line failures following a cyber-physical attack on the power grid," *IEEE Transactions on Control of Network Systems*, **6**(1), pp. 451–461.

[40] Chung, H.-M., W.-T. Li, C. Yuen, W.-H. Chung, Y. Zhang, and C.-K. Wen (2018) "Local cyber-physical attack for masking line outage and topology attack in smart grid," *IEEE Transactions on Smart Grid*, **10**(4), pp. 4577–4588.

[41] Jamei, M., R. Ramakrishna, T. Tesfay, R. Gentz, C. Roberts, A. Scaglione, and S. Peisert (2019) "Phasor Measurement Units Optimal Placement and Performance Limits for Fault Localization," *IEEE Journal on Selected Areas in Communications*, **38**(1), pp. 180–192.

[42] Yan, Y., Y. Qian, H. Sharif, and D. Tipper (2012) "A survey on smart grid communication infrastructures: Motivations, requirements and challenges," *IEEE communications surveys & tutorials*, **15**(1), pp. 5–20.

[43] Galli, S., A. Scaglione, and Z. Wang (2011) "For the grid and through the grid: The role of power line communications in the smart grid," *Proceedings of the IEEE*, **99**(6), pp. 998–1027.

[44] Liang, H., B. J. Choi, A. Abdrabou, W. Zhuang, and X. S. Shen (2012) "Decentralized economic dispatch in microgrids via heterogeneous wireless networks," *IEEE journal on Selected Areas in communications*, **30**(6), pp. 1061–1074.

[45] Association, I. S. et al. (2012), "IEEE 1815-2012–IEEE Standard for Electric Power Systems Communications-Distributed Network Protocol (DNP3)," .

[46] Coffrin, C. and P. Van Hentenryck (2015) "Transmission system restoration with co-optimization of repairs, load pickups, and generation dispatch," *International Journal of Electrical Power & Energy Systems*, **72**, pp. 144–154.

[47] Huang, Y., T. He, N. R. Chaudhuri, and T. L. Porta (2021), "Link State Estimation under Cyber-Physical Attacks: Theory and Algorithms," Technical Report, `https://sites.psu.edu/nsrg/files/2021/10/YudiStateRecoveryReport.pdf`.

[48] Bi, S. and Y. J. Zhang (2014) "Graphical methods for defense against false-data injection attacks on power system state estimation," *IEEE Transactions on Smart Grid*, **5**(3), pp. 1216–1227.

[49] Yang, Q., L. Jiang, W. Hao, B. Zhou, P. Yang, and Z. Lv (2017) "PMU placement in electric transmission networks for reliable state estimation against false data injection attacks," *IEEE Internet of Things Journal*, **4**(6), pp. 1978–1986.

[50] Yang, Q., D. An, R. Min, W. Yu, X. Yang, and W. Zhao (2017) "On optimal PMU placement-based defense against data integrity attacks in smart grid," *IEEE Transactions on Information Forensics and Security*, **12**(7), pp. 1735–1750.

[51] Kundur, P. (1994) *Power System Stability and Control, ser. The EPRI power system engineering series*, New York: McGraw-Hill.

[52] Lu, M., W. ZainalAbidin, T. Masri, D. Lee, and S. Chen (2016) "Under-Frequency Load Shedding (UFLS) Schemes-A Survey," *International Journal of Applied Engineering Research*, **11**(1), pp. 456–472.

[53] Terlaky, T. (2013) *Interior point methods of mathematical programming*, vol. 5, Springer Science & Business Media.

[54] Zimmerman, R. D. and C. E. Murillo-Sánchez (2019) "MATPOWER 7.0 user's manual," *Power Systems Engineering Research Center*, **9**.

[55] Dasgupta, S., C. H. Papadimitriou, and U. V. Vazirani (2008) *Algorithms*, McGraw-Hill Higher Education New York.

[56] Mangasarian, O. L. (1994) *Nonlinear programming*, SIAM.

[57] (2012), "Wide Area Monitoring, Protection, and Control Systems (WAMPAC) Standards for Cyber Security Requirements," National Electric Sector Cybersecurity Organization Resource (NESCOR), `https://smartgrid.epri.com/doc/ESRFSD.pdf`.

[58] Dagle, J. E. (2010) "The North American SynchroPhasor Initiative (NASPI)," in *IEEE PES General Meeting*, IEEE, pp. 1–3.

[59] (2014), "SynchroPhasor Technology Fact Sheet," North American SynchroPhasor Initiative, https://www.naspi.org/sites/default/files/reference_documents/33.pdf.

[60] (2015), "NASPI synchrophasor starter kit (draft)," North American SynchroPhasor Initiative (NASPI), https://www.naspi.org/sites/default/files/reference_documents/4.pdf.

[61] Jones, K. D., J. S. Thorp, and R. M. Gardner (2013) "Three-phase linear state estimation using phasor measurements," in *2013 IEEE Power & Energy Society General Meeting*, IEEE, pp. 1–5.

[62] Jones, K. D., A. Pal, and J. S. Thorp (2014) "Methodology for performing synchrophasor data conditioning and validation," *IEEE Transactions on Power Systems*, **30**(3), pp. 1121–1130.

[63] Chaojun, G., P. Jirutitijaroen, and M. Motani (2015) "Detecting false data injection attacks in AC state estimation," *IEEE Transactions on Smart Grid*, **6**(5), pp. 2476–2483.

[64] Bhatt, N., S. Sarawgi, R. O'keefe, P. Duggan, M. Koenig, M. Leschuk, S. Lee, K. Sun, V. Kolluri, S. Mandal, et al. (2009) "Assessing vulnerability to cascading outages," in *IEEE/PES Power Systems Conference and Exposition*, IEEE, pp. 1–9.

