The Pennsylvania State University

The Graduate School

College of Engineering

NETWORK TOPOLOGY INFERENCE WITH PARTIAL PATH INFORMATION AND PROBABILISTIC CASCADING FAILURES THROUGH INTERDEPENDENT NETWORKS

A Thesis in

Computer Science and Engineering

by

Brett D. Holbert

Submitted in Partial Fulfillment of the Requirements for the Degree of

Master of Science

December 2014
The thesis of Brett D. Holbert was reviewed and approved* by the following:

Thomas La Porta  
William E. Leonard Professor  
Director of the Institute for Networking and Security Research  
Thesis Advisor

Guohong Cao  
Professor of Computer Science and Engineering

Raj Acharya  
Professor of Computer Science and Engineering  
Head of the Department of Computer Science and Engineering

*Signatures are on file in the Graduate School.
Abstract

Network operation and management are reliant upon accurate knowledge of the network topology. In order to carry out these network management tasks the topology must first be obtained, often using probing. However, full knowledge of the topology may neither be known nor easily obtainable due to heterogeneous ownership and configuration of the routers. In these cases, the network must be inferred. We consider two topics highly related to these problems; network inference required to obtain the topology when it is unknown and modeling cascading failures in interdependent networks as a network operation with full knowledge of the topology.

In this thesis we first present iTop, an algorithm for topology inference when complete information about a network's topology is not available. We describe how the iTop algorithm processes the partial information that is obtainable to detect the presence of missing network components and then resolve them towards an inferred topology by identifying links which may be the same. Lastly, we show that iTop outperforms pre-existing network inference algorithms both in terms of the metrics of the inferred networks as well as the suitability of the inferred networks for use with a fault diagnosis algorithm.

We further investigate the subject of network failures by proposing a probabilistic model to estimate the spread of cascading failures through an interdependent network. We use this model to calculate the estimated network outages over time for a variety of interdependent network structures. These estimations are compared to simulations of failures over the same networks in order to show that the model accurately predicts the cascade with only a small margin of error. Based on the patterns of failure for the different interdependent network structures we provide insight as to how the cascade spreads through different topologies.
# Table of Contents

List of Figures ........................................................................................................ v
List of Tables ......................................................................................................... vii
Acknowledgments ................................................................................................ viii

Chapter 1. Introduction ............................................................................................ 1

Chapter 2. Related Work ......................................................................................... 5
  2.1. Topology Discovery ............................................................................................ 5
  2.2. Cascading Failures ............................................................................................. 7
  2.3. Interdependent Networks ................................................................................. 10

Chapter 3. iTop Algorithm for Topology Inference ..................................................... 13
  3.1. Network Model for Partial Information ............................................................. 15
    3.1.1. Traceroute .................................................................................................... 15
    3.1.2. Measuring Non-Cooperative Routers ......................................................... 16
    3.1.3. Network Model ........................................................................................... 19
  3.2 The iTop Algorithm ........................................................................................... 20
    3.2.1 Virtual Topology Construction .................................................................... 20
    3.2.2 Merge Options ........................................................................................... 24
    3.2.3 Merging Links ............................................................................................ 28
  3.3. Evaluation of Inferred Network Metrics .......................................................... 34
    3.3.1 Comparison Approaches ............................................................................. 34
    3.3.2 Topology Inference Experiments ................................................................. 36
  3.4. Fault Diagnosis with Inferred Topologies ....................................................... 47
    3.4.1 Max-Coverage with Partial Information ..................................................... 47
    3.4.2 Fault Diagnosis Experiments ..................................................................... 50

Chapter 4. Model for Probabilistic Cascading Failures ................................................. 54
  4.1. Propagation Model ............................................................................................ 56
    4.1.1. Case 1 - Independent Parent Nodes ......................................................... 58
    4.1.2. Case 2 - Shared Dependencies in Parent Nodes ....................................... 60
    4.1.3. Case 3 - Inter-Edge Loops ...................................................................... 62
  4.2. Model Verification ............................................................................................ 64
    4.2.1. Experiment Setup ..................................................................................... 64
    4.2.2. Failure Propagation Experiments ............................................................. 68

Chapter 5. Conclusions ............................................................................................ 75

References ............................................................................................................... 78

Appendix: Full Propagation Results ......................................................................... 81
List of Figures

Figure 2.1: Example Bayesian Network ......................................................... 9
Figure 2.2: Example Dynamic Bayesian Network Timestep ............................... 10
Figure 3.1: Avg. Number of Responding and Anonymous Routers per Path ........ 18
Figure 3.2: Example GT and Virtual Topologies ............................................. 24
Figure 3.3: Node Class Hierarchy ..................................................................... 26
Figure 3.4: iTop Merging Algorithm ................................................................. 27
Figure 3.5: Example $G_{MT}$ Topologies before, during, and after the Merging Process ..... 33
Figure 3.6: Example Inferred Topologies with 10% Anonymous and 10% Blocking Router for GT, iTop, MN, and Isomap 38
Figure 3.7: Realistic Networks with Anonymous and Blocking Routers: .......... 40
  Number of Nodes, Number of Links, Cumulative Distribution of
  Node Degree, and Betweeness Centrality
Figure 3.8: Joint Degree Distribution with Respect to GT: iTop, MN, and Isomap 42
Figure 3.9: Realistic Networks with Only Anonymous Routers: Number of Nodes, ...... 44
  Number of Links, Cumulative Distribution of Node Degree, and
  Betweeness Centrality
Figure 3.10: Random Networks with Anonymous and Blocking Routers: .......... 46
  Number of Nodes, Number of Links, Cumulative Distribution of
  Node Degree, and Betweeness Centrality
Figure 3.11: Max-Coverage Algorithm .............................................................. 48
Figure 3.12: Average Accuracy of MC with GT and Inferred Topologies .......... 51
Figure 3.13: Average False Positives of MC with GT and Inferred Topologies .......... 52
Figure 3.14: Average Redundancy of MC with GT and Inferred Topologies .......... 53
Figure 4.1: Independent Parent Nodes Example .............................................. 58
Figure 4.2: Shared Dependencies Example ..................................................... 60
Figure 4.3: Inter-Edge Loop Example ............................................................. 63
List of Tables

Table 3.1: Percent of Traces with Unreachable Destinations ........................................ 18
Table 3.2: Compatible Endpoint Classes and Resulting Classes after Merging ............. 27
Acknowledgments

I would like to thank Dr. Thomas La Porta for being my research advisor and guiding me through my time in graduate school. Over the last few years I could always count on his support to refocus me when I lost sight of the goal of my work and help me find what really mattered out of a mountain of data.

I would also like to thank Dr. Guohong Cao for being on my thesis committee and providing his input and comments in reviewing my thesis.

This page would be incomplete without thanking Dr. Simone Silvestri for the guidance and assistance he has provided. His help has been indispensable for brainstorming, writing, and noticing when I was missing something blatantly obvious.

Lastly, I would like to thank my family for their help and moral support during my time at Penn State, even though any mention of the content of this thesis was only met with looks of bewilderment and confusion.

This work was supported by the Defense Threat Reduction Agency under grant HDTRA 1-10-1-0085.
Chapter 1

Introduction

A network’s topology is a critical aspect that drives the way the network responds to faults or how failures may propagate through the network. Likewise, knowledge of the topology is important for diagnosing failures and performing other management tasks. Although it is frequently assumed that this topology information is known, in reality it is not always easily obtainable. Unplanned changes to the topology and heterogeneous network ownership can easily interfere with methods used to measure the network. In particular, topology inference algorithms using Traceroute [1] often overestimate the amount of data that can be collected. On its own, Traceroute returns information which is far from complete and must be further supplemented and processed in order to obtain a usable topology. The resulting topology can then be used as input for the required tasks, such as identifying network components that have failed.

Beyond IP networks, many other aspects of both national and local infrastructures are also commonly modeled with networks. While the topologies and purposes of each of these types of networks are vastly different, in practice they frequently rely on each other in order to maintain a state of correct operation. For example, the power grid provides electricity to routing hardware, and that hardware transmits the data required for proper load balancing. These networks are often considered separately, but when examined together they provide an example of an interdependent network. Interdependent networks
consist of two or more separate networks connected by a set of links which represent the manner in which they interact and support one another. This provides the ability to examine the interactions between the networks and detect changes in the operation of a single network that may occur in response to events occurring in the others.

One such event is the occurrence of cascading failures. In a cascading failure, a set of initial failures occurring in one or more nodes spreads to additional nodes. This can be due to a variety of domain-specific reasons such as the nodes receiving extra load to compensate for failures in a power grid or no longer being able to send or receive data through failed nodes in the Internet. Interdependent networks introduce additional ways for cascades to propagate that are not present in the individual networks when considered separately. A failure within one network may cause secondary failures to occur in portions of the other networks but not in its own network. These secondary failures may then cause additional failures to occur in the first network and continue propagation of the cascade throughout the interdependent network instead of stopping after the initial failure.

In this thesis we introduce a topology inference algorithm for use when complete information is not available and a model for the propagation of probabilistic cascading failures. Our inference algorithm, known as iTop, infers network topologies from partial Traceroute results so that the information can be used for fault diagnosis and other network management tasks. We further continue with network failures in our propagation model. With this model we are able to estimate how a failure scenario is expected to evolve over time and what the network’s end state will be after the resulting cascade.
In Chapter 2 we review previous work that focuses on approaches and models concerning topology discovery, interdependent networks, and cascading failures. We begin by describing two common classes of topology discovery algorithms, network tomography and Traceroute-based inference. Algorithms of these types are widely used and each has its own benefits and drawbacks depending on what metrics of the network are desired. We continue with an overview of cascading failures. Study of these types of failures has been popular in a variety of fields in recent years, and we highlight popular approaches for modeling them both in general and with regards to the power grid specifically. Additionally, some of these models have begun to consider interdependent networks and how cascades will affect them.

Chapter 3 presents the iTop algorithm for topology inference with partial information. We give an overview of the operation of the Traceroute protocol and describe the network scenario for which iTop is intended. It first constructs a virtual topology from the information collected by Traceroute and end-to-end path lengths. This virtual topology vastly overestimates the number of components in the measured network, but provides a starting topology in which all nodes and links in the end-to-end paths are guaranteed to be present at least once. A set of rules are applied to the links in the virtual topology in order to rule out links which cannot be the same in the real topology. Lastly, iTop iteratively merges pairs of nodes in the virtual topology in order to construct an inferred topology which best estimates the ground truth network.

We demonstrate the effectiveness of iTop by presenting metrics for the topologies it infers on a variety of networks. We compare these metrics to those of the ground truth as well as those measured from topologies inferred by other Traceroute-based inference
algorithms. Based on these measures, iTop most closely infers the structure and size of
the original network. We further show that the topologies inferred by iTop are more
suitable for use with Max-Coverage [2], a popular algorithm for fault diagnosis. This
demonstrates that iTop is also more useful for tasks which rely upon knowing the
network topology.

Chapter 4 introduces our model for representing interdependent networks and
probabilistically estimating cascading failures as they spread over time. We define the
formal representation of the interdependent network and the formulas used to calculate
the probabilities of cascade propagation along the inter-edges connecting the individual
networks. These formulas allow for estimations of when individual nodes are expected to
fail and the predicted cascade size at times following the initial failure. While the final
state of the network and resulting size of the outage are important, the functionality of
individual nodes during the cascade provides insight as to the reason the failures evolved
into a cascade. Our model allows these intermediate states of the network to be examined
while also allowing for the occurrence of failures which may not be immediate but
instead occur after a delay.

