UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

CAUSAL FAILURES AND COST-EFFECTIVE EDGE AUGMENTATION IN NETWORKS

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

By

ZUYUAN ZHANG
Norman, Oklahoma
2023

CAUSAL FAILURES AND COST-EFFECTIVE EDGE AUGMENTATION IN NETWORKS

A DISSERTATION APPROVED FOR THE
SCHOOL OF COMPUTER SCIENCE

BY THE COMMITTEE CONSISTING OF

Dr. Sridhar Radhakrishnan(Chair)

Dr. Kash Barker

Dr. Sina Khanmohammadi

Dr. Qi Cheng

Acknowledgments

I really appreciate these four years at the University of Oklahoma, working as a graduate teaching/research assistant with so many fabulous people!

I would like to thank Prof. Sridhar Radhakrishnan for being my committee chair and mentor. His guidance and tireless persistence helped me finish this work. I couldn't have had these achievements without his help. He also taught me how to communicate with people professionally and respectfully.

I also want to thank my parents for their endless love and support throughout this journey. Without them, I would have given up several times. It is their support that encourages me to realize my dream. Moreover, I am grateful to my grandparents for taking care of me when I was a child. In my mind, my grandmother is always the young lady who was always waiting in front of the school gate and riding her bike to take me home!

Next, I would like to thank Prof. Kash Barker. He is an excellent professor from the School of Industrial and Systems Engineering. I am grateful for his efforts on the ideas of my research, the wording of my papers, and efficient communication. In addition, I would like to thank Prof. Qi Cheng, and Prof. Sina Khanmohammadi on my committee for their time and advice.

Then, I want to thank Prof. Changwook Kim and Prof. Dimitrios Diochnos for the teaching assistant work on the Theory of Computation. The training helps me improve my English.

Specifically, I want to thank my friends, Mr. Justin Reynolds and Ms. Alison Smith. We often get together to talk about interesting things that happen in this country and I learn so much about the culture, beliefs, and living style of the US from them. I am also grateful for my wonderful colleagues, Dr. Sudhindra Gopal Krishna, Dr. Aditya Narasimhan, Mr. Reza Gheibi, and Mr. Ashesh Gaur. I really miss the time I worked with them and the laughter we had when playing with each other.

Finally, I would like to thank everyone who loves me, who hates me, whom I love, whom I hate. This PhD is not the end of my life, but a new start of my career. I will take the fortune I got and be ready to accept new challenges in the future! Hope everyone's life is going smoothly and happy forever!

Table of Contents

List Of Tables

List Of Figures

Abstract

Node failures have a terrible effect on the connectivity of the network. In traditional models, the failures of nodes affect their neighbors and may further trigger the failures of their neighbors, and so on. However, it is also possible that node failures would indirectly cause the failure of nodes that are not adjacent to the failed one. In a power grid, generators share the load. Failure of one generator induces extra load on other generators in the network, which could further trigger their failures. We call such failures causal failures. In this dissertation, we consider the impact of causal failures on multiple aspects of one network. More specifically, we list the content as follows.

- In Chapter 1, we introduce basic concepts of networks and graphs, classical models of failures and formally define causal failures in a given network.

- Chapter 2 addresses the network's robustness and aims to find the maximum number of causal failures while maintaining a connected component with a size of at least a given integer. More specifically, we are looking into the number of causal node failures we can tolerate yet have most of the system connected with α being used to parametrize.

- Chapter 3 deals with vulnerability, wherein we aim to find the minimum number of causal failures such that there are at least k connected components remaining. We are looking for the set of causal failures that will result in the network being disconnected into k or more components.

- In Chapter 4, we consider causal node failures occurring in a cascading manner. Cascading causal node failures affect communication within nodes, which is dependent on the paths that connect them. Therefore, in this context of the cascading causal failure model, we study the impact of cascading causal failures on the distance between a pair of nodes in the network. More precisely, given a network G, a set of causal failures (containing possible cascading failures), a pair of nodes s and t, and a constant $\alpha \geq 1$, we would like to determine the maximum number of causal failures that can be applied (meaning that the nodes in the causal failures are removed), such that in the resulting network G', $d_{G'}(s,t) \leq \alpha \times d_G(s,t)$, where $d_G(s,t)$ and $d_{G'}(s,t)$ are the distance between nodes s and t in the networks G and G', respectively.