[65] Lachs, W. (1987) "Transmission-line overloads: real-time control," in *IEE Proceedings C (Generation, Transmission and Distribution)*, vol. 134, IET, pp. 342–347.

[66] Pavella, M., D. Ernst, and D. Ruiz-Vega (2012) *Transient stability of power systems: a unified approach to assessment and control*, Springer Science & Business Media.

[67] Sou, K. C. (2019) "Protection Placement for Power System State Estimation Measurement Data Integrity," *IEEE Transactions on Control of Network Systems*, **7**(2), pp. 638–647.

[68] Yuan, Y., Z. Li, and K. Ren (2011) "Modeling load redistribution attacks in power systems," *IEEE Transactions on Smart Grid*, **2**(2), pp. 382–390.

[69] Lakshminarayana, S., E. V. Belmega, and H. V. Poor (2021) "Moving-target defense against cyber-physical attacks in power grids via game theory," *IEEE Transactions on Smart Grid*.

[70] Liu, X., Z. Li, X. Liu, and Z. Li (2016) "Masking transmission line outages via false data injection attacks," *IEEE Transactions on Information Forensics and Security*, **11**(7), pp. 1592–1602.

[71] Li, Z., M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah (2015) "Bilevel model for analyzing coordinated cyber-physical attacks on power systems," *IEEE Transactions on Smart Grid*, **7**(5), pp. 2260–2272.

[72] Zhang, J. and L. Sankar (2016) "Physical system consequences of unobservable state-and-topology cyber-physical attacks," *IEEE Transactions on Smart Grid*, **7**(4).

[73] Li, Z., M. Shahidehpour, A. Alabdulwahab, and A. Abusorrah (2016) "Analyzing locally coordinated cyber-physical attacks for undetectable line outages," *IEEE Transactions on Smart Grid*, **9**(1), pp. 35–47.

[74] Tian, M., M. Cui, Z. Dong, X. Wang, S. Yin, and L. Zhao (2019) "Multilevel programming-based coordinated cyber physical attacks and countermeasures in smart grid," *IEEE Access*, **7**, pp. 9836–9847.

[75] Dan, G. and H. Sandberg (2010) "Stealth attacks and protection schemes for state estimators in power systems," in *IEEE SmartGridComm*, IEEE, pp. 214–219.

[76] Deng, R., G. Xiao, and R. Lu (2015) "Defending against false data injection attacks on power system state estimation," *IEEE Transactions on Industrial Informatics*, **13**(1), pp. 198–207.

[77] Liu, X., Z. Li, and Z. Li (2016) "Optimal protection strategy against false data injection attacks in power systems," *IEEE Transactions on Smart Grid*, **8**(4), pp. 1802–1810.

[78] Kim, T. T. and H. V. Poor (2011) "Strategic protection against data injection attacks on power grids," *IEEE Transactions on Smart Grid*, **2**(2), pp. 326–333.

[79] Li, Q., T. Cui, Y. Weng, R. Negi, F. Franchetti, and M. D. Ilic (2012) "An information-theoretic approach to PMU placement in electric power systems," *IEEE Transactions on Smart Grid*, **4**(1), pp. 446–456.

[80] Kosut, O., L. Jia, R. J. Thomas, and L. Tong (2011) "Malicious data attacks on the smart grid," *IEEE Transactions on Smart Grid*, **2**(4), pp. 645–658.

[81] Aminifar, F., A. Khodaei, M. Fotuhi-Firuzabad, and M. Shahidehpour (2009) "Contingency-constrained PMU placement in power networks," *IEEE Transactions on Power Systems*, **25**(1), pp. 516–523.

[82] Li, X., A. Scaglione, and T.-H. Chang (2013) "A framework for phasor measurement placement in hybrid state estimation via Gauss–Newton," *IEEE Transactions on Power Systems*, **29**(2), pp. 824–832.

[83] Wu, X. and A. J. Conejo (2016) "An efficient tri-level optimization model for electric grid defense planning," *IEEE Transactions on Power Systems*, **32**(4), pp. 2984–2994.

[84] Xiang, Y. and L. Wang (2017) "A game-theoretic study of load redistribution attack and defense in power systems," *Electric Power Systems Research*, **151**, pp. 12–25.

[85] Yao, Y., T. Edmunds, D. Papageorgiou, and R. Alvarez (2007) "Trilevel optimization in power network defense," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, **37**(4), pp. 712–718.

[86] Yuan, W., J. Wang, F. Qiu, C. Chen, C. Kang, and B. Zeng (2016) "Robust optimization-based resilient distribution network planning against natural disasters," *IEEE Transactions on Smart Grid*, **7**(6), pp. 2817–2826.

[87] Krumpholz, G., K. Clements, and P. Davis (1980) "Power system observability: a practical algorithm using network topology," *IEEE Transactions on Power Apparatus and Systems*, **PAS99**(4), pp. 1534–1542.

[88] Huang, Y., T. He, N. R. Chaudhuri, and T. La Porta (2021) "Preventing Outages under Coordinated Cyber-Physical Attack with Secured PMUs," in *IEEE SmartGridComm*, IEEE, pp. 258–263.

[89] Brahma, S., R. Kavasseri, H. Cao, N. Chaudhuri, T. Alexopoulos, and Y. Cui (2016) "Real-time identification of dynamic events in power systems using PMU data, and potential applications—models, promises, and challenges," *IEEE transactions on Power Delivery*, **32**(1), pp. 294–301.

[90] Mahapatra, K., M. Ashour, N. R. Chaudhuri, and C. M. Lagoa (2019) "Malicious corruption resilience in PMU data and wide-area damping control," *IEEE Transactions on Smart Grid*, **11**(2), pp. 958–967.