We evaluate the accuracy of this model by comparing it to an event-driven
simulation of the spread of cascading failures. Our results show that the model is able to
closely match the average outcome of the simulations for a variety of network structures
and failure propagation rates. We analyze the pattern of the cascade across the different
networks to provide insight as to how the structure and connectivity of the network affect
how quickly the cascade begins to spread and fully permeate the network.
Chapter 2  

Related Work

Since the structure of a network must be known in order to operate upon it, much work has been done to address the subject of topology inference. In recent years, interdependent networks and cascading failures have also been examined on numerous occasions, both separately and together, as they concern disastrous events in vital infrastructures. This chapter reviews previous work which has been carried out in these areas and how it relates to the iTop algorithm and interdependent cascading model presented in this thesis.

2.1 Topology Discovery

Algorithms for topology discovery generally fall under the two areas of Traceroute-based approaches and network tomography. In both types of algorithms, monitor nodes are used to obtain information about the topology by sending probes to measure the network. The structure of the network is then inferred by processing the collected information.

Traceroute-based approaches [3, 4, 5] use the Traceroute protocol to probe paths in the network [1]. A monitor uses Traceroute to probe the path to a destination in the network by sending hop-limited packets. When all intermediate nodes are cooperating the
monitor receives a *trace* of the path, or a list of IP addresses for all of the routers that the path contains. The traces collected by all of the monitors are sent to the Network Operation Center (NOC), which processes them and constructs a network topology based on their contents. Many existing Traceroute-based inference algorithms assume that intermediate routers all send a response to Traceroute and instead focus on the alias resolution problem when a single router may reply with different IP addresses on different paths [6, 7]. In reality, many routers are configured to not operate as expected with regards to Traceroute [8], so additional methods of inference are required. We further describe the operation of Traceroute in Chapter 3.

Network tomography approaches focus on collecting measurements of additive metrics on end-to-end paths between monitor nodes, such as the time it takes for a probe to traverse the path [9, 10, 11]. While this has the advantage of not relying upon any cooperation from intermediate routers within the network, tomography approaches do have their own set of drawbacks. They are only able to infer a simplified, network-layer representation of the topology since they do not directly measure the underlying links and routers in the network. Additionally, many of these approaches use systems of linear equations to find a best fit for assigning the metrics to components in the network, which requires advance knowledge of the number of links and nodes that the network contains.
2.2 Cascading Failures

In many types of networks the effects a single component’s failure are not limited to affecting the performance of only that node. The node no longer forwards traffic in a communication or data network, forces its load to be carried by other nodes in a power network, spreads a contagious disease in a network of social interactions, etc. In all cases, the failure of a small group of nodes can quickly cascade, compromising additional nodes until the failure has spread to a large portion of the network. Models of these cascades are often concerned with determining the extent to which the cascade will spread. In some cases, this is carried out by identifying the average rate of transmission and node degree and subsequently calculating whether the cascade is expected to permeate through the entire network or whether the network is expected to stabilize and limit the spread of the cascade to only a small area [12].

Cascading failures are of a particular interest to power networks due to how load balancing redistributes demand upon the failure of a node. CASCADE, a common model for cascading failures caused by load balancing, sets load limits for each node in the network [13]. When a node's load exceeds its limit, it fails and shifts a portion of its load to each of its neighbors. This may then cause them to exceed their load limits and continue to propagate the failure. While this effectively models cascades as they pertain to power grids, this cause of failure does not apply to other types of networks. Other work has focused on estimating the original state of the network once a cascade has occurred [14]. By observing cases of the final network state after the cascade has terminated, it attempts to recreate the initial failures and how they spread throughout the grid. This estimated failure process takes into account the fact that failures occur progressively over
time, but it considers that the effects of any given failure take place only in the next defined time step and not beyond that.

While these works deterministically model cascading failures due to exceeding power load limits, others have taken a more probabilistic approach in which the failure of a node has a certain likelihood of causing failures in its neighbors. These models are generally implemented through the use of Bayesian networks [15]. A Bayesian network is an acyclic directed graph in which each node in the graph has a set of mutually exclusive states that it can be in. Additionally, each node has a conditional probability which denotes the likelihood it is in each of its states given the states that its parent nodes are in. This makes it possible to calculate the expected state of any node in the network either in general or if the states of some subset of nodes in the network are known. Figure 2.1 shows an example of a simple Bayesian network in which each node can be in either the “True” or “False” states. It also lists the expected value that Node D is in the false state for different sets of known states for the other nodes. With similar expected values for B and C given the state of A, the probability of D being true or false can be calculated given the state of A.
Bayesian networks are often used to model cascading failures due to their directed and probabilistic nature. Each node in a Bayesian network is used to represent a different component or node from the represented network in either a failed or working state [15]. Some models include additional states to indicate that a node is in the process of failing [16]. The directed edges in the Bayesian network lay out possible paths of propagation for the cascade with conditional probabilities denoting the probabilities of the cascade continuing. A major limitation of basic Bayesian Networks is the inability to model the state of the network over time. Each node's states only pertain to whether it is operating or failed at a single moment in time. Dynamic Bayesian networks (DBNs) compensate for this limitation by modeling the change in node states from one time step to the next [16, 17]. The simple Bayesian network representing the structure of the modeled network is duplicated for each time step. Edges are added to the DBN to connect each node to the

\[
\begin{align*}
P(D|B=\text{false}, C=\text{false}) &= 0.1 \\
P(D|B=\text{true}, C=\text{false}) &= 0.3 \\
P(D|B=\text{false}, C=\text{true}) &= 0.3 \\
P(D|B=\text{true}, C=\text{true}) &= 0.4
\end{align*}
\]
node representing the same network component in the next time step. Additionally, if a node affects another node in the current time step then an edge is added that connects it to the node in the next time step. The conditional probabilities taking into account these edges allow the node's state to be tracked over time and be maintained once it has reached the failed state. Figure 2.2 shows an example of a DBN representing the transition from time $t$ to time $t'$ for the Bayesian network shown in Figure 2.1.

![Dynamic Bayesian Network Timestep](image)

**Figure 2.2 Dynamic Bayesian Network Timestep**

### 2.3 Interdependent Networks

Traditionally networks are considered on their own in the context of the role that they perform. Power networks, information networks, and others each have typical structures and problems which are specifically associated with them and are therefore
considered separately. However, in practice these networks are not completely isolated and performance degradations or failures in one may affect the operation of the others. Interdependent networks are composed of two or more of these individual networks and include additional links that indicate nodes in the two separate networks which rely upon each other. Due to the heterogeneous nature of the individual networks, interdependent networks often have structures which do not follow the typical assumptions and structures often applied when examining single, homogenous networks [18].

Typically interdependent networks are used to model the relationship between the power grid and information or communication networks. Electricity is required in order to transmit and route Internet or voice traffic, and data communication is frequently used to determine power generation and load balancing. Interdependent networks modeling these two networks are common, but are generally domain-specific [19]. While this enhances the ability to represent how these two networks interact, it also has resulted in models of interdependent networks that are too specific to be applied to others types of individual networks.

Due to the reliance between the individual networks in an interdependent network, a failure in one of the networks that alone would be isolated can instead start a cascade when the interdependence between the networks is taken into account. This was clearly evidenced in a blackout which took place in Italy in 2003 [20]. While the initial failure took place in the power grid, this caused the loss of power in some critical portions of the Internet routing infrastructure. These secondary failures inhibited the ability of the power grid to effectively load balance, leading to further power grid failures. This process continued until the majority of the country was without power.
The vulnerability of interdependent networks to cascades is due to the differences between single, homogeneous networks and heterogeneous interdependent networks. The interdependent links added between the individual networks introduce many changes as to what makes a node vulnerable to failure [18]. A high degree node in a communication network may have access to multiple backup paths if an adjacent node fails and therefore be considered robust in the context of that single network. However, if it depends upon a low-degree node in the power grid then that will become prone to failure since it relies on a node which can fail much easier than it can.
Chapter 3

iTop Algorithm for Topology Inference

Knowledge of a network’s routing topology is of fundamental importance for a wide variety of management tasks. For example, the design of overlay networks [21] and the localization of failures [2] can significantly benefit from accurate and thorough knowledge of the network topology [9, 22]. This information is often unavailable for various reasons such as unplanned changes, network dynamics and heterogeneous network ownership. As a result, the network topology must often be inferred by means of topology inference algorithms.

Traceroute-based inference techniques [3, 4, 5, 6, 7] use the Traceroute utility [1] to collect path information. Traceroute sends hop-limited packets from a source node to a destination in a network. The source collects responses from intermediate routers and constructs a Traceroute trace. The Network Operation Center (NOC) collects the traces from all monitors and uses this information to infer the topology. Traceroute-based techniques are more effective in inferring the routing topology than network tomography approaches when the number of network components is unknown.

Most previous approaches based on Traceroute [3, 6, 7] assume that internal routers fully comply with the information collection. However, in practice router configurations, privacy policies, and firewalls may prevent some routers from correctly cooperating, as demonstrated in [8] and through real experiments in Section 3.1. We
classify the behavior of internal routers with respect to Traceroute probes in three
categories: responding, anonymous, and blocking routers. Responding routers correctly
participate in Traceroute operations. Anonymous routers do not send responses back to
the source but do forward requests to other routers in the path. Blocking routers drop all
Traceroute packets and do not send responses, preventing the collection of any further
information about the path. We refer to blocking and anonymous routers collectively as
non-cooperative routers.

In this chapter we present iTop, our algorithm for topology inference when there
are non-cooperative routers in the network. This algorithm is a Traceroute-based
approach that identifies the existence of non-cooperative routers in the traces and
accounts for them by measuring path lengths with additional packets. The resulting
topology contains duplicates of some network components that were not directly
observed, so iTop applies a set of rules to determine which links and nodes are different
in the real topology. It then iteratively merges pairs of links that may be the same in order
to resolve towards the final inferred topology.

We compare the networks that iTop infers to those inferred by other inference
algorithms which consider anonymous and/or blocking routers [4, 5]. We consider both
network metrics for the topologies that the algorithms infer and the suitability of the
topologies when used as input for fault diagnosis. Through these experiments, we show
that iTop is able to infer topologies which are both more accurate and more useful than
those inferred by the other algorithms.
3.1 Network Model for Partial Information

Here we describe the model of the network for which iTop is designed to operate. We begin by giving an overview of Traceroute, as the traces it provides are pivotal for the inference that iTop performs. Unlike many previous inference algorithms, we do not assume that all routers in the network reply to Traceroute probes as expected. We classify routers into three types in order to capture the way that they react when receiving a probe. In order to show that this is an accurate representation of routers in the Internet, we present an experiment designed to confirm the existence and impact of these router classes.

3.1.1 Traceroute

Traceroute [1] is a well-known tool for path discovery in IP-based networks. The tool sends hop-limited packets from a source towards a destination. Traceroute works in rounds, where in the $k$-th round packets are sent with a Time to Live (TTL) value of $k$. Intermediate routers on the path decrement the TTL and alert the source if the packet expires before reaching the destination. By collecting these error messages, the source constructs a trace, or the list of routers on the path to the destination.