- In Chapter 5, we consider causal edge failures in flow networks and investigate the impact of causal edge failures on flow transmission. We formulate an optimization problem to find the maximum number of causal edge failures after which the flow network can still deliver d units from source node s to terminal node t.

- In Chapter 6, we consider edge-weighted network augmentation when facing causal failures. We look for a set of edges with minimum weight such that the network maintains an α-giant component when applying each causality individually.

We show that the optimization problems in these chapters are NP-hard and provide the corresponding mixed integer linear programming models. Moreover, we design polynomial-time heuristic algorithms to solve them approximately. In each chapter, we run experiments on multiple synthetic and real networks to compare the performance of the mixed integer linear programming models and the heuristic algorithms. The

results show that the heuristic algorithms show their efficacy and efficiency compared to the mixed-integer linear programming models.

Chapter 1

Introduction

1.1 Networks and Graphs

Networks are becoming much more complicated and complex. Components (i.e., nodes, edges) in these networks are tending to be correlated with each other in terms of functionality in engineering systems. It is the dependence on components that may trigger causal failures if the system experiences a malfunction.

Several examples indicate this phenomenon. For a power transmission network, Lin et al. (1) considered the correlated failures of physical lines in each edge and demonstrated that a high correlation of the physical lines would cause a critical degradation of network performance. In large-scale data centers, the fault of a small part may cause the failure of tasks running on them, severely damaging the reliability of those data centers (2). For the latest 5G networks, Ganjalizadeh et al. (3) investigated the impact of the real correlation among different wireless links on end-to-end reliability for two selected architectures from 3GPP.

Several works have investigated network performance, such as reliability and stability, in the case of such correlated failures. Lin et al. (4) constructed a stochastic flow network model to quantify the impact of correlated failures on the system reliability of a computer network, where a correlation is built between physical lines in edges and routers in nodes; Rahnamay-Naeini et al.(5) considered correlated link failures in the physical infrastructure of communication networks, presented a stochastic model to

facilitate the spatial properties, and compared network reliability for various scenarios of correlated link failures; Reis et al. (6) showed that the stability of a network system depends on the relation between the internal structure of a network and its pattern of connections to other networks.

There are also some works focusing on network design when there are correlated failures. Yang et al. (7) proposed two correlated link-weight models and analyzed the shortest path and min-cut problems defined on one of the two correlated models. Nath et al. (8) pointed out that node failures are often correlated and happen (nearly) simultaneously in wide-area storage systems and provided techniques about how to design systems to tolerate such failures. Similarly, Bakkaloglu et al. (9) utilized availability modeling to accommodate correlated failures in survival storage systems and proposed a design decision tool by validating the efficacy of the proposed model based on conditional probabilities. Lindley et al. (10) defined exchangeability in networks for designers of networks and presented a simple correlated failure model.

Sen et al. (11) proposed an operational model termed as the Implicative Interdependency Model (IIM). As an example of an operational model, consider the operational equation: $v_1 \leftarrow v_2v_3 + v_4v_5 + v_6$ where $v_i, i = 1, 2, \cdots, 6$ are nodes in the network. The semantic interpretation of the above equation is that node v_1 is operational if both v_2 and v_3 are operational or v_4 and v_5 are both operational or v_6 is operational. This operational model can be converted into the causal failure model proposed in this paper. In the above dependence equation, we know that if one of v_2, v_3 fails, one of v_4 and v_5 fails and v_6 fails, then v_1 would fail. Banerjee et al. (12) used IIM to launch protection analysis in multilayered interdependent networks. The authors aimed to maximize the benefit for network operators after finding specific k entities that need to be hardened. Furthermore, Banerjee et al. (13) considered the robustness of a multilayer interdependent network based on IIM.

1.2 Causal Failure Models

Infrastructure systems, such as electric grids, telecommunications, transportation, and the Internet, among others, often exhibit network structures. Such infrastructures play an essential role in our daily lives. Components (i.e., nodes, edges) in these infrastructures are often interdependent and subject to disruption caused by common failures, natural disasters, or deliberate attacks. Mathematically, a network is modeled as an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes, and $\mathcal{E}$ is the set of edges connecting nodes. The number of nodes is $n(= |\mathcal{V}|)$ and of edges is $m(= |\mathcal{E}|)$.