[91] Fischetti, M., I. Ljubić, M. Monaci, and M. Sinnl (2017) "A new general-purpose algorithm for mixed-integer bilevel linear programs," *Operations Research*, **65**(6), pp. 1615–1637.

[92] Chu, Z., J. Zhang, O. Kosut, and L. Sankar (2020) "Vulnerability assessment of large-scale power systems to false data injection attacks," in *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, IEEE, pp. 1–6.

[93] Liang, J., L. Sankar, and O. Kosut (2015) "Vulnerability analysis and consequences of false data injection attack on power system state estimation," *IEEE Transactions on Power Systems*, **31**(5), pp. 3864–3872.

[94] Chu, Z., J. Zhang, O. Kosut, and L. Sankar (2021) "N-1 Reliability Makes It Difficult for False Data Injection Attacks to Cause Physical Consequences," *IEEE Transactions on Power Systems*.

[95] Yang, Z., K. Xie, J. Yu, H. Zhong, N. Zhang, and Q. Xia (2018) "A general formulation of linear power flow models: Basic theory and error analysis," *IEEE Transactions on Power Systems*, **34**(2), pp. 1315–1324.

[96] Coffrin, C., H. L. Hijazi, and P. Van Hentenryck (2015) "The QC relaxation: A theoretical and computational study on optimal power flow," *IEEE Transactions on Power Systems*, **31**(4), pp. 3008–3018.

[97] Babaeinejadsarookolaee, S., A. Birchfield, R. D. Christie, C. Coffrin, C. DeMarco, R. Diao, M. Ferris, S. Fliscounakis, S. Greene, R. Huang, et al. (2019) "The power grid library for benchmarking ac optimal power flow algorithms," *arXiv preprint arXiv:1908.02788*.

[98] Chakrabarti, S., E. Kyriakides, and D. G. Eliades (2008) "Placement of synchronized measurements for power system observability," *IEEE Transactions on power delivery*, **24**(1), pp. 12–19.

[99] Coffrin, C., H. L. Hijazi, and P. Van Hentenryck (2015) "DistFlow extensions for AC transmission systems," *arXiv preprint arXiv:1506.04773*.

[100] Vielma, J. P. (2015) "Mixed integer linear programming formulation techniques," *Siam Review*, **57**(1), pp. 3–57.

[101] Sitaraman, R. K., M. Kasbekar, W. Lichtenstein, and M. Jain (2014) "Overlay networks: An Akamai perspective," *Advanced Content Delivery, Streaming, and Cloud Services*, **51**(4), pp. 305–328.

[102] Chen, L., S. Liu, and B. Li (2021) "Optimizing Network Transfers for Data Analytic Jobs Across Geo-Distributed Datacenters," *IEEE Transactions on Parallel and Distributed Systems*, **33**(2), pp. 403–414.

[103] Holterbach, T., S. Vissicchio, A. Dainotti, and L. Vanbever (2017) "Swift: Predictive fast reroute," in *SIGCOMM*, pp. 460–473.

[104] Tootaghaj, D. Z., F. Ahmed, P. Sharma, and M. Yannakakis (2020) "Homa: An efficient topology and route management approach in SD-WAN overlays," in *IEEE INFOCOM*, pp. 2351–2360.

[105] Tran, H.-A., D. Tran, and A. Mellouk (2022) "State-Dependent Multi-Constraint Topology Configuration for Software-Defined Service Overlay Networks," *IEEE/ACM Transactions on Networking*.

[106] Zhang, Y., J. Jiang, K. Xu, X. Nie, M. J. Reed, H. Wang, G. Yao, M. Zhang, and K. Chen (2018) "BDS: a centralized near-optimal overlay network for inter-datacenter data replication," in *EuroSys*, pp. 1–14.

[107] Yang, Z., Y. Cui, X. Wang, Y. Liu, M. Li, S. Xiao, and C. Li (2019) "Cost-efficient scheduling of bulk transfers in inter-datacenter WANs," *IEEE/ACM Transactions on Networking*, **27**(5), pp. 1973–1986.

[108] Zheng, X., C. Cho, and Y. Xia (2008) "Optimal peer-to-peer technique for massive content distribution," in *IEEE INFOCOM*, IEEE, pp. 151–155.

[109] Zhu, Y. and B. Li (2008) "Overlay networks with linear capacity constraints," *IEEE Transactions on Parallel and Distributed systems*, **19**(2), pp. 159–173.

[110] Zhu, Y., B. Li, and K. Q. Pu (2008) "Dynamic multicast in overlay networks with linear capacity constraints," *IEEE Transactions on Parallel and Distributed Systems*, **20**(7), pp. 925–939.

[111] Katabi, D. and C. Blake (2001) "Inferring congestion sharing and path characteristics from packet interarrival times," *MIT Report*.

[112] He, T., L. Ma, A. Swami, and D. Towsley (2021) *Network Tomography: Identifiability, Measurement Design, and Network State Inference*, Cambridge University Press.

[113] Ratnasamy, S. and S. McCanne (1999) "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *IEEE INFOCOM*, vol. 1, IEEE, pp. 353–360.

[114] Duffield, N. G., J. Horowitz, F. L. Presti, and D. Towsley (2002) "Multicast topology inference from measured end-to-end loss," *IEEE Transactions on Information Theory*, **48**(1), pp. 26–45.

[115] Duffield, N. G. and F. L. Presti (2004) "Network tomography from measured end-to-end delay covariance," *IEEE/ACM Transactions On Networking*, **12**(6), pp. 978–992.

[116] Ni, J., H. Xie, S. Tatikonda, and Y. R. Yang (2009) "Efficient and dynamic routing topology inference from end-to-end measurements," *IEEE/ACM transactions on networking*, **18**(1), pp. 123–135.