Three main variants of Traceroute are commonly used: ICMP-, UDP- and TCP-based, which differ in the type of packets that are exchanged [8]. ICMP Traceroute sends ICMP Echo Request packets towards the destination and intermediate routers reply with an ICMP Time Exceeded or Time to Live Expired in Transit packet when the TTL reaches zero. When the destination receives the packet it replies with an ICMP Echo
packet. UDP- and TCP- based Traceroute work in a similar manner, but use different
types of exchanged packets. UDP Traceroute transmits UDP packets to an invalid
destination port value, while TCP Traceroute sends a TCP SYN packet to a well-known
port, such as the default port of a web server.

Due to these differences, the three variants of Traceroute possess varying
capabilities to acquire path information to the destination. For example, intermediate
routers can be configured to not reply to or completely discard ICMP packets.
Furthermore, firewalls may filter out packets to unknown ports or ports that do not
belong to already established TCP connections.

### 3.1.2 Measuring Non-Cooperative Routers

In order to process the gathered traces, we classify network routers into three
categories with respect to their behavior regarding Traceroute probes. *Responding* routers
are routers that correctly participate in Traceroute and reply to the source as described
above. *Anonymous* routers do not send a reply to the source when the TTL of a
Traceroute probe expires, but do forward Traceroute requests and replies coming from
other routers. Finally, *blocking* routers neither respond to the source as the TTL expires
nor forward Traceroute probes and replies coming from other nodes.

In this section we describe real experiments we performed to show that all
Traceroute variants only allow the collection of partial path information. Our results
confirm the experimental analysis provided in [8]. The experiments were executed as
follows. Traces were collected by using ICMP, UDP, and TCP Traceroute to probe the
paths to a set of 100 destination websites from a single source located on the
Pennsylvania State University, University Park campus. For UDP and TCP Traceroute,
traces were collected using the default destination port numbers. We also collected traces
using other ports and observed similar results. The destinations were selected to include a
wide variety of sites from various locations around the world. The maximum path length
was set to 30, so each individual trace is of the form \((x_1, x_2, \ldots, x_n)\) for \(n \leq 30\), where
each \(x_i\) contains either an identifier of the router located \(i\) hops from the source if a
response was received or the indicator * for no response otherwise. If a response was
received for router \(x_i\) then it must be a responding router. Any router \(x_j\) which did not
provide a response is classified as anonymous if there is some responding router \(x_i\) for
which \(i > j\). A path contains a blocking router if there exists an \(x_k\), such that for each \(j \geq k\),
\(x_j\) equals *. Note that we cannot conclude that \(x_k\) is the blocking router, as the same trace
can be generated by a series of anonymous routers immediately preceding a blocking
router.

Figures 3.1 (a) and (b) show the average number of responding and anonymous
routers in each path. ICMP Traceroute is able to identify more responding routers than
the other versions, as ICMP packets are not discarded as often as UDP and TCP packets.
For this reason, ICMP Traceroute also discovers a higher number of anonymous routers,
as shown in Figure 3.1 (b). These results highlight that on average more than 10% of the
routers in a path are anonymous.
Table 3.1 shows the percentage of unreachable destinations for each type of Traceroute. In this case, ICMP Traceroute performs better than UDP and TCP Traceroute. Note that the percentage of unreachable destinations using ICMP Traceroute can be underestimated due to some routers spoofing the source address of the ICMP response [8]. Regardless, all versions are significantly affected by the presence of blocking routers, as even with ICMP Traceroute 34% of the destinations cannot be reached. As a result, a large portion of the topology remains unobserved. These results motivate the need for an algorithm for topology inference that specifically takes into account the limitations of information provided by Traceroute.

<table>
<thead>
<tr>
<th>Trace Type</th>
<th>ICMP</th>
<th>UDP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unreachable Destinations</td>
<td>34%</td>
<td>90%</td>
<td>67%</td>
</tr>
</tbody>
</table>

Table 3.1: Percent of Traces with Unreachable Destinations
3.1.3 Network Model

We refer to the real topology of the network as the *Ground Truth* (GT) topology. This is represented by an undirected graph $G_{GT} = (V_{GT}, E_{GT})$, where $V_{GT}$ is the set of nodes in the network and $E_{GT}$ is the set of links connecting them. A set of nodes in $V_{GT}$ is designated as monitors in the network. Similar to previous works on topology inference [4, 5], we assume that the routing algorithm uses shortest paths and that a single fixed path is used between each pair of monitors. Our approach can be easily extended to different routing strategies. We further assume that pairs of monitors have a mechanism to measure the hop distance of the path between them. For example, the hop distance can be measured using the TTL field of UDP packets exchanged between monitors. By setting the TTL to a large known value, monitors can measure how much it has decreased when the packet is received and calculate the number of hops traversed. These packets are not Traceroute packets and are exchanged between cooperative hosts on open ports. As a result, they are not discarded by intermediate routers.

As previously described, we assume that anonymous routers forward Traceroute packets but do not send responses when the TTL expires, while blocking routers discard and ignore any Traceroute packets they receive. We assume that routers are consistent in their behavior with respect to Traceroute probes during the monitor information collection. We assume that traces are pre-processed at the NOC through standard alias resolution techniques [3, 6, 7] before the execution of topology inference algorithms. Our objective is to infer a topology from the partial information collected by monitors, which is as close as possible to the GT topology.
3.2 The iTop Algorithm

In this section we present iTop, our topology inference approach for partial path information. The iTop algorithm operates in three phases. In the first phase, it analyzes the traces and constructs the virtual topology (VT) $G_{VT} = (V_{VT}, E_{VT})$, which is a vastly overestimated topology compared to the ground truth. During the construction of the virtual topology, iTop classifies nodes in the virtual topology on the basis of their observed behavior with respect to Traceroute probes. In the second phase, iTop determines the merge options for each link in the virtual topology. These options indicate pairs of links in the virtual topology that can be merged while preserving consistency with respect to characteristics of the ground truth topology observed in the traces and the node classifications assigned in the previous phase. In the third phase, iTop infers the merged topology (MT) $G_{MT} = (V_{MT}, E_{MT})$, by iteratively merging pairs of links based on their merge options and removing any options become invalid as a result.

3.2.1 Virtual Topology Construction

The NOC collects the information gathered by all monitors and constructs the virtual topology as follows. Consider two monitors $m_1$ and $m_2$ connected by the path $m_1, v_1, v_2, \ldots, v_{n-1}, m_2$. In order to maximize the gathered information, both monitors take part in the information collection process by probing the path. The monitors first estimate their mutual hop distance $d(m_1, m_2)$ and then execute Traceroute. The gathered traces and hop distances are sent to the NOC, which analyzes them to infer the virtual topology and
partition the nodes into five classes. These classes are used to guide the merging process of iTop and reflect the router behavior as observed by the Traceroute probes.

We define five classes of routers. $R$ is the class of responding nodes, $A$ is the class of anonymous nodes, $B$ is the class of blocking nodes, $NC$ is the class of non-cooperative nodes, and $HID$ is the class of hidden nodes which occur between blocking nodes. When processing the collected traces, the NOC marks each router as belonging to one of these classes as follows.

Consider the case in which a monitor $m_1$ uses Traceroute to probe the path to $m_2$ and successfully receives a response from $m_2$. Since a reply was received, the NOC can conclude that the path does not contain any blocking router. As described in Section 2, the trace from $m_1$ will be of the form $(x_1, ..., x_{n-1}, m_2)$ where $d(m_1, m_2) = n$ and each $x_i$ either identifies a router $v_i$ that is responding or is a * to denote no response. As there are no blocking routers in the path, all routers corresponding to a * in the trace must be anonymous. The NOC adds a node in the virtual topology for each responding router observed in the trace and connects them accordingly. It marks these nodes as responding and combines multiple instances of the same responding routers reported by other monitors to avoid duplication of observed components. The anonymous routers and the links connecting them are also added to the virtual topology. A node is added for each * in the trace and it is marked as anonymous.

If the path does contain at least one blocking router then the trace acquired by $m_1$ will not contain a response from $m_2$. In this case the NOC can combine the traces obtained by $m_1$ and $m_2$ to create the trace $(m_1, x_1, ..., x_i, *, ..., *, x_{n-j}, ..., x_{n-1}, m_2)$ where $x_i$
and $x_{n-j}$ are the last responses received by $m_1$ and $m_2$, respectively. The NOC treats the trace fragments $(x_i, \ldots, x_i)$ and $(x_{n-j}, \ldots, x_{n-1}, m_2)$ as described above, adding responding and anonymous routers in the virtual topology. The NOC uses the hop distance $d(m_1, m_2)$ to infer that there are $n-j-i-1$ unobserved routers in the trace fragment $(x_{i+1}, \ldots, x_{n-j-1})$. The NOC adds these routers to the virtual topology and connects them to fill the gap in the path between the routers already added. How these nodes are marked depends upon the number of unobserved routers. If there is only one router between $x_i$ and $x_{n-j}$, that is $i+1 = n-j-1$, then that router must be blocking and is marked as such. Otherwise, the NOC can only conclude that there is at least one blocking router in the fragment, but cannot uniquely identify it. As a result, it marks $x_{i+1}$ and $x_{n-j-1}$ as non-cooperative while all routers between them, if any, are marked as hidden since there is no way to determine whether they are responding or not. This process is repeated for all communicating monitor pairs.

The fully constructed VT topology overestimates the network because it contains multiple anonymous, blocking, and hidden nodes which are the same router in the GT topology. Since these routers are all represented in the traces by a * there is no simple way to determine which ones are the same. Therefore they are assumed to be separate nodes until merged in the third phase of iTop.

Figures 3.2 (a-b) show an example of a GT topology and the corresponding virtual topology. The legend in Figure 3.2 (a) indicates the type of each node as it exists in the GT topology and is marked by the NOC in the VT topology. In this example we assume that nodes A, E, F, G, and I act as monitors and the following pairs of monitors

In the path A-F, the blocking router B drops all Traceroute packets sent from A. As a result, A receives no reply from C or F. In the other direction, F can detect the segment F-C, but B prevents the collection of further information. By using the measured hop distance, F and A can determine that the path is composed of exactly three hops. Therefore, the NOC determines that there is only one blocking router between them and adds the path A-B1-C-F to the virtual topology, with B1 marked as blocking.

In the path A-E, the blocking routers B and D prevent the identification of the responding router C. The monitors measure a path length of four, and the NOC adds the path A-B2-C1-D2-E to the virtual topology. Both B2 and D2 are marked as non-cooperative, while C1 is marked as hidden. This is repeated for all paths in the network. The resulting virtual topology is shown in Figure 3.2 (b).

Note that the virtual topology contains a larger number of nodes and links than the GT topology. Some of the nodes in Figure 3.2 (b) corresponding to the same nodes in the GT topology. In the case of nodes B, D, and H, this is because they are non-cooperative, thus the traces for paths they occur in only contain a * for their response. C is located between blocking routers B and D, so it also appears in the VT as a hidden node.
3.2.2 Merge Options

In order to infer $G_{MT}$ from $G_{VT}$, iTop identifies the valid merge options for each link $e_i \in E_{MT}$, i.e. the set of links with which $e_i$ can be merged. We introduce three conditions which have to be satisfied for a merge option to be valid. These conditions check the consistency of a merge option with the information gathered from the traces and with the node classification provided in the previous phase.
The set $M_i$ denotes the set of links which are valid merge options for link $e_i$ and initially $M_i = \emptyset$ for each $e_i \in E_{MT}$ The sets of merge options are then used during the merging phase to determine which merges occur and in what order. We define the following conditions for merge options, which are checked for each pair of links.