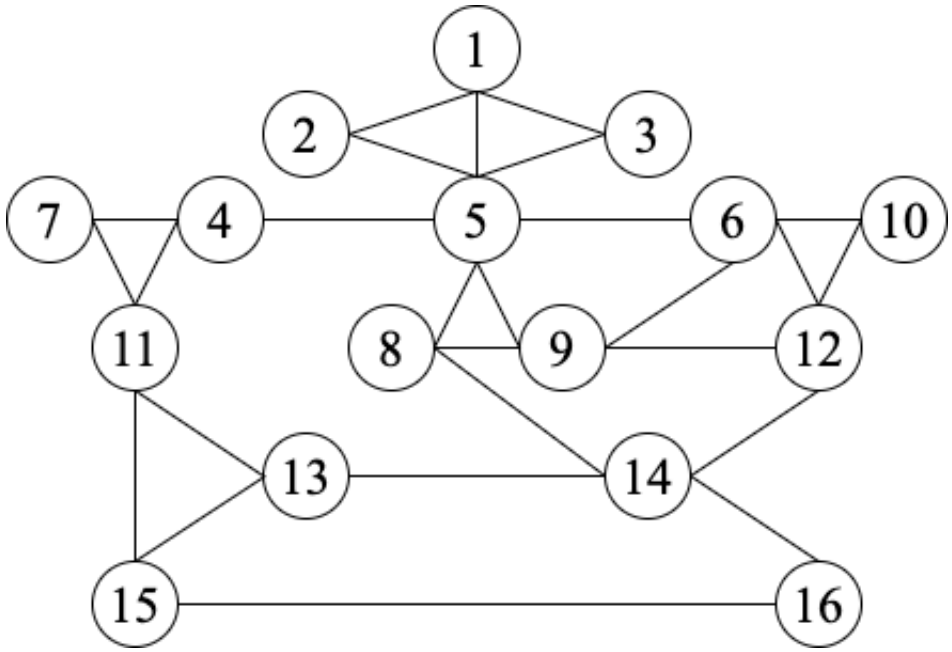


Figure 1.1: An example of a network

In classical models, a *failure* of a node in the network results in the removal of that node along with all its links. In the example network in Fig. 1, if node 5 fails, then edges linked to node 5, i.e. edges $(1, 5)$, $(2, 5)$, $(3, 5)$, $(4, 5)$, $(6, 5)$, $(8, 5)$, $(9, 5)$, are removed, leading to a disconnected network, where nodes $\{1, 2, 3\}$ and $\{4, 6, 7, 8, \cdots, 16\}$ form two connected sub-networks.

It is also possible that node failures would indirectly cause the failure of nodes that are not adjacent to the failed one. We are motivated to study the problem of *causal node failures* by the following examples.

Electrical Grid System

An electrical grid system consists of several generating stations (i.e., power stations) as nodes and transmission lines as links that connect the nodes. The power grid allows more reliability in the sense that the failure of one generating station will cause other nodes in the network to share the load serviced by the failed station. Additionally, when the peak load serviced by a station increases beyond its capacity, the grid distributes the extra load by planning with the other stations. A failure of a node i due to, for example, an increase in peak demand that is greater than its capacity, could cause the load to be serviced by another node j. If node j is unable to satisfy the demand, it will fail. The failure of node j as a result of the failure of node i is a causal failure.

Distributed Computing System

Consider a wide-area network such as the Internet. A cluster is a set of computers that are situated at a location, and are connected to one of the edge routers on the Internet. A distributed system consists of a network of nodes and links. Each node is a cluster of computers (as a single computer is also a cluster), and a link in the communication channel (via the Internet) between two clusters. A scheduler is a program that distributes work among the various clusters, which are chosen to execute tasks based on their capabilities. In such a distributed system, the most common cause for casual failure is server overload. Suppose a frontend for cluster A is handling 200 requests per second (QPS), and the frontend for cluster B handles 1000 QPS. One may

assume A and B need not be directly connected in the distributed network. If cluster A fails, requests that come to A may be routed to B by the scheduler. In this case, the front-end in B will have to handle 1200 QPS, which could result in it running out of resources, causing it to crash, miss deadlines, or otherwise act differently. In such a case, the failure of A causes a breakdown of B, a causal failure. The service rate of B will dip below 1000 QPS, and this could cause the scheduler to transport work to other clusters, which may, in turn, cause them to fail, and so on.