[117] Ni, J. and S. Tatikonda (2011) "Network tomography based on additive metrics," *IEEE Transactions on Information Theory*, **57**(12), pp. 7798–7809.

[118] Coates, M., R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang (2002) "Maximum likelihood network topology identification from edge-based unicast measurements," *ACM SIGMETRICS*, **30**(1), pp. 11–20.

[119] Rabbat, M. G., M. J. Coates, and R. D. Nowak (2006) "Multiple-source Internet tomography," *IEEE Journal on Selected Areas in Communications*, **24**(12), pp. 2221–2234.

[120] Sattari, P., M. Kurant, A. Anandkumar, A. Markopoulou, and M. G. Rabbat (2014) "Active learning of multiple source multiple destination topologies," *IEEE Transactions on Signal Processing*, **62**(8), pp. 1926–1937.

[121] Krishnamurthy, A. and A. Singh (2012) "Robust multi-source network tomography using selective probes," in *IEEE INFOCOM*, pp. 1629–1637.

[122] Berkolaiko, G., N. Duffield, M. Ettehad, and K. Manousakis (2018) "Graph reconstruction from path correlation data," *Inverse Problems*, **35**(1), p. 015001.

[123] Lin, Y., T. He, S. Wang, K. Chan, and S. Pasteris (2020) "Looking glass of NFV: Inferring the structure and state of NFV network from external observations," *IEEE/ACM Transactions on Networking*, **28**(4), pp. 1477–1490.

[124] Lin, Y., T. He, S. Wang, K. Chan, and S. Pasteris (2019) "Multicast-Based Weight Inference in General Network Topologies," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–6.

[125] Smith, K. D., S. Jafarpour, A. Swami, and F. Bullo (2022) "Topology Inference With Multivariate Cumulants: The Möbius Inference Algorithm," *IEEE/ACM Transactions on Networking*.

[126] Chaudhari, S. S. and R. C. Biradar (2015) "Survey of bandwidth estimation techniques in communication networks," *wireless personal communications*, **83**(2), pp. 1425–1476.

[127] Downey, A. B. (1999) "Using pathchar to estimate Internet link characteristics," *ACM SIGCOMM*, **29**(4), pp. 241–250.

[128] Dovrolis, C., P. Ramanathan, and D. Moore (2004) "Packet-dispersion techniques and a capacity-estimation methodology," *IEEE/ACM Transactions On Networking*, **12**(6), pp. 963–977.

[129] Liu, X., K. Ravindran, and D. Loguinov (2008) "A stochastic foundation of available bandwidth estimation: Multi-hop analysis," *IEEE/ACM Transactions on Networking*, **16**(1), pp. 130–143.

[130] Jain, M. and C. Dovrolis (2003) "End-to-end available bandwidth: measurement methodology, dynamics, and relation with TCP throughput," *IEEE/ACM Transactions on networking*, **11**(4), pp. 537–549.

[131] Hayes, D. A., S. Ferlin, and M. Welzl (2014) "Practical passive shared bottleneck detection using shape summary statistics," in *Annual IEEE Conference on Local Computer Networks*, IEEE, pp. 150–158.

[132] Hayes, D., S. Ferlin, M. Welzl, and K. Hiorth (2018) "Shared bottleneck detection for coupled congestion control for RTP media," *Internet Draft draft-ietf-rmcat-sbd-01*.

[133] Kanuparthy, P., C. Dovrolis, and M. Ammar (2008) "Spectral probing, crosstalk and frequency multiplexing in internet paths," in *IMC*, pp. 291–304.

[134] Chen, Y., D. Bindel, H. H. Song, and R. H. Katz (2007) "Algebra-based scalable overlay network monitoring: algorithms, evaluation, and applications," *IEEE/ACM Transactions on Networking*, **15**(5), pp. 1084–1097.

[135] Asano, Y. (2000) "Experimental Evaluation of Approximation Algorithms for the Minimum Cost Multiple-source Unsplittable Flow Problem." in *ICALP Satellite Workshops*, pp. 111–122.

[136] Barnhart, C., C. A. Hane, and P. H. Vance (2000) "Using branch-and-price-and-cut to solve origin-destination integer multicommodity flow problems," *Operations Research*, **48**(2), pp. 318–326.

[137] Ver Hoef, J. M. (2012) "Who invented the delta method?" *The American Statistician*, **66**(2), pp. 124–127.

[138] Knight, S., H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan (2011) "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, **29**(9), pp. 1765–1775.

[139] Gay, S., P. Schaus, and S. Vissicchio (2017) "Repetita: Repeatable experiments for performance evaluation of traffic-engineering algorithms," *arXiv preprint arXiv:1710.08665*.

[140] Jiang, H. and C. Dovrolis (2005) "Why is the internet traffic bursty in short time scales?" in *SIGMETRICS*, pp. 241–252.

[141] Svigelj, A., M. Mohorcic, G. Kandus, A. Kos, M. Pustisek, and J. Bester (2004) "Routing in ISL networks considering empirical IP traffic," *IEEE Journal on Selected areas in Communications*, **22**(2), pp. 261–272.

[142] Roughan, M. (2005) "Simplifying the synthesis of Internet traffic matrices," *ACM SIGCOMM*, **35**(5), pp. 93–96.

[143] Calinescu, G., C. Chekuri, M. Pál, and J. Vondrák (2011) "Maximizing a Monotone Submodular Function Subject to a Matroid Constraint," *SIAM Journal on Computing*, **40**(6), pp. 1740–1766.