**Trace Preservation:** Since paths do not contain loops, a link will never appear twice in the same path. A merge option between two links satisfies the trace preservation if these links do not appear together in any path.

**Distance Preservation:** The distance between two monitors in $G_{VT}$ is consistent with the traces. The merging process keeps this consistency by preventing any merges which would decrease this distance. As a result, a merging option between two links satisfies the distance preservation if merging these links will not make the distance between any two communicating monitors in $G_{MT}$ shorter than in $G_{VT}$.

**Link Endpoint Compatibility:** The node classification gives us additional information on valid merging options. In particular, the sets of node classes form a hierarchy, as shown in Figure 3.3. This hierarchy represents the precision with which we are able to determine the behavior of nodes with respect to the received traces. Leaf classes are definite while internal classes are more generic, as the traces do not provide enough information for a more specific classification. Intuitively, the merging process can only increase the specialization of a node.
A merge option between two links is valid if there is a way to combine their endpoints without violating the hierarchy. Table 3.2 shows the types of endpoints that compatible links can have. Entries in this table can be reversed, for example $R-A$ is the same as $A-R$. If two links have incompatible endpoints then the entry is marked with “-“. Otherwise the entry contains the endpoint classes of the link that would result if the links were merged. Bolded entries indicate merges that will combine two routers with class $R$. These options are valid only if the responding router is the same for both links. How the endpoint types of the merged links are decided is further described in the merging phase section.
Referring to the example VT topology in Figure 3.2 (b), it is possible to see how combinable endpoints supplement distance and trace preservation. Those two rules alone provide no restrictions on the classes of link endpoints that can be combined, so merging all vertically aligned links, such as (A, B1), (A, B2), (A, H1), and (G, H2), would be possible without checking endpoint compatibility. Table 3.2 shows that R-B and R-A links should not have a merge option, so this will prevent (A, B1) from having a merge option with either (A, H1) or (A, H2). Further, the fact that the identities of responding nodes are known will prevent (A, H1) and (G, H2) from satisfying endpoint
compatibility, but will allow (H1, I) and (H2, I) to satisfy all three merge option requirements.

### 3.2.3 Merging Links

The next phase merges the links in the virtual topology to derive the iTop topology $G_{MT}$. Initially, $G_{MT} = G_{VT}$. The merging phase reduces $G_{MT}$ by iteratively merging pairs of links based upon the existing merge options. Each merge performed combines two links in $E_{MT}$, combining their endpoints and reducing the number of components in the network accordingly. When no merge options remain, the merging phase is complete and the iTop topology is finalized.

---

**Algorithm 1: iTop Merging Phase**

**Input:** Initial iTop Topology $G_{MT} = (V_{MT}, E_{MT})$.

**Output:** Merged iTop Topology $G_{MT} = (V_{MT}, E_{MT})$.

1. while $\exists e_i \in E_{MT} \land M_i \neq \emptyset$ do
2.   $e_i = \arg\min_{e_i \in E_{MT}} |M_i|$;
3.   $e_j = \arg\min_{e_j \in M_i} |M_j|$;
4.   // Endpoint compatibility check
5.   if $C(e_i, e_j) = true$ then
6.     // Link merging
7.     Merge($e_i, e_j$);
8.   else
9.     $M_i = M_i \setminus \{e_j\}$;
10.    $M_j = M_j \setminus \{e_i\}$;
11. return $G_{MT} = (V_{MT}, E_{MT})$.

---

Figure 3.4: iTop Merging Algorithm
Figure 3.4 shows the pseudo-code of the merging phase. In each step, iTop chooses two links and attempts to merge them. Several alternatives are possible to determine the order in which links are merged, which influence the resulting final topology. Since links with few merging options have less merging possibilities, they are more likely to be the same link in the ground truth topology. On the basis of this observation, we first select the link $e_i$ with the fewest merging options, and then link $e_j$ which has the fewest merging options out of the links with which $e_i$ can be merged (Figure 3.4, lines 2-3). We experimented with several alternative heuristics and the one described above provides the best results.

**Endpoint Compatibility Check:** Before merging two links, their endpoint compatibility is rechecked by the function $C()$, (Figure 3.4, line 4), according to Table 2. This check is necessary as previous merges may have changed the links' endpoint classes since the initial check during the merge option phase.

Additionally, each path containing one of the two links is checked to make sure it will retain a coherent ordering of link endpoints should the merge occur. This coherence check is necessary because iTop is designed to keep the endpoint classes as generic as possible during the merging phase. Furthermore, as explained in the Link merging section below, iTop does not commit to one specific direction if there are two alternatives to combine the endpoints of the links being merged until this is implied by the merge operation. Since each link has two endpoints, there are two different ways in which the endpoints can be combined and both may result in the same amount of generality without
violating distance preservation. A path consisting of links \( e_i, \ldots, e_n \) is coherent if there exists a mapping of node classes \( c_1, \ldots, c_{n+1} \) such that the endpoint classes of each \( e_i \) are combinable with classes \( c_i \) and \( c_{i+1} \), according to the node class hierarchy. The mapping should contain specific responding routers instances, instead of just the responding class, because each responding router is identifiable.

For the VT topology in Figure 3.2 (b), consider checking the coherence of path A-B2-C-D2-E if the link (B1, C), of type \( R-B \), is merged with (B2, C1), which has type \( NC-HID \). This merge will result in a link with endpoint classes \( R-B \). The node class ordering Node A, B, Node C, B, Node E shows that this path is coherent.

If the link types are not compatible or the merge will cause an incoherent path then the function \( C(e_i, e_j) \) in the pseudo-code of Figure 3.4 returns false. In this case, links are removed as merging options for each other (Figure 3.4, lines 7-8) and the algorithm repeats the step to choose a new pair of links. Otherwise, if the link types are compatible and all paths are coherent, the link \( e_i \) is merged with \( e_j \) as described in the following.

**Link Merging:** If two links \( e_i \) and \( e_j \) pass the compatibility check, they are merged by the function \( Merge(e_i, e_j) \) (Figure 3.4, line 5). For ease of exposition we describe the merging of \( e_j \) into \( e_i \), as the same result would be obtained by the opposite merging. All paths containing \( e_j \) are modified to contain \( e_i \) in its place and the set \( M_i \) is changed so that \( M_i = M_i \cap M_j \). Any links which could have been merged with both \( e_i \) and
$e_j$ retain their merge option with $e_i$ and the merge option for $e_j$ is removed. Any links which had a merge option with either $e_i$ or $e_j$ but not both have that option removed. This ensures that $e_i$ can only be further merged with links that before were valid merges for both $e_i$ and $e_j$. The link $e_j$ is then removed from $E_{MT}$.

$\text{Merge}(e_i, e_j)$ also changes the endpoint classes of $e_i$ according to Table 3.2, where the endpoint classes of $e_i$ and $e_j$ are matched to the first row and to the first column of the table, respectively. The classes are changed according to the hierarchy in Figure 3.3. The link with the most specialized endpoint (i.e. lower level in the hierarchy) determines the class of the resulting endpoint. We keep the endpoints as generic as possible during the merging operations in order to facilitate further merging.

Since links have two endpoints, there might be two alternatives to merge them. In most scenarios, one of these alternatives is generally ruled out by the classes of the endpoints, the coherence of paths, or the links sharing that endpoint. If none of these cases apply, iTop does not immediately commit to either way of combining them. This is done to avoid specifying the classes of link endpoints beyond what is implied by each merge. As a result, iTop has more freedom in what merges can be performed in later iterations. Only when no further merges can be carried out and there are still links for which both combinations are valid is one of the two ways chosen randomly.

Figures 3.5 (a-c) show three of the steps in the merging phase for the example shown in Figure 3.2. The virtual topology and the number of merging options for each link are depicted in Figure 3.5 (a). For the first merging steps, one of the links with a single merging option is picked and combined with its only available option. In this

31
example, (A, B1) is merged with (A, B2), (B1, C) is merged with (B2, C1), and (C, D1) is merged with (C1, D2). There is only one way in which the classes of the endpoints of these links can be combined, thus iTop commits to combining the endpoints. The resulting partially merged network and the classes of the merged nodes are shown in Figure 3.5 (b). Note that the endpoint classification prevents several erroneous merging from occurring. As an example, (A, B1) is not merged with (A, H1), since B1 and H1 have two incompatible classes, blocking and anonymous, and thus they must be two different nodes in the GT.

There are two possible merges remaining. (H1, I) is merged with (H2, I) and (D1, E) is merged with (D3, E). This results in the merged topology shown in Figure 3.5 (c), which correctly represents the ground truth topology depicted Figure 3.2 (a).

Note that depending on the order in which links with the same number of options are merged, iTop may infer a topology slightly different from the GT topology. Our results in Section 3.3 show that the topologies inferred by iTop closely match the ground truth, even with networks that are significantly larger than in this example.
Figure 3.5: Example $G_{MT}$ Topologies before (a), during (b),
and after (c) the Merging Process

33
3.3 Evaluation of Inferred Network Metrics

In order to confirm the effectiveness of iTop, we compare it to a selection of existing Traceroute-based inference algorithms. First we describe those other algorithms so that the differences in the networks they infer can be fully understood. Then we present the metrics for the topologies they each infer from both realistic and randomly-generated networks. By comparing these results, the advantage of the way iTop treats non-cooperative routers and merging of links becomes evident.

3.3.1 Comparison Approaches

Here we describe the approaches we consider for performance comparison with iTop. Merging Nodes (MN) [4] is a Traceroute-based approach to infer the network topology in the presence of anonymous routers. MN collects the path information between monitors and uses it to construct an initial induced topology. Similar to the virtual topology, the induced topology contains several duplicated components which are reduced by performing merging operations.

MN iteratively merges nodes in the induced topology. In each iteration it builds an equivalence class which is constructed by incrementally adding one node at a time, chosen from the nodes that have not yet been merged. A node is added into the class if it can be merged with every node already in the class. The iteration terminates when the class cannot be extended further, and all of the nodes it contains are merged into a single node in the resulting MN topology.
According to MN, two nodes can be merged only if they never appear in the same path and if the resulting topology will not shorten the minimum distance between any two nodes as observed in the induced topology.

Since MN is designed only for anonymous routers, the construction of the induced topology does not take into account the lack of information caused by blocking routers and therefore may result in a disconnected network.

Isomap [5] is another Traceroute-based approach, but it considers the presence of both anonymous and blocking routers. It constructs an initial that contains a virtual router for each anonymous router observed in the traces. Unlike iTop, no virtual router is added to the network when a blocking router is detected on a path. Instead a link is added between the last responding routers identified in the traces on either side of the unobserved section. Intuitively, Isomap's initial topology may underestimate the ground truth topology, as the portion of the network hidden by blocking routers is not considered at all.

Isomap has two merging phases: initial pruning and router merging. The initial pruning merges virtual nodes that share the same neighbors. In the router merge phase, each router is represented as a point in a multi-dimensional space using hop distance or round trip time as a distance metric. A mapping to a lower dimensional space is performed using the algorithm proposed in [23]. The merging process is governed by the setting of two thresholds, \( \Delta_1 \) and \( \Delta_2 \). Two nodes are merged together if in the lower dimensional mapping the distance between two virtual routers is less than \( \Delta_1 \) or if their distance is less than \( \Delta_2 \) and they share at least one common neighbor.
We refer the reader to [4] and [5] for more details about MN and Isomap, respectively.