Networked Systems with Multiple Operating Systems

Consider a network where the routers run different versions of an operating system. A cyber attack on a router with a specific version of the operating system will result in exposure to the attack of other routers with that version of the operating system. That is, a failure (compromised due to a cyber-attack) of a router causes other routes to fail.

Hence, we formalize causal node failures with the following.

Definition 1.1. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, nodes $u, v \in \mathcal{V}$, the failure of node v as a result of the failure of node u is said to be a* causal failure *or simply* causal. *The causality is denoted as $C : u \Rightarrow v$.*

Definition 1.2. *A causality $C : u \Rightarrow v$ is* applied *implies that when node u is removed, v is also removed.*

Definition 1.1 is an extension of classical node failure models. Note that it is not required that nodes in C be adjacent, leading to the following.

Definition 1.3. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{V}_1, \mathcal{V}_2 \subseteq \mathcal{V}$, the failure of each node in $\mathcal{V}_2$ as a result of the failures of* all *nodes in $\mathcal{V}_1$ is said to be a* causal failure *(or causality)*

from $\mathcal{V}_1$ *to* $\mathcal{V}_2$. *One would then say that* the failure of $\mathcal{V}_1$ causes the failure of $\mathcal{V}_2$. *The causality is denoted as* $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$.

Definition 1.3 generalizes causal failures to node sets. In particular, failure of only a proportion of nodes in V_1 would not cause failures of any node in V_2 unless there exist other predefined causalities. In Fig.1.2, nodes of V_1 and V_2 in a causality C are denoted by dotted circles and dotted squares, respectively. We have causalities $C_1 : \{1, 4\} \Rightarrow \{9, 12\}, C_2 : \{5\} \Rightarrow \{11, 16\}, C_3 : \{3\} \Rightarrow \{7\}$. If C_1 is applied to the network, failures of nodes 1 and 4 cause the failure of nodes 9 and 12. However, the failure of node 1 alone does not cause any failure of nodes 9 or 12.

For the case that V_2 also affects V_1, we have the following.

Definition 1.4. *Given a network* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *and node sets* $\mathcal{V}_1, \mathcal{V}_2 \subseteq \mathcal{V}$, $C : \mathcal{V}_1 \leftrightarrow \mathcal{V}_2$ *is called a* correlated failure *if and only if* $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$ *and* $C : \mathcal{V}_2 \Rightarrow \mathcal{V}_1$ *are applied.*

Definition 1.4 illustrates that nodes in $\mathcal{V}_1$ and $\mathcal{V}_2$ affect each other mutually. For example, if Fig. 1.2 has correlated failures $C_1 : \{1, 4\} \leftrightarrow \{9, 12\}$, then the failures of nodes 9 and 12 also cause the failures of nodes 1 and 4.

A deeper analysis of all the works cited above informs us:

- The causal failure model is new. It is complementary to the operational model IIM described by Sen et al. (11). The work on correlated failures has not established a formal model of failures; only this work and Sen et al. (11) are those who have provided the formal model.

- Our work is the first to analyze the impact of causal failures on multiple aspects of one network.