[144] Shin, S., V. Yegneswaran, P. Porras, and G. Gu (2013) "Avant-guard: Scalable and vigilant switch flow management in software-defined networks," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 413–424.

[145] Wang, H., L. Xu, and G. Gu (2015) "FloodGuard: A DoS Attack Prevention Extension in Software-Defined Networks," in *45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, pp. 239–250.

[146] Shang, G., P. Zhe, X. Bin, H. Aiqun, and R. Kui (2017) "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks," in *IEEE INFOCOM*, pp. 1–9.

[147] Dhawan, M., R. Poddar, K. Mahajan, and V. Mann (2015) "SPHINX: detecting security attacks in software-defined networks." in *NDSS*, vol. 15, pp. 8–11.

[148] Zhang, S. (2019) "An overview of network slicing for 5G," *IEEE Wireless Communications*, **26**(3), pp. 111–117.

[149] Li, X., M. Samaka, H. A. Chan, D. Bhamare, L. Gupta, C. Guo, and R. Jain (2017) "Network slicing for 5G: Challenges and opportunities," *IEEE Internet Computing*, **21**(5), pp. 20–27.

[150] Kotulski, Z., T. W. Nowak, M. Sepczuk, M. Tunia, R. Artych, K. Bocianiak, T. Osko, and J.-P. Wary (2018) "Towards constructive approach to end-to-end slice isolation in 5G networks," *EURASIP Journal on Information Security*, **2018**(1), pp. 1–23.

[151] 3GPP (2016), "Feasibility study on new services and markets technology enablers for network operation; Stage 1," TR 22.864.
URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3016`

[152] Achleitner, S., T. La Porta, T. Jaeger, and P. McDaniel (2017) "Adversarial Network Forensics in Software Defined Networking," in *ACM Symposium on SDN Research (SOSR)*, ACM, pp. 8–20.

[153] Yu, M., T. Xie, T. He, P. McDaniel, and Q. K. Burke (2021) "Flow Table Security in SDN: Adversarial Reconnaissance and Intelligent Attacks," *IEEE/ACM Transactions on Networking*, **29**(6), pp. 2793–2806.

[154] Cunha, V. A., E. da Silva, M. B. de Carvalho, D. Corujo, J. P. Barraca, D. Gomes, L. Z. Granville, and R. L. Aguiar (2019) "Network slicing security: Challenges and directions," *Internet Technology Letters*, **2**(5), p. e125.

[155] Olimid, R. F. and G. Nencioni (2020) "5G network slicing: a security overview," *IEEE Access*, **8**, pp. 99999–100009.

[156] security group, N. G. (2016), "5G security recommendations Package 2: Network Slicing," NGMN Alliance.
URL `https://ngmn.org/wp-content/uploads/Publications/2016/160429_NGMN_5G_Security_Network_Slicing_v1_0.pdf`

[157] Liu, Q., T. Han, and N. Ansari (2020) "Learning-assisted secure end-to-end network slicing for cyber-physical systems," *IEEE Network*, **34**(3), pp. 37–43.

[158] Sattar, D. and A. Matrawy (2019) "Towards secure slicing: Using slice isolation to mitigate DDoS attacks on 5G core network slices," in *2019 IEEE Conference on Communications and Network Security (CNS)*, IEEE, pp. 82–90.

[159] Phillips, C. A. (1993) "The network inhibition problem," in *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing (STOC)*, pp. 776–785.

[160] Fu, X. and E. Modiano (2019) "Network Interdiction Using Adversarial Traffic Flows," in *IEEE INFOCOM*, pp. 1765–1773.

[161] Yao, H., S. Jaggi, and M. Chen (2012) "Passive Network Tomography for Erroneous Networks: A Network Coding Approach," *IEEE Transactions on Information Theory*, **58**(9), pp. 5922–5940.

[162] Lin, Y., T. He, and G. Pang (2021) "Queuing Network Topology Inference Using Passive Measurements," in *IFIP Networking*, IEEE, pp. 1–9.

[163] Ni, J., H. Xie, S. Tatikonda, and Y. R. Yang (2010) "Efficient and Dynamic Routing Topology Inference From End-to-End Measurements," *IEEE/ACM Transactions on Networking*, **18**(1), pp. 123–135.

[164] Coates, M., R. Castro, R. Nowak, M. Gadhiok, R. King, and Y. Tsang (2002) "Maximum likelihood network topology identification from edge-based unicast measurements," *ACM SIGMETRICS*, **30**(1), pp. 11–20.

[165] Salvat, J. X., L. Zanzi, A. Garcia-Saavedra, V. Sciancalepore, and X. Costa-Perez (2018) "Overbooking network slices through yield-driven end-to-end orchestration," in *CoNEXT*, pp. 353–365.

[166] Sattari, P., M. Kurant, A. Anandkumar, A. Markopoulou, and M. G. Rabbat (2014) "Active Learning of Multiple Source Multiple Destination Topologies," *IEEE Transactions on Signal Processing*, **62**(8), pp. 1926–1937.

[167] Duffield, N. G., J. Horowitz, and F. L. Prestis (2001) "Adaptive multicast topology inference," in *IEEE INFOCOM*, vol. 3, pp. 1636–1645.

[168] Rabbat, M., M. Coates, and R. Nowak (2006) "Multiple Source Internet Tomography," *IEEE Journal on Selected Areas in Communications*, **24**(12), pp. 2221–2234.

[169] Berkolaiko, G., N. Duffield, M. Ettehad, and K. Manousakis (2018) "Graph reconstruction from path correlation data," *Inverse Problems*, **35**(1), p. 015001.

[170] Harrison, P. and N. M. Patel (1992) *Performance Modelling of Communication Networks and Computer Architectures*, Addison–Wesley.