### 3.3.2 Topology Inference Experiments

We compare the performance of iTop, MN, and Isomap through simulations on both realistic and random networks. In both cases we deploy 40 monitors as edge nodes randomly. We consider 10 monitor deployments for each type of network and average the obtained results.

We consider random and realistic starting networks. Given a starting network and a monitor placement, the GT topology is obtained from the union of the paths between monitors. In each GT topology, a fraction of the nodes are randomly designated as anonymous and blocking routers. On the basis of this assignment, MN, Isomap, and iTop determine the induced, initial and virtual topologies, respectively, which are used as input for the subsequent merging operations.

In our experiments we use hop distance as the distance metric for Isomap as the potential instability of round trip time may negatively affect merging, as described in [5]. For Isomap we reduce to a 5-dimensional space as in [5] and set the thresholds to $\Delta_1 = 10$ and $\Delta_2 = 4 \Delta_1$. Note that an optimal setting of the thresholds highly depends on the specific topology and monitor placement. We simulated several scenarios and selected a setting which performs well in the majority of the cases.
First we compare the merged topologies produced by iTop, MN, and Isomap for realistic topologies. We use the Autonomous System (AS) topologies from the CAIDA [24] project, which represents IP-level connections between backbone/gateway routers of several ASes from major Internet Service Providers (ISPs) around the globe. The starting CAIDA network from which the GT topologies are derived consists of 250 nodes and 290 edges.

We consider two scenarios regarding the types of routers which are present in the network. In the first scenario networks include both anonymous routers and blocking routers. This is a realistic setting as shown by our experiments in Section 2, and it is this case for which iTop and Isomap are designed. Since MN is only intended for anonymous routers, the second scenario has only responding and anonymous routers and does not include any blocking ones.

In Figures 3.6 (a-d) we show an instance of the GT topologies and the resulting topologies inferred by iTop, MN and Isomap. This portrays a case in which 10% of the routers in the network are anonymous and another 10% of the routers are blocking.

The most notable feature of the GT topology (Figure 3.6 (a)) is the presence of a central hub router which connects several branches of the network. The topology inferred by iTop (Figure 3.6 (b)) clearly reflects the existence of this hub and resembles the original topology. MN (Figure 3.6 (c)) significantly overestimates the network due to its notion of distance preservation, as preserving the distance between any two known nodes in the network prevents a significant number of merges. Furthermore, the inferred topology is disconnected because MN does not take into account the presence of blocking
routers when constructing the induced topology from the traces. Isomap (Figure 3.6 (d)) is able to perform more merges than MN, but the structure of the network is still significantly different from the GT topology. In addition, it underestimates several parts of the network, as shown by the branches that are shorter than in the GT topology. This is due to the fact that the construction of the induced topology uses only a single link to represent portions of a path hidden between two blocking routers.

Figure 3.6: Example Inferred Topologies with 10% Anonymous and 10% Blocking Routers for GT (a), iTop (b), MN (c), and Isomap (d).
Figures 3.7 (a) and (b) show the average number of nodes and links, respectively, in the inferred topologies as the fraction of non-cooperative nodes in the networks increases. We consider the number of non-cooperative nodes to be half anonymous and half blocking. These figures highlight that iTop outperforms other approaches, as the number of nodes and links are closer to the ground truth with iTop in all considered cases. MN and Isomap both overestimate the number of links because they perform merging operations on nodes whereas iTop merges links. This makes it less likely that MN and ISOMAP will combine two links, as both endpoints of the links must be merged together in order for the links to be merged. MN tends to achieve results closer to the GT topology as we increase the percentage of non-cooperative routers. This does not imply a better inferred topology. On the contrary, the increased number of blocking routers results in a larger portion of the network being hidden and therefore not present in the induced topology. As a consequence, as we increase the fraction of non-cooperative routers, the topology inferred by MN contains more disconnected components and each of those components is overestimated with respect to the GT topology. Isomap underestimates the number of nodes in the network as the initial topology does not contain the portions of the network hidden by blocking routers.
Figure 3.7: Realistic Networks with Anonymous and Blocking Routers: Number of Nodes (a), Number of Links (b), Cumulative Distribution of Node Degree (c), and Betweenness Centrality (d)

In Figure 3.7 (c) we show the cumulative distribution of node degree. Note that we show the absolute number of nodes on the y-axis in order to better highlight the differences in the inferred topologies. These differences may have been hidden by normalization. iTop infers a better topology which closely approximates the degree distribution of the GT topology. MN has more nodes with a higher degree, which are not present in the ground truth. Isomap underestimates the number of nodes, and as a result the cumulative distribution significantly deviates from the distribution for the GT topology. These results highlight the substantial structural differences between the topologies inferred by MN and Isomap with respect to the ground truth.
Figure 3.7 (d) shows the cumulative distribution of the normalized betweenness centrality. The normalized betweenness centrality of a node is calculated as the percentage of shortest paths in the network that go through that node. This metric indicates the degree of structural similarity of the inferred topologies with respect to the GT topology. As the figure shows, MN overestimates and Isomap underestimates the real distribution. This is a consequence of the respective overestimation and underestimation of the GT topology that they perform. In comparison, iTop closely matches the ground truth, highlighting structural similarity of its inferred topology to the GT topology. These results show that the topologies inferred by iTop are within 5% of the GT topologies with regards to all of the considered metrics.

In order to further compare the structure of the inferred topologies to the GT topology, we consider the Joint Degree Distribution (JDD), which has been recognized as a meaningful metric for topology comparison [25]. Given a network $G = (V, E)$, the JDD counts the number of links that connect a node with degree $x$ to a node of degree $y$ for $x, y \in [0, |V|-1]$. In Figures 3.8 (a-c) we show 3D representations of the relative JDD for the inferred topologies with respect to the JDD of the GT topology. Flatness in these diagrams indicates that the JDD of the inferred topology is close to that of the GT topology.
As depicted in Figure 3.8 (a), the topology inferred by iTop results in a flat relative JDD, once again indicating a good inference of the GT topology. Since MN is not able to correctly infer hub nodes, it has more nodes with a high degree connected to
nodes with low degree, as Figure 3.8 (b) shows. These nodes should have been merged together to correctly infer the hub. The JDD of the Isomap topology highlights the fact that this algorithm underestimates the nodes with low degree with respect to GT, as shown in Figure 3.8 (c). The presence of blocking routers causes several nodes located on the branches of the realistic topology to not be represented in the induced topology 3.6 (d). Furthermore, Isomap is not able to correctly infer the hub, so the inferred topology has more nodes with high degree than the GT does.

In the second scenario we consider the presence of only anonymous routers in order to study the performance of iTop in the setting for which MN is designed. Results are shown in Figures 3.9 (a-d). Although MN is designed to operate in this setting, it significantly overestimates the number of nodes and links in the network as shown in Figures 3.9 (a-b). MN ensures the shortest distances between any two nodes are preserved, thereby preventing many correct merges from occurring. In comparison, Isomap and iTop better infer the real topology, although Isomap performs some extra merging operations which result in a slightly lower number of nodes than are present in the GT topology.

Figures 3.9 (c-d) show the degree distribution and normalized betweenness centrality distribution, respectively. These metrics highlight that MN does not perform enough merging operations, resulting in an inferred topology that significantly differs from the GT topology. The distributions of iTop and Isomap better match the distribution of the GT topology, but Isomap slightly underestimates the ground truth.
In this scenario iTop still achieves results which are within 5% of the GT for all of the considered metrics. Note that Isomap performs better in this scenario because the absence of blocking routers enables the construction of a better initial topology that contains all nodes and links of the network. However, real scenarios are characterized by the presence of both anonymous and blocking routers, as shown by our experiments in Section 3.1 and the experiments in [8]. In these real contexts Isomap performs poorly as previously shown.
The simulated random networks are generated by ensuring that a single connected component is present. First a tree with a given number of nodes is randomly created and then additional links are added randomly to increase the network connectivity. The starting random network from which the GT topologies are derived contains 200 nodes and 350 edges before the addition of monitors.

We consider random networks in order to study the performance of the algorithms with GT topologies having significantly different structures than are present in the realistic networks. Due to space limitations we only consider the case with both blocking and anonymous routers. Results are shown in Figures 3.10 (a-d).

The random topologies are characterized by degree distributions that are more even than those of the realistic networks, as shown in Figure 3.10 (c). As a result, there is more variation in the links that the paths contain, and blocking routers cause a larger portion of the network to be unobservable. Since MN and Isomap do not specifically handle the effects of these blocking routers, the induced and initial topologies from which they start merging underestimate the random GT topologies more than they do realistic GT topologies. For this reason, MN underestimates the number of nodes and links (Figures 3.10 (a-b)). Isomap also underestimates the number of nodes, but it significantly overestimates the number of links because it performs merging operations on nodes. iTop outperforms the other two approaches as its inferred topology closely matches the number of nodes and links in the GT topology even as the number of non-cooperative routers increases.
Figure 3.10: Random Networks with Anonymous and Blocking Routers: Number of Nodes (a), Number of Links (b), Cumulative Distribution of Node Degree (c), and Betweenness Centrality (d).

Figures 3.10 (c-d) show the distribution of node degree and normalized betweenness centrality. In this case, iTop also outperforms MN and Isomap, showing a closer match to the GT topology for these distributions as well.

These results show that even with random networks the topologies inferred by iTop are within 5% of the GT topologies.
3.4 Fault Diagnosis with Partial Information

In this section we evaluate iTop, MN and Isomap by considering fault localization as an application of inferred topologies. Link failures are common in modern networks due to maintenance procedures, hardware malfunctions, energy outages, or disasters [26] and may cause degradation in performance. Fault localization techniques [2, 27] generally assume complete knowledge of the network topology. When this knowledge is not available, failures are diagnosed on the inferred topologies. Intuitively, a better inferred topology enables a fault localization algorithm to achieve performance closer to what is obtained when the full GT topology is known.

In this section, we consider the Max-Coverage (MC) algorithm [2], one of the most referenced approaches. We first describe MC and then introduce the performance metrics used to evaluate the output of MC in the context of inferred topologies. Finally, we present the experimental results.

3.4.1 Max-Coverage with Partial Information

MC is a greedy algorithm for diagnosing network link failures. Its input is the network topology, represented as a graph \( G = (V, E) \), and a set of symptoms \( S \). Symptoms represent disconnections between network monitor pairs. Each symptom represents a pair of monitors that cannot communicate because of link failures in the network.

The pseudo-code for MC is shown in Figure 3.11. The function \( \text{explained}(G, S, e) \) returns the set of symptoms in \( S \) that are explained by the failure of the link \( e \). A
symptom is explained by the failure of $e$ if such a failure causes the disconnection between the monitors that generated the symptom. The list $FL$ contains the links which MC has identified as the cause of the observed symptoms and is initialized as an empty set. In each iteration of the while loop, MC selects a link which explains the most unexplained symptoms and adds it to $FL$. In some cases there may be a tie between two or more links that all explain the most unexplained symptoms. When this occurs, MC randomly selects one of the tied links to add to $FL$. The loop terminates when all symptoms are explained by the links in $FL$ and then outputs the list.