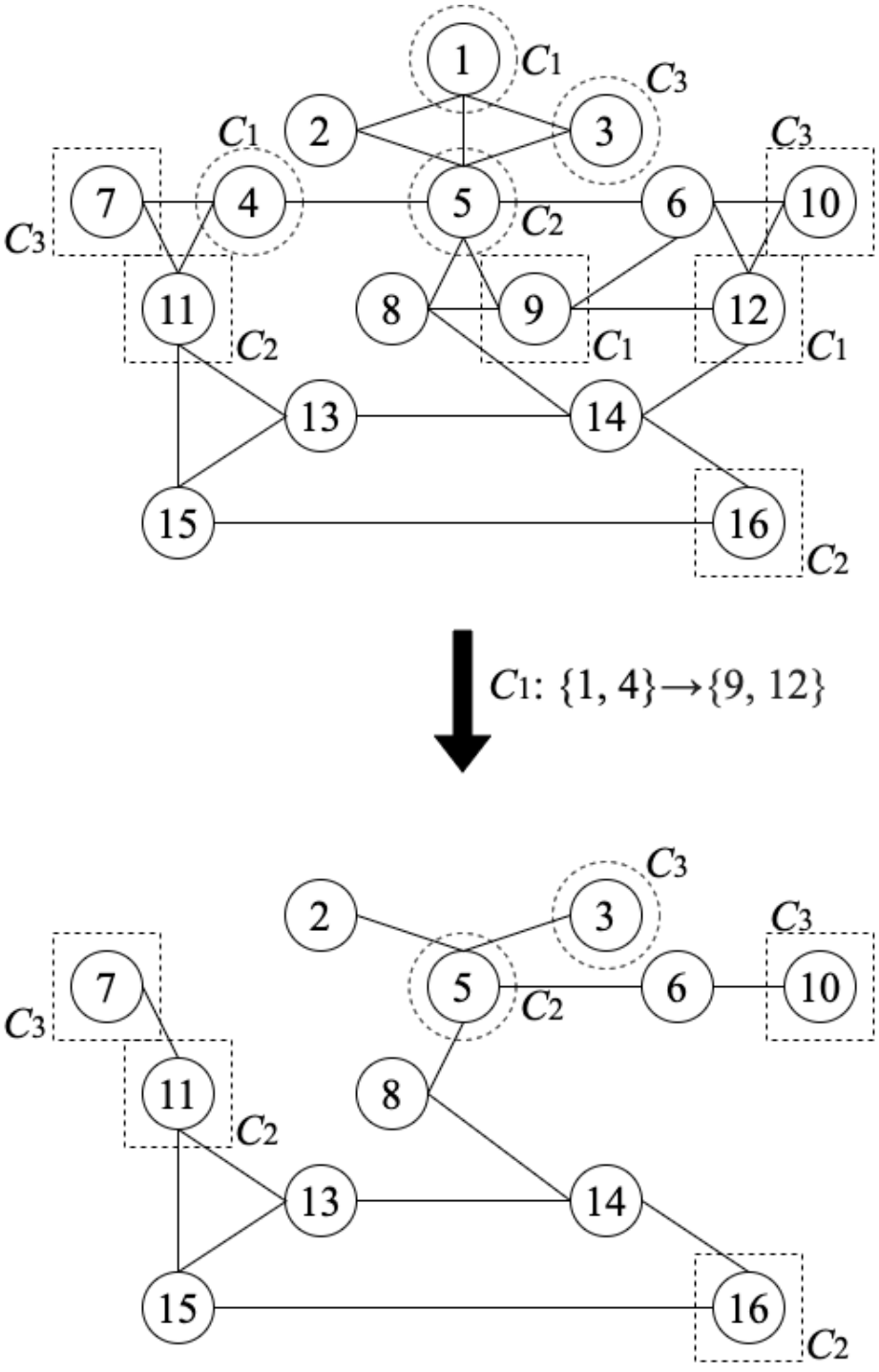


Figure 1.2: Residual network after applying causality $C_1 : \{1, 4\} \Rightarrow \{9, 12\}$

1.3 Extension of Causal Failures

Causal failures are applicable to not only nodes but also a mixture of nodes and edges (e.g., a subgraph). Mathematically, in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, a causality is denoted as $C : \mathcal{G}_1 \Rightarrow \mathcal{G}_2$, where $\mathcal{G}_1, \mathcal{G}_2 \subseteq \mathcal{G}$.

1. We replace $e = (u, v)$ in $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$with a new node and connect this node to u and v.

2. We replace each predefined failure with the failure of the corresponding node added in the last step.

Such a method generates a new graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{C}')$ where $\mathcal{V} \subseteq \mathcal{V}'$.

To make the reduction more concrete, we present an example in Fig. 1.3. Suppose a causality for edges $C : \{e_1, e_3\} \Rightarrow \{e_5, e_9\}$ is applied to $\mathcal{G}$, we have corresponding node causality $C : \{7, 9\} \Rightarrow \{11, 15\}$ applied to $\mathcal{G}'$. Therefore, all edge failures in $\mathcal{G}$ can be transformed into node failures in $\mathcal{G}'$. Consequently, causal node failures are the general version of causal failures related to all possible combinations of components (nodes/edges) in the network.