[171] Baccelli, F., B. Kauffmann, and D. Veitch (2009) "Inverse problems in queueing theory and Internet probing," *Queueing Systems*, **63**, p. 59.

[172] Cao, J., W. S. Cleveland, D. Lin, and D. X. Sun (2001) "On the Nonstationarity of Internet Traffic," *SIGMETRICS Performance Evaluation Review*, **29**(1), p. 102–112.

[173] Hoffman, K. L. (1981) "A METHOD FOR GLOBALLY MINIMIZING CONCAVE FUNCTIONS OVER CONVEX SETS," *Mathematical Programming*, **20**, pp. 22–32.

[174] Henderson, T. R., M. Lacage, G. F. Riley, C. Dowell, and J. Kopena (2008) "Network simulations with the ns-3 simulator," *SIGCOMM demonstration*, **14**(14), p. 527.

[175] Patriciello, N., S. Lagen, B. Bojovic, and L. Giupponi (2019) "An E2E simulator for 5G NR networks," *Simulation Modelling Practice and Theory*, **96**, p. 101933.

[176] Gvozdiev, N., S. Vissicchio, B. Karp, and M. Handley (2018) "On low-latency-capable topologies, and their impact on the design of intra-domain routing," in *SIGCOMM*, pp. 88–102.

[177] Alasmar, M., R. Clegg, N. Zakhleniuk, and G. Parisis (2021) "Internet traffic volumes are not Gaussian—They are log-normal: An 18-year longitudinal study with implications for modelling and prediction," *IEEE/ACM Transactions on Networking*, **29**(3), pp. 1266–1279.

[178] John, W. and S. Tafvelin (2007) "Analysis of internet backbone traffic and header anomalies observed," in *Proceedings of the 7th ACM SIGCOMM conference on Internet measurement*, pp. 111–116.

[179] Conn, A. R., K. Scheinberg, and L. N. Vicente (2009) *Introduction to Derivative-Free Optimization*, Society for Industrial and Applied Mathematics.

[180] 3GPP (2018), "NR; Study on integrated access and backhaul," TR 38.874.
URL `https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=3232`

[181] Ronkainen, H., J. Edstam, A. Ericsson, and C. Östberg (2020), "Integrated access and backhaul – a new type of wireless backhaul in 5G," Ericsson Technology Review.
URL `https://www.ericsson.com/4ac691/assets/local/reports-papers/ericsson-technology-review/docs/2020/introducing-integrated-access-and-backhaul.pdf`

[182] De Cicco, L., S. Mascolo, V. Palmisano, and G. Ribezzo (2019) "Reducing the network bandwidth requirements for 360° immersive video streaming," *Internet Technology Letters*, **2**(4), p. e118.

[183] (2017), "Preparing for a Cloud AR/VR Future," Huawei public white paper. URL `https://www-file.huawei.com/-/media/corporate/pdf/x-lab/cloud_vr_ar_white_paper_en.pdf`.

[184] Roughan, M. and C. Kalmanek (2003) "Pragmatic modeling of broadband access traffic," *Computer Communications*, **26**(8), pp. 804–816.

[185] Donoghue, W. F. (2014) *Distributions and Fourier transforms*, Academic Press.

[186] Sohn, K.-S., S. Y. Nam, and D. K. Sung (2006) "A distributed LSP scheme to reduce spare bandwidth demand in MPLS networks," *IEEE transactions on communications*, **54**(7), pp. 1277–1288.

[187] Yang, X. (2002) "Designing traffic profiles for bursty internet traffic," in *Global Telecommunications Conference, 2002. GLOBECOM'02. IEEE*, vol. 3, IEEE, pp. 2149–2154.

[188] Hasnat, M. A. and M. Rahnamay-Naeini (2019) "A data-driven dynamic state estimation for smart grids under DoS attack using state correlations," in *North American Power Symposium (NAPS)*, IEEE, pp. 1–6.

[189] Chen, B., S. Mashayekh, K. L. Butler-Purry, and D. Kundur (2013) "Impact of cyber attacks on transient stability of smart grids with voltage support devices," in *IEEE Power & Energy Society General Meeting*, IEEE, pp. 1–5.

[190] McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017) "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, PMLR, pp. 1273–1282.

[191] Liu, L., J. Zhang, S. Song, and K. B. Letaief (2023) "Hierarchical Federated Learning With Quantization: Convergence Analysis and System Design," *IEEE Transactions on Wireless Communications*, **22**(1), pp. 2–18.

[192] Kairouz, P. et al. (2021) *Advances and Open Problems in Federated Learning*, Now Foundations and Trends.

[193] "Google AI Blog: Federated Learning: Collaborative Machine Learning without Centralized Training Data," `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html`.

[194] "Google Assistant using federated learning on Android to improve 'Hey Google' accuracy," `https://9to5google.com/2021/03/26/google-assistant-hotword-federated-learning/`.

[195] "Federated Learning of Cohorts (FLoC)," `https://github.com/WICG/floc`.

[196] "Using Federated Learning to Improve Brave's On-Device Recommendations While Protecting Your Privacy," `https://brave.com/federated-learning/`.

[197] Lian, X., C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu (2017) "Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent," in *NeurIPS*, p. 5336–5346.

[198] Luo, L., P. West, J. Nelson, A. Krishnamurthy, and L. Ceze (2020) "PLink: Discovering and Exploiting Locality for Accelerated Distributed Training on the public Cloud," in *Proceedings of Machine Learning and Systems*, vol. 2, pp. 82–97.

[199] Koloskova, A., T. Lin, S. U. Stich, and M. Jagg (2020) "Decentralized Deep Learning with Arbitrary Communication Compression," in *ICLR*.