![Figure 3.11: Max-Coverage Algorithm](image)

When MC is applied to an inferred topology, the resulting list $FL$ may contain links that do not correspond to only a single link in the GT topology. One or more links may represent one or multiple real links at the same time, due to possible erroneous inference. Therefore, we propose different definitions of basic performance metrics, such as accuracy and false positives, in order to evaluate the performance of MC in terms of real links. Additionally, we propose an additional new metric called redundancy.
Note that when using MC, if two or more links occur in the same set of paths in a network topology then they will always explain the same set of symptoms. We refer to these links as *indistinguishable* links. MC treats indistinguishable links by breaking the tie randomly and adding one of them to FL. This may result in an increase in the number of false positives and a decrease in accuracy.

The effect of indistinguishable links is much more noticeable in inferred topologies than the GT topology because one or more inferred links may represent one or multiple real links. Based on this observation, given a failed list FL we define an extended list FL\textsubscript{EXT} which includes all links that are indistinguishable with those in FL. We define $FL_{\text{EXT}} = \bigcup_{e \in FL} \{ q \in E \text{ s.t. } P(e) = P(q) \}$ where for a link e, $P(e)$ refers to the set of paths between monitors in which q occurs in the inferred topology.

In order to take into account the possible many-to-many relationship between links in FL\textsubscript{EXT} and the real links in the GT topology, we define the *hypothesis list*, $H = \{ Q, D \}$, as a multi-set. $Q$ is a set of real links that are represented by the links in FL\textsubscript{EXT}, and $D: Q \rightarrow \mathbb{N}$ gives the multiplicity for each link in $Q$. More formally, $Q = \bigcup_{e \in FL} rl(e)$ where $rl(e)$ is the set of real links represented by $e$ in the inferred topology. The function $D$ for a real link $q \in Q$ is defined as $D(q) = |\{ e \text{ s.t. } e \in FL_{\text{EXT}} \text{ and } q \in rl(e) \}|$. Based on these definitions, the size of the hypothesis list $H$ is $|H| = \sum_{q \in Q} D(q)$.

Given the set of Actual Failed Links $AFL \subseteq E_{GT}$, we define the following performance metrics for accuracy, redundancy, and false positives for hypothesis lists containing inferred links.
Accuracy (ACC) = \frac{|Q \cap AFL|}{|AFL|}

Accuracy represents the fraction of real links that fail and are represented by at least one link of the inferred topology in the output of MC.

Redundancy (RED) = \sum_{q \in Q \cap AFL} (D(q) - 1)

Some of the real links in AFL may appear multiple times in the output of MC for an inferred topology. The redundancy metric measures the number of redundant times failed links are reported in MC.

False Positives (FP) = \sum_{q \in Q \setminus AFL} D(q)

False positives count the number of links which have not failed, but are present in the output of MC.

According to the above definitions, it can be seen that \(|H| = ACC \cdot |AFL| + RED + FP|.

3.4.2 Fault Diagnosis Experiments

To test the performance of the inferred topologies with MC, we simulate link failures on the realistic network shown in Figure 3.6 (a). We consider a scenario with 10% anonymous and 10% blocking routers. The topologies inferred by iTop, MN and Isomap are shown in Figures 3.6 (b), (c) and (d), respectively.

We consider two failure scenarios. In the first scenario, we randomly fail an unobservable link in the ground truth topology. An unobservable link is one for which at
least one endpoint is a non-cooperative router. The unobservable nature of the failed links makes this scenario particularly meaningful for the performance of failure localization algorithms, as additional probing cannot be used to confirm and refine the returned hypothesis list. In the second scenario, we randomly fail a generic link in the network, no matter the classes of its endpoints. For each scenario we averaged the results over 100 trials.

Figure 3.12 shows the accuracy of MC when applied to GT and the inferred topologies. The good inference provided by iTop is reflected in the accuracy achieved by MC. iTop outperforms the other approaches and achieves an accuracy close to that of GT. MN in particular suffers in the unobserved scenario as blocking routers cause only a fraction of the unobserved links to appear in the initial topology of MN. Isomap does not provide a sufficiently precise estimation of the ground truth to enable MC to correctly localize the failure, resulting in a lower accuracy.

![Figure 3.12: Average Accuracy of MC with GT and Inferred Topologies.](image)
The number of false positives is shown in Figure 3.13. iTop shows only a slight increase in the number of false positives with respect to GT, highlighting that our approach enables MC to provide a hypothesis list that only contains a few links erroneously reported as failed. MN shows a lower number of false positives, but this is highly counterbalanced by the low accuracy that this approach provides, as Figure 3.12 shows. Isomap incurs a high number of false positives. This is due to the construction of the induced topology, which represents any portion of the network included between two blocking routers with a single link. This aggregate representation enables MC to only identify failures at a coarse granularity, resulting in a large number of false positives.

Figure 3.13: Average False Positives of MC with GT and Inferred Topologies.

Figure 3.14 shows the redundancy, as defined in Section 3.4.1, normalized by the accuracy. This metric represents the amount of redundant information reported by MC when the algorithm correctly reports a failed link. Note that this metric is meaningful only for inferred topologies, as redundancy is caused by multiple representations of the same link. The plot reaffirms that the topologies inferred by MN are inappropriate for
failure localization. The hypothesis list contains a large amount of redundant information for the few times that MC is able to localize the failed link. In comparison, iTop and Isomap show low redundancy. In particular, the correctness of the merging process of iTop limits multiple representations of the same link in the inferred topology, resulting in the lowest redundancy.

![Average Redundancy of MC with GT and Inferred Topologies.](image)

These results show that not only does iTop infer topologies which closely match the ground truth, but it can also be effectively used in real network management applications such as failure localization.
Chapter 4

Modeling Probabilistic Cascading Failures

While networks are often considered separately in order to analyze their inner workings, in the real world many networks are interconnected and rely upon each other. The most popular example is that of power grid and Internet, where electricity is required for routing hardware and that hardware is responsible for transmitting the data used to balance the load of electricity, but it is far from the only one [12,18]. Examining an individual network in the context of an interdependent network reveals additional insights and weaknesses about its operation. When considering a widespread natural or man-made disaster affecting the Internet, the routers outside the area may be assumed to be unharmed when considering only that network. However, many of those routers may receive electricity from a power plant located inside the area. As a result they would not be functioning, but this is only made obvious when examining the situation as an interdependent network.

The interesting effects of interdependent networks are due to the inter-edges, or connections between the individual networks. These inter-edges generally involve some notion of spatial locality, as they represent physical connections and reliance between the networks [20]. Another factor is the size or importance of the nodes within a network, such as a nuclear power plant generating more electricity than a coal plant and therefore having more nodes relying upon it for power [13, 14, 16]. While other variables may also
influence which nodes are connected by inter-edges, these two factors are the most consistent across all types of interdependent networks.

The shared reliability inherent in interdependent networks make them important when considering the occurrence and spread of failures. As described above, a node in an interdependent network can fail even if the entirety of its own network is still functioning. This makes nodes reliant upon other networks which may have a different notion of robustness. A highly robust node in one network which is dependent upon a failure-prone node in the other network no longer remains robust when considered in the context of the interdependent network [18]. This reveals vulnerabilities in the networks that were not present before. The 2003 blackout in Italy provides a real example of a cascading failure resulting from interdependent networks [20]. The original failure occurred in the power grid and may not have spread any further. However, the lack of electricity led to failures in Internet routing, which then prevented effective load balancing and caused additional failures in the power grid. From there the failure continued to propagate both within the individual networks and between them until total failure was achieved.

These types of failures can also occur within an individual network, and much work has examined how to model cascading failures for the power grid and other types of networks [12, 13, 14]. While some of these consider failures deterministically [19, 28], others have assigned probabilities to the chance of propagation based upon dependencies within the network [15]. In general, these models tend to minimize the impact of time in the spread of failures. In many cases the models are solely concerned with estimating which nodes will have failed once the cascade has ceased regardless of when they failed.
While other models do include a time aspect to the failures, they only consider that the failure can spread from a failed node regardless of how long it has been since the node went down [16, 17].

In this chapter we introduce and describe our model for the probabilistic representation of the spread of cascading failures as they propagate through a pair of interdependent networks over time. For simplicity we focus on the case in which there are two separate networks, but this model can easily be extended to three or more. We consider that failures may have a delay before they propagate, thereby allowing the cascade to cease after a certain period of time has elapsed. Through this model we are able to calculate the times at which each node in the network is expected to fail and thereby the expected size of the cascade of failures as well.

4.1 Propagation Model

We model the interdependencies between two separate networks $G_1 = <V_1, E_1>$ and $G_2 = <V_2, E_2>$ where $V_1$ and $V_2$ are the sets of nodes in the two networks and $E_1$ and $E_2$ are the sets of links connecting the nodes in their respective networks. $G_1$ and $G_2$ each represent a network infrastructure that is traditionally considered to be self-contained, such as the power grid or telecommunications network. These networks are connected by a set of directed inter-edges $E_{Int}$ in order to construct the full interdependent network $G_{Int} = <V_{Int}, E_{Int}>$, where $V_{Int} = V_1 \cup V_2$. $E_{Int}$ represents the interdependencies between $G_1$ and $G_2$ and is defined as $E_{Int} \subseteq (V_1 \times V_2) \cup (V_2 \times V_1)$. 
Each inter-edge indicates a node in one network that could potentially cause a node in the other network to fail, thereby propagating the cascade. Failures are modeled as propagating in discrete time steps. For the inter-edge \((u, v)\) from parent node \(u\) to child node \(v\), \(P_{u,v}(t)\) denotes the probability that the failure propagates from \(u\) to \(v\) exactly \(t\) time steps after \(u\) has failed. The values of this function at specific time steps can be varied in order to model different types of failure behavior, such as a high degree of failure immediately after \(u\) fails or the chance of failure steadily increasing as time goes on. Note that \(\sum_{t=0}^{\infty} P_{u,v}(t)\) may be less than 1 if there is some chance that \(v\) can survive the failure of \(u\). We define the set of parent nodes of node \(v\) as \(I(v) = \{u | (u,v) \in E_{\text{int}}\}\).

Given an initial failure scenario \(F_0\), which defines the set of nodes having experienced a failure at time \(t=0\), this model calculates the likelihood that each node in the network has failed for times \(t \geq 0\). This allows for calculation of the expected scope of the cascade over time as well as identifying when nodes of importance reach certain thresholds of failure chance. Given a node \(v\), the probability that it fails at time \(t\) is indicated by \(P_v(t)\). Note that for \(v \subseteq F_0\), \(P_v(0) = 1\) and \(P_v(t) = 0\) for \(t > 0\). The exact calculation of \(P_v(t)\) for any nodes not in \(F_0\) changes based on the relationship between the nodes on which its parent nodes depend. We say that a node \(x\) depends on another node \(y\), or \(y \in D(x)\), if there is a set of inter-edges in \(E_{\text{int}}\) that form a path from any node in \(F_0\) to \(x\) and pass through \(y\). This means that the probability that \(x\) has failed is directly affected by whether or not the cascade has propagated to \(y\) yet.

Based on these dependencies, there are three ways in which to best calculate \(P_v(t)\) in order to capture the effects of dependence in the network. Below we detail these three
cases in order to describe when they are most appropriate to use and what information about the network they take into account.

4.1.1 Case 1 - Disjoint Parent Nodes

In the first case, all of the parent nodes of \( v \) are dependent on disjoint sets of nodes excluding those in \( F_0 \), or \( D(u_1) \cap D(u_2) \subseteq F_0 \ \forall \ u_1, u_2 \in I(v) \) s.t. \( u_1 \neq u_2 \). This means that \( \forall u \in I(v) \), \( P_u(t) \) is independent given the initial failures. An example of this independence can be seen in Figure 4.1. In the example, node \( v \) depends on both \( w \) and \( u \). Node \( w \) depends on \( x \) and \( u \) depends on \( y \). Since \( F_0 \) is the initial failure, \( P_w(t) \) and \( P_u(t) \) can be calculated independently.

![Figure 4.1: Independent Parent Nodes Example](image-url)
When the nodes in \( I(v) \) have independent causes of failure, \( P_v(t) \) can be calculated using the following formula:

\[
P_v(t) = \left( \prod_{u \in I(v)} (1 - \sum_{t'=0}^{t-1} P_{u,v}(t - 1 - t') P_u(t')) \right) - \left( \prod_{u \in I(v)} (1 - \sum_{t'=0}^{t} P_{u,v}(t - t') P_u(t')) \right)
\]

In this formula the \( P_{u,v}(t-t')P_u(t') \) term calculates the probability that node \( u \) failed at time \( t' \) and propagated the failure to \( v \) at time \( t \). The summation from \( t'=0 \) to \( t-1 \) or \( t \) adds these terms together to determine whether the failure propagated to \( v \) at time \( t-1 \) or \( t \), respectively, given that \( u \) failed anywhere at any time before \( v \) did. Subtracting these sums from 1 gives the complement scenario, that node \( u \) did not propagate the failure to node \( v \) within the considered time period. This may be due to either \( u \) not failing or failing but not propagating the failure along the inter-edge.

The first product in the formula therefore calculates the probability that none of the nodes in \( I(v) \) have propagated the failure to \( v \) at or before time \( t-1 \). The second product calculates that none of those nodes have propagated the failure at or before time \( t \). The difference between these two products is the probability that \( v \) was functional at time \( t-1 \) but not functional at time \( t \). This leaves only the probability that \( v \) failed at exactly time \( t \).
4.1.2 Case 2 - Shared Dependencies in Parent Nodes

In the second case some of the nodes in $I(v)$ may be dependent on the same nodes, or $D(u_1) \cap D(u_2) \not\subseteq F_0 \exists u_1, u_2 \in I(v)$ s.t. $u_1 \neq u_2$, but the inter-edges do not form any loops. Due to this relaxation, the probabilities $P_u(t)$ for $u \in I(v)$ are no longer all independent. The probabilities of multiple nodes in $I(v)$ having failed are dependent upon whether or not a given node closer to the initial failures has failed. An example of this type of shared dependency is shown in Figure 4.2, in which nodes $w$ and $u$ are both dependent up on node $x$. If $x$ has not failed at time $t$ then neither $w$ nor $u$ can have failed at time $t$.

![Figure 4.2: Shared Dependencies Example](image-url)
Since the probabilities of the parent nodes failing are no longer independent, separate scenarios must be considered based on the possible failure times of these shared dependencies. We generate the set $S$ of all possible dependence scenarios, where each scenario is an assignment of failure times to each shared dependency $s_i$ from $t' = 0$ to $t'+1$ of the form $(s_1 = t_1, s_2 = t_2, \ldots, s_n = t_n)$ to denote that the shared dependency node $s_1$ failed at $t_1$, node $s_2$ failed at $t_2$, and so on. Since $P_v(t)$ only considers times from 0 to $t$, assigning a failure time of $t+1$ effectively treats the node as still being operational as $P_{u,v}(t) = 0$ for $t<0$. Based on these scenarios, we calculate $P_v(t)$ as follows:

$$P_v(t) = \sum_{s \in S} P(s)(\prod_{u \in I(v)}(1 - \sum_{t' = 0}^{t-1} P_{u,v}(t - t')P_u(t'|s))) - (\prod_{u \in I(v)}(1 - \sum_{t' = 0}^{t} P_{u,v}(t - t')P_u(t'|s)))$$

where $P(s) = P_{s_1}(t_1)P_{s_2}(t_2|s_1=t_1)\ldots P_{s_n}(t_n|s_1=t_1,\ldots,s_{n-1}=t_{n-1})$

The chain of probabilities $P_{s_1}(t_1)\ldots P_{s_n}(t_n|s_1=t_1,\ldots,s_{n-1}=t_{n-1})$ calculates the probability of the dependence scenario occurring in which each node in $s$ occurs at the defined time. The probabilities $P_u(t'|s)$ are independent because these scenarios assign failure times to all of the nodes on which more than one member of $I(v)$ depends. These dependence scenarios are disjoint and consider all possible failure times for the shared dependencies, so summing up the probability that the failure propagates to $v$ at exactly time $t$ across all scenarios results in the full probability $P_v(t)$. Note that Case 1 is a subset of Case 2 in which $S$ is an empty set. The formula for Case 1 is therefore a simplification of the one for Case 2 with $S$ removed.

The formula below for $P_v(t|s')$ calculates the probability that $v$ fails at time $t$ given a set of established failure times $s'$. If a failure time for $v$ is given in $s'$ then $P_v(t|s') = 1$ if the failure time is $t$ and 0 otherwise. If neither v nor any node it depends on have a failure
time in $s'$ then it does not affect the probability of failure and $P_v(t|s') = P_v(t)$. Otherwise, failure scenarios in $S$ are only relevant if all of the defined failure times agree with those defined in $s'$. $P(s|s') = 0$ if $s'$ defines a failure time for a node that is different from the failure time it has in $s$. When calculating the probabilities of failure for all parents of $v$, all given failure times must be considered, hence the change to $P_u(t|s' \cup s')$.

$$P_v(t|s') = \sum_{s \in S} P(s|s')(\prod_{u \in I(v)} (1 - \sum_{t' = 0}^{t-1} P_{u,v}(t - t') P_u(t'|s \cup s'))) - (\prod_{u \in I(v)} (1 - \sum_{t' = 0}^{t} P_{u,v}(t - t') P_u(t'|s \cup s')))$$

where $P(s|s') = P_{s_1}(t_1|s') P_{s_2}(t_2|s',s_1=t_1) \ldots P_{s_n}(t_n|s',s_1=t_1,\ldots,s_{n-1}=t_{n-1})$

### 4.1.3 Case 3 - Inter-Edge Loops

The last case addresses networks in which the inter-edges form loops containing a subset of the nodes in $V_{\text{int}}$. An example of this case is shown in Figure 4.3. When calculating $P_v(t)$, the value for $P_u(t)$ will be calculated as well. However, calculating $P_u(t)$ requires $P_v(t)$ as well. In this case the formulas from Cases 1 and 2 cannot be applied because each node in a loop will be dependent on itself.
For the example in Figure 4.3, when calculating $P_r(t)$ we are concerned with the probabilities that nodes $u$ and $y$ have failed given that $v$ has still yet to fail. If $v$ has already failed then calculating $P_r(t)$ is unnecessary. Therefore we can add an additional failure constraint $v = t+1$ to the set of givens in order to calculate the probability that $u$ or $y$ cause the failure of $v$. This results in the formula shown below:

$$P_r(t) = \sum_{s \in S} P(s)\left(\prod_{u \in I(v)}(1 - \sum_{t'=0}^{t-1} P_{u,v}(t - t')P_u(t'|s, v = t + 1))\right) - \left(\prod_{u \in I(v)}(1 - \sum_{t'=0}^{t} P_{u,v}(t - t')P_u(t'|s, v = t + 1))\right)$$

Note that the formula for Case 2 is a simplification of this formula just as the formula for Case 1 is a simplification of the one for Case 2. In this case we add $v = t+1$ to the set of given failure times $s$ that is passed when calculating the probability that $u$ has failed. If $u$ depends on $v$, this will result in $P_{u,v}(t'-t)P_u(t'|s, v = t+1)$ representing the event in
which \( u \) propagates the failure to \( v \) when \( v \) has not yet failed. If \( u \) does not depend on \( v \) then the additional given doesn't affect the calculation of \( P_u(t'|s) \) and it will return the same value as it would using the formula in Case 2. The same changes is made to the formula for \( P_v(t|s') \), as shown below.

\[
P_v(t|s') = \sum_{s\in S} P(s|s')(\prod_{u\in I(v)}(1 - \sum_{t'=0}^{t-1} P_{u,v}(t - t')P_u(t'|s \cup s', v = t + 1)))
- (\prod_{u\in I(v)}(1 - \sum_{t'=0}^{t} P_{u,v}(t - t')P_u(t'|s \cup s', v = t + 1)))
\]

### 4.2 Model Verification

In this section, we evaluate the accuracy of the model by comparing the failures it predicts to simulations of failure propagation for a collection of interdependent networks. These networks have vastly different structures and we vary the probabilities of inter-edge propagation in order to show that the model closely predicts the size of the cascade regardless of the specific parameters of the network and failures.

#### 4.2.1 Experiment Setup

We consider failure scenarios which consist of a single initial failure and compute the probabilities of each node in the interdependent network failing for the next 300 time steps, which is long enough for the cascade to either cease or fully spread through the network. The probabilities of propagation along the inter-edges are set to fail with a uniform distribution over the 20 time steps following the failure of the parent node, so \( P_{u,v}(1) = P_{u,v}(2) = \ldots = P_{u,v}(20) \). The total chance of propagation is varied across the
experiments and set to sum from 0.1 to 1.0 in increments of 0.1, that is \( \sum_{t=1}^{20} P_{u,v}(t) = 0.1, 0.2, \ldots, 1.0 \) and \( P_{u,v}(t) = 0 \) for \( t=0 \) and \( t > 20 \).

We consider three different interdependent network structures. In the first structure, each of the individual networks consists of a hub-centric topology of 150 nodes in which 8 nodes are designated as hubs with high degree. 188 edges are added randomly between nodes in the first network and 197 edges are added randomly between nodes in the second network, with a 40% chance of connecting a node to one of the 8 hubs and a 60% chance of connecting two non-hub nodes. 375 inter-edges are randomly added to connect the two networks. The probability of a node \( v \) being selected as an inter-edge endpoint is \( \frac{d(v)}{\sum_{u \in E} d(u)} \), where \( d(v) \) is the degree of \( v \). This results in a network of inter-edges in which hubs are highly connected to nodes in the other network. Other nodes have few inter-edges and are often only connected to hubs. Figure 4.4 shows the structure of one of the individual networks. The large black nodes are the hubs.