[200] Lu, Y. and C. De Sa (2020) "Moniqua: Modulo quantized communication in decentralized SGD," in *ICML*, PMLR, pp. 6415–6425.

[201] Tang, H., S. Gan, C. Zhang, T. Zhang, and J. Liu (2018) "Communication compression for decentralized training," in *NeurIPS*.

[202] Wang, J. and G. Joshi (2019) "Adaptive communication strategies to achieve the best error-runtime trade-off in local-update SGD," *Proceedings of Machine Learning and Systems*, **1**, pp. 212–229.

[203] Tran, N. H., W. Bao, A. Zomaya, M. N. Nguyen, and C. S. Hong (2019) "Federated learning over wireless networks: Optimization model design and analysis," in *IEEE INFOCOM*, IEEE, pp. 1387–1395.

[204] Wang, S., T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan (2019) "Adaptive Federated Learning in Resource Constrained Edge Computing Systems," *IEEE Journal on Selected Areas in Communications*, **37**(6), pp. 1205–1221.

[205] Wang, J., A. K. Sahu, Z. Yang, G. Joshi, and S. Kar (2019) "MATCHA: Speeding up decentralized SGD via matching decomposition sampling," in *2019 Sixth Indian Control Conference (ICC)*, IEEE, pp. 299–300.

[206] Chiu, C.-C., X. Zhang, T. He, S. Wang, and A. Swami (2023) "Laplacian Matrix Sampling for Communication- Efficient Decentralized Learning," *IEEE Journal on Selected Areas in Communications*, **41**(4), pp. 887–901.

[207] Hsieh, K., A. Harlap, N. Vijaykumar, D. Konomis, G. R. Ganger, P. B. Gibbons, and O. Mutlu (2017) "Gaia: Geo-Distributed machine learning approaching LAN speeds," in *USENIX NSDI*, pp. 629–647.

[208] Luping, W., W. Wei, and L. Bo (2019) "CMFL: Mitigating communication overhead for federated learning," in *ICDCS*, IEEE, pp. 954–964.

[209] Singh, N., D. Data, J. George, and S. Diggavi (2022) "SPARQ-SGD: Event-triggered and compressed communication in decentralized optimization," *IEEE Transactions on Automatic Control*, **68**(2), pp. 721–736.

[210] ——— (2021) "SQuARM-SGD: Communication-Efficient Momentum SGD for Decentralized Optimization," *IEEE Journal on Selected Areas in Information Theory*, **2**(3), pp. 954–969.

[211] Wang, J., A. K. Sahu, G. Joshi, and S. Kar (2020) "Exploring the error-runtime trade-off in decentralized optimization," in *Asilomar Conference on Signals, Systems, and Computers*, IEEE, pp. 910–914.

[212] Hua, Y., K. Miller, A. L. Bertozzi, C. Qian, and B. Wang (2022) "Efficient and reliable overlay networks for decentralized federated learning," *SIAM Journal on Applied Mathematics*, **82**(4), pp. 1558–1586.

[213] Le Bars, B., A. Bellet, M. Tommasi, E. Lavoie, and A.-M. Kermarrec (2023) "Refined convergence and topology learning for decentralized SGD with heterogeneous data," in *International Conference on Artificial Intelligence and Statistics*, PMLR, pp. 1672–1702.

[214] Marfoq, O., C. Xu, G. Neglia, and R. Vidal (2020) "Throughput-optimal topology design for cross-silo federated learning," *Advances in Neural Information Processing Systems*, **33**, pp. 19478–19487.

[215] Chen, X., G. Zhu, Y. Deng, and Y. Fang (2022) "Federated learning over multihop wireless networks with in-network aggregation," *IEEE Transactions on Wireless Communications*, **21**(6), pp. 4622–4634.

[216] Jia, Z., Z. Yu, H. Liao, Z. Wang, Z. Zhou, X. Wang, G. He, S. Mumtaz, and M. Guizani (2022) "Dispatching and Control Information Freshness-Aware Federated Learning for Simplified Power IoT," in *GLOBECOM*, IEEE, pp. 1097–1102.

[217] Shu, Y., Z. Wang, H. Liao, Z. Zhou, N. Nasser, and M. Imran (2022) "Age-of-Information-Aware Digital Twin Assisted Resource Management for Distributed Energy Scheduling," in *GLOBECOM*, IEEE, pp. 5705–5710.

[218] Xing, H., O. Simeone, and S. Bi (2021) "Federated learning over wireless device-to-device networks: Algorithms and convergence analysis," *IEEE Journal on Selected Areas in Communications*, **39**(12), pp. 3723–3741.

[219] Pei, J., W. Liu, L. Wang, C. Liu, A. K. Bashir, and Y. Wang (2023) "Fed-IoUT: Opportunities and Challenges of Federated Learning in the Internet of Underwater Things," *IEEE Internet of Things Magazine*, **6**(1), pp. 108–112.

[220] Pinyoanuntapong, P., W. H. Huff, M. Lee, C. Chen, and P. Wang (2022) "Toward scalable and robust AIoT via decentralized federated learning," *IEEE Internet of Things Magazine*, **5**(1), pp. 30–35.

[221] Huang, Y. and T. He (2023) "Overlay Routing Over an Uncooperative Underlay," in *The 24th International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing (MobiHoc'23)*, pp. 151–160.

[222] Tang, H., X. Lian, M. Yan, C. Zhang, and J. Liu (2018) "D^2: Decentralized training over decentralized data," in *ICML*, PMLR, pp. 4848–4856.

[223] Lu, Y. and C. De Sa (2021) "Optimal complexity in decentralized training," in *ICML*, PMLR, pp. 7111–7123.