The second network structure consists of two networks, each containing 10 clusters. Each cluster consists of 10 to 15 highly connected nodes, for a total of 127 nodes in the first network and 135 in the second. Most of the links in the network connect nodes in the same cluster, and few links connect nodes in different clusters. Each cluster is paired with a cluster in the other network. 275 inter-edges are added randomly with a 90% chance of connecting nodes in paired clusters and a 10% chance of connecting nodes in two clusters that are not paired. This creates a scenario in which failures cascade quickly within two paired clusters but propagation between two unpaired clusters occurs much less frequently. One of the two cluster networks is shown in Figure 4.5.
Figure 4.4: Hub-Centric Topology

Figure 4.5: Cluster Topology
The last interdependent network contains two realistic networks consisting of Sprint’s Rocketfuel AS Topology [6]. Each network contains 134 nodes with latitude-longitude coordinates and 193 edges connecting them. 335 inter-edges are randomly added one at a time, with each edge \((u, v)\) having probability

\[
\frac{1}{d(u, v) \sum_{u' \in V_1, v' \in V_2} \frac{1}{d(u', v')}}
\]

of being added, where \(d(u, v)\) is the geographic distance between \(u\) and \(v\). Nodes with the same coordinates are always connected. This creates a scenario in which failures tend to propagate over shorter distances due to more inter-edges being between nodes that are close to each other. Propagation over long distances is more infrequent due to there being fewer inter-edges that connect distant nodes. The Rocketfuel topology is shown in Figure 4.6.
For each of the three interdependent network structures we select one node to be the initial failure $F_0$. We then calculate the likelihood of each node in the network failing for $t=0$ to $t=200$ with inter-edge total propagation rates $\sum_{t=1}^{20} P_{u,v}(t)$ from 0.1 to 1.0 as previously described. These likelihoods are compared to the average failure times for the nodes based on simulations of the cascade using the same initial failure and inter-edge propagation rates. When a node fails in the simulation it spreads the failure through each of the inter-edges for which it is the parent node with probability $\sum_{t=1}^{20} P_{u,v}(t)$. If an inter-edge is chosen to propagate the failure then the child node will fail in the next 1 to 20 time steps, chosen by a uniform random distribution. For each network structure and propagation rate we average the simulated failure times of each node in the network over 500 runs and compare the averages to the estimated failure times calculated by the model.

### 4.2.2 Failure Propagation Experiments

Here we show the results for the model and simulations with the hub-centric topology. In this topology $G_1$ and $G_2$ each have 150 nodes, including 8 hubs, and 188 and 197 edges, respectively. They are connected by 375 inter-edges for an average inter-edge node degree of 2.5. Figure 4.7 shows the total number of failures estimated and simulated at each time step for total propagation rates 0.5, 0.8, and 1.0. Only these three propagation rates are shown here because they highlight an interesting range of cascade sizes and patterns. The failures for all propagation rates $\geq 0.5$ can be viewed in Appendix 1, as failures for lower rates are nearly negligible. For all propagation rates the number of
failures estimated at each time step is within 5% of the average number of simulated failures. This difference is due to a combination of variance in the simulation failure times and rounding made in the implementation of the model in order to speed up execution when dealing with very small probability values. The largest difference between the model and the simulation occurs towards the end of the cascade after time step 100. Nodes likely to fail in this period of time are furthest from the initial failure and as a result have a wider range of times in which they are able to fail due to variance in failure times for the nodes they depend on. The simulation fails almost all nodes in both networks by time step 200, but the model predicts fewer nodes failing from time steps 100 onwards. The model never has more than a 5% error in the total number of failed nodes, although it does not expect that all nodes will fail until time step 280.

Figure 4.7: Selected Results for Hub-Centric Topology
Figures 4.8 and 4.9 show the same selection of results for the clustered and Rocketfuel topologies, respectively. In the clustered topology the individual networks each consist of 10 clusters with a total of 127 nodes connected by 192 links for G1 and 135 nodes connected by 206 links for G2. There are 275 inter-edges connecting the two nodes. The individual networks in the Rocketfuel topology are identical and each consist of 134 nodes and 193 links. They are connected by 335 inter-edges. The full results for these interdependent networks are also shown in Appendix 2 and 3. The differences between the number of failures predicted by the model and simulation are very similar for these two networks. In both networks the model estimates at worst 2% less failures than the simulation average for any time step. Like with the hub-centric topologies, the greatest difference occurs during the time period in which the cascade is reaching the nodes furthest from the initial failure. However, the amount of error is smaller than with the hub-centric topologies because the time period is longer and fewer nodes are failing, thereby reducing the variance. Overall these results show that the propagation model is able to closely calculate the expected spread of the cascade at each time step with only a small margin of error.
Figure 4.8: Selected Results for Cluster Topology

Figure 4.9: Selected Results for Rocketfuel Topology
Of particular note, the failure propagates through the Rocketfuel topology in two main phases. This is noticeable in Figure 4.9 through the pair of first sharp increase in failures between time steps 25 and 125 and then the second, larger cascade from time step 125 onward. This is due to the spatial locality used in the connections in this interdependent network. The first cascade propagates through the nodes close to the initial failure, and then the second follows when the failure is finally able to spread to nodes further away. This pattern is different than the propagation in the Cluster topology for a few reasons. The geographical locations of the nodes separate them into loose groups based on continent. These groups contain more nodes than the clusters in the Cluster topology, but are also more sparsely connected. It therefore takes longer for the failure to reach a node that is connected to nodes in other groups than it does to spread to a node connected other clusters. This separates the failures in the local cascade from those in the larger cascade throughout the entire network.

This selection of total propagation rates highlights the vast increase in the total size of the cascade as the propagation rate increases. At \( \sum_{t=1}^{20} P_{uv}(t) = 0.5 \), less than 5% of the nodes in the network are expected to fail regardless of the network structure. Each of the nodes directly connected to the initial failure has a 50% chance of failing, but the cascade quickly stabilizes as each successive node is more likely to remain functioning.

When \( \sum_{t=1}^{20} P_{uv}(t) \) increases to 0.8, the cascade size increases to 25% of the maximum size for the hub-centric topology and 15% for the cluster and Rocketfuel topology. While these are still a small portion of the network, they demonstrate a significant increase in the size of the cascade. The high degree hubs in the hub-centric topology lead to a greater number of nodes that are expected to fail for the same
propagation rate than for the other two networks. Also, note that at the 0.8 propagation rate the secondary cascade for the Rocketfuel is not as nearly as significant because the cascade is expected to stabilize before it spreads beyond the local nodes.

These results also give insight into how the structure of the interdependent networks affects the speed at which the cascade propagates. As can be seen in Figures 4.7-9 for total propagation rate 1.0, the number of failed nodes begins to quickly increase between time steps 20 and 40. For the hub-centric topology, the size of the cascade increases sharply and propagates through 90% of the network by time step 150. This quick start to the cascade and fast speed with which it propagates is due to the highly connected nature of the hubs. Since most nodes in the network are connected to one or more hubs, it is almost guaranteed that at least one hub will fail within the first 20 time steps. This gives the cascade direct access to a large number of nodes to fail due to the high degree of the hubs.

In comparison, the interdependent networks with clustered and Rocketfuel topologies take until at least time step 200 to fully fail despite containing fewer nodes. This is due to the local nature of the inter-edges and more evenly distributed degree of the nodes. For the clustered topologies, the speed at which the cascade can spread is limited by the sparse inter-edges connecting non-paired clusters. All of the nodes in a cluster quickly fail as soon as one member of the cluster or its pair fails, but it takes longer for the failure to propagate to other clusters in either network. This results in the cascade spreading through the network at a slower rate.
This slower speed of failure spread is also visible in the Rocketfuel topology. The cascade is limited by geographical distance since the inter-edges are weighted towards connecting nodes that are in close proximity. This creates a more even distribution of node degree, and therefore a slower rate of failure than in the hub-centric network since there are no nodes of very high degree which can cause a large number of nodes to fail. Additionally, it is rare that the failure will spread to a node that is geographically distant and therefore likely to have a much different set of nodes to which it can propagate the failure.

As we have shown, our model is able to closely estimate the size of the cascading failure between two interdependent networks. It is able to match the average simulated cascade size to within 5% regardless of the structure of the individual networks and the manner in which they are connected with inter-edges. We have also explained how these structures and inter-edges affect the spread of the cascade through the network.
Chapter 5

Conclusions

Many operations and tasks for identifying and repairing network outages require knowledge of the network topology, which may not always be available. In this thesis we describe an algorithm for topology inference when only partial information about the network structure is available and show how a common fault diagnosis algorithm operates on the networks that it infers. We then continue to examine the issue of network failures by proposing a probabilistic model to estimate cascading failures as they spread over time.

We first addressed the problem of incomplete topology information by describing the iTop algorithm in Chapter 3. This algorithm infers the areas of the network topology that do not respond to the Traceroute protocol as expected by constructing a Virtual Topology which adds in the missing network components based on end-to-end path lengths between pairs of monitor nodes. It then applies a set of rules to the links that have been added in order to determine which of them may represent the same links in the real topology. Lastly, iTop iteratively merges links from the Virtual Topology in order to build an inferred network that estimates the true topology. We applied iTop to both random and realistic networks and showed that it infers topologies that have network metrics within 5% of the ground truth, thereby outperforming the results obtained by
using existing inference techniques. We then further showed that the networks inferred by iTop are more suitable for use as input to a common fault diagnosis algorithm.

In Chapter 4 we proposed a probabilistic model for estimating the expected failure times of nodes in an interdependent network when a set of initial failures begins to evolve into a cascade. We presented formulas for determining a node's expected chance of failure at each time step based on the probabilities of the failure propagating over the network's inter-edges. The accuracy of the model was demonstrated by comparing its predicted failure times to a series of simulations performed over a selection of interdependent networks. At any point in time, the size of the cascade predicted by the model is within 5% of the average size as determined by the simulation. An analysis of the propagation through each interdependent network structure provided insight into how network connectivity affects the cascade’s rate of spread.

In the future, this model can also be used for decision making when choosing a set of intervention strategies to enact in order to delay or halt the spread of failures. A selection of operations can be defined which represent real world actions such as providing a routing hub with a battery backup or preemptively shifting load away from a node in the power grid. Each of these operations has some amount of time required to implement them and effect on the propagation rates of the inter-edges connected to the affected nodes once completed. These changes may shift the current failure probabilities so that they occur in later time steps, lessen the probabilities at each time step, or otherwise affect the distribution of the $P_{u,v}(t)$ values in any way that accurately reflects the real-world action taking place. These changes to the probability distributions also affect the expected failure probabilities calculated by the model, thereby providing a view
of how performing the operation will affect the spread of the cascade and its size once it terminates. Certain operations on specific nodes will clearly be more advantageous based on how they affect the cascade. However, due to the time required to implement an operation it may not be guaranteed that the optimal set of actions can be completed before the failure of the nodes involved. This presents an interesting problem in which the optimal intervention strategy may require performing operations on nodes that weakly affect the cascade in order to delay later failures long enough for the implementation of other operations that more strongly impact the spread.

Additionally, topology inference can be combined with the cascading propagation model in order to evaluate the accuracy of the predicted failure times when some portions of the interdependent network must be inferred. If the inference is not accurate enough then the failure may propagate in unexpected ways and be more severe than predicted. If used with the implementation of intervention strategies, this could lead to suboptimal actions being performed due to the importance of some nodes being estimated incorrectly. This use for inferred topologies therefore emphasizes accurate inference of the most vital portions of the network.
References


Appendix: Full Propagation Results

Cascade Size with Hub Topologies

No. of Nodes Failed

Time

Est-0.5 ▲
Sim-0.5 □
Est-0.6 ▤
Sim-0.6 □
Est-0.7 □
Sim-0.7 □
Est-0.8 □
Sim-0.8 □
Est-0.9 □
Sim-0.9 □
Est-1.0 □
Sim-1.0 □
Cascade Size with Rocketfuel Topologies

No. of Nodes Failed

Time

Est-0.5
Sim-0.5
Est-0.6
Sim-0.6
Est-0.7
Sim-0.7
Est-0.8
Sim-0.8
Est-0.9
Sim-0.9
Est-1.0
Sim-1.0