[224] Nedić, A., A. Olshevsky, and M. G. Rabbat (2018) "Network Topology and Communication-Computation Tradeoffs in Decentralized Optimization," *Proceedings of the IEEE*, **106**(5), pp. 953–976.

[225] Neglia, G., G. Calbi, D. Towsley, and G. Vardoyan (2019) "The role of network topology for distributed machine learning," in *IEEE INFOCOM*, IEEE, pp. 2350–2358.

[226] Neglia, G., C. Xu, D. Towsley, and G. Calbi (2020) "Decentralized gradient methods: does topology matter?" in *AISTATS*, PMLR, pp. 2348–2358.

[227] Jiang, Z., Y. Xu, H. Xu, L. Wang, C. Qiao, and L. Huang (2023) "Joint Model Pruning and Topology Construction for Accelerating Decentralized Machine Learning," *IEEE Transactions on Parallel and Distributed Systems*.

[228] Vogels, T., H. Hendrikx, and M. Jaggi (2022) "Beyond spectral gap: The role of the topology in decentralized learning," *Advances in Neural Information Processing Systems*, **35**, pp. 15039–15050.

[229] Wang, J., B. Liang, Z. Zhu, E. T. Fapi, and H. Dalal (2022) "Joint Consensus Matrix Design and Resource Allocation for Decentralized Learning," in *2022 IFIP Networking Conference (IFIP Networking)*, pp. 1–9.

[230] Gong, Y., B. He, and J. Zhong (2015) "Network Performance Aware MPI Collective Communication Operations in the Cloud," *IEEE Transactions on Parallel and Distributed Systems*, **26**(11), pp. 3079–3089.

[231] LaCurts, K., S. Deng, A. Goyal, and H. Balakrishnan (2013) "Choreo: Network-aware task placement for cloud applications," in *Proceedings of the 2013 conference on Internet measurement conference*, pp. 191–204.

[232] Chen, J., H. Zhang, W. Zhang, L. Luo, J. Chase, I. Stoica, and D. Zhuo (2022) "{NetHint}:{White-Box} Networking for {Multi-Tenant} Data Centers," in *USENIX NSDI*, pp. 1327–1343.

[233] Koloskova, A., N. Loizou, S. Boreiri, M. Jaggi, and S. Stich (2020) "A unified theory of decentralized sgd with changing topology and local updates," in *ICML*, PMLR, pp. 5381–5393.

[234] Goemans, M. X. and Y.-S. Myung (1993) "A catalog of Steiner tree formulations," *Networks*, **23**(1), pp. 19–28.

[235] Lubin, M. (2017) *Mixed-integer convex optimization: Outer approximation algorithms and modeling power*, Ph.D. thesis, Massachusetts Institute of Technology, Sloan School of Management, Operations Research Center.

[236] Zhang, X., C. Chiu, and T. He (2024) "Energy-efficient Decentralized Learning via Graph Sparsification," in *The 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[237] Jiang, H., T. Kathuria, Y. T. Lee, S. Padmanabhan, and Z. Song (2020) "A faster interior point method for semidefinite programming," in *IEEE FOCS*, IEEE, pp. 910–918.

Vita

Yudi Huang

Education:

- Ph.D. in Computer Science, Pennsylvania State University, 2019-present
- M.S. in Telecommunications, University of Electronic Science and Technology of China, 2019
- B.S. in Telecommunications, University of Electronic Science and Technology of China, 2016

Selected Publication:

1. **Y. Huang**, Y. Lin, and T. He, "Optimized Cross-Path Attacks via Adversarial Reconnaissance," ACM SIGMETRICS, 2023.
2. **Y. Huang**, and T. He, "Overlay Routing Over an Uncooperative Underlay," ACM MobiHoc, 2023.
3. **Y. Huang**, T. He, N. R. Chaudhuri, and T. La Porta, "Preventing Outages under Coordinated Cyber-Physical Attack with Secured PMUs," IEEE TSG, 2022.
4. **Y. Huang**, T. He, N. R. Chaudhuri, and T. La Porta, "Link State Estimation under Cyber-Physical Attacks: Theory and Algorithms," IEEE TSG, 2022.
5. **Y. Huang**, T. He, N. R. Chaudhuri, and T. La Porta, "Preventing Outages under Coordinated Cyber-Physical Attack with Secured PMUs," IEEE SmartGridComm, 2021.
6. **Y. Huang**, T. He, N. R. Chaudhuri, and T. La Porta, "Power grid state estimation under general cyber-physical attacks," IEEE SmartGridComm, 2020. (**Finalist of Best Paper Award**)
7. **Y. Huang**, Y.-C. Liang, Feifei Gao, "Channel estimation in FDD massive MIMO systems based on block-structured dictionary learning", IEEE GLOBECOM, 2019.
8. **Y. Huang**, P. Liang, Q. Zhang, Y.-C. Liang, "A Machine Learning Approach to MIMO Communications", IEEE ICC, 2018.
9. **Y. Huang**, J. Tan, Y.-C. Liang, "Wireless Big Data: Transforming Heterogeneous Networks to Smart Networks", Journal of Communications and Information Networks, 2017.
10. **Y. Huang**, G. Yang, Y.-C. Liang, "A Fuzzy Support Vector Machine Algorithm for Cooperative Spectrum Sensing with Noise Uncertainty", IEEE GLOBECOM, 2016.

Selected Honor and Award:

- NSF Student Travel Grant, ACM MobiHoc — Oct. 2023
- Best Paper Finalist, IEEE SmartGridComm — Oct. 2020
- NSF Student Travel Grant, IEEE SmartGridComm — Oct. 2020

Selected Service:

Reviewer of IEEE TWC, TON, TNSE, TCAS-II, TCNS, TSG, INFOCOM, MILCOM