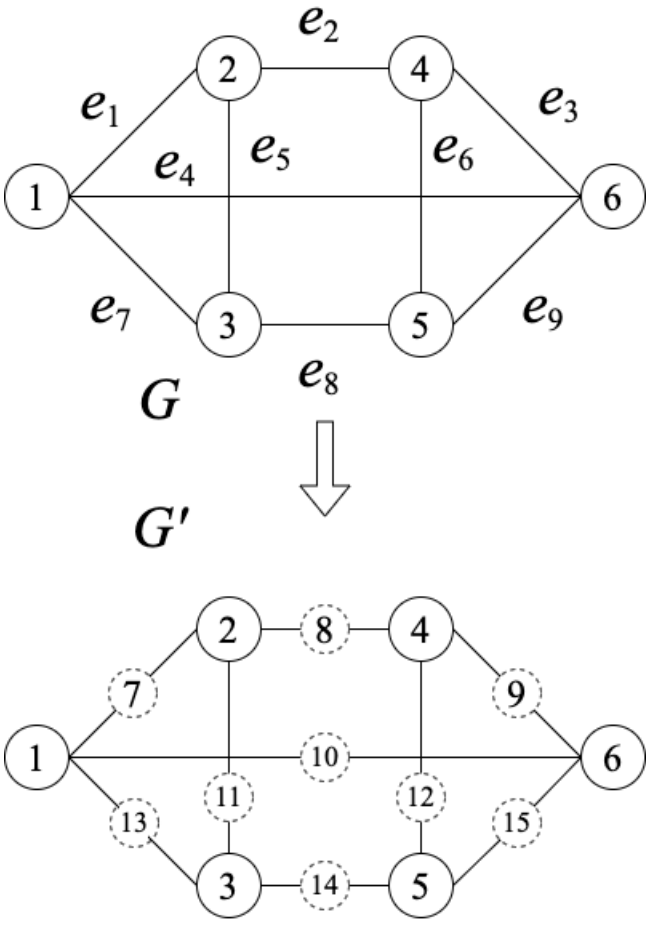


Figure 1.3: Transformation from edges to nodes

1.4 Nomenclature and Notations

Table 1.1: Nomenclature

QPS	reQuest Per Second
3GPP	3rd Generation Partnership Project
IIM	Implicative Inter-dependency Model
MILP	Mixed Integer Linear Programming
NP	Non-deterministic Polynomial
MAX-2SAT	MAXimum-2-SATisfiability problem
LHS	Left-Hand Side
RHS	Right-Hand Side
$VfCL$	The decision version of Problem 2.1
$VkCS$	The decision version of Problem 3.1
VkS	The decision version of Vertex $k-$cut problem
$VfCD$	The decision version of Problem 4.1
$VfCd$	The decision version of Problem 5.1

Table 1.2: Notations

$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	An undirected graph
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$	An undirected graph with causal failures
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{C})$	A directed graph with edge capacities and causal failures
$\mathcal{V}$	The set of nodes
$\mathcal{E}$	The set of edges
$\mathcal{C}$	The set of all causalities
$\mathcal{W}$	The capacities of edges
$n = \lvert\mathcal{V}\rvert$	The number of nodes
$m = \lvert\mathcal{E}\rvert$	The number of edges
$h = \lvert\mathcal{C}\rvert$	The number of causalities
$\mathcal{V}_i$	The i-th subset of $\mathcal{V}$
$\mathcal{E}_i$	The i-th subset of $\mathcal{E}$
$\mathcal{G}_i$	The i-th subgraph of $\mathcal{G}$
$C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$	A causality from $\mathcal{V}_1$ to $\mathcal{V}_2$
$C : \mathcal{E}_1 \Rightarrow \mathcal{E}_2$	A causality from $\mathcal{E}_1$ to $\mathcal{E}_2$
$C : \mathcal{G}_1 \Rightarrow \mathcal{G}_2$	A causality from $\mathcal{G}_1$ to $\mathcal{G}_2$
$\mathcal{V}(C)$	All nodes in causality C
$\mathcal{E}(C)$	All edges in causality C
α	Required factor of the size of a giant component
k	Required number of connected components
s	The source node
t	The terminal node
D	The length constraint on the shortest path from s to t
d	Required flow demand from s to t
$\mathcal{MC}$	The approximate solution obtained by a heuristic algorithm
t_h	The time consumed by a heuristic algorithm
opt	The solution obtained by MILP
t_o	The time consumed by MILP

Chapter 2

Causal Node Failures on the Robustness of a Network

2.1 Introduction

Many metrics have been proposed to evaluate network performance, such as reliability, dependability, and resilience, among others (14; 15). For example, in cyber-physical systems, a node's failure in one network would initiate a failure of other nodes, thereby reducing the reliability of the system (16). Network robustness refers to the ability of a network to maintain its functionality even when faced with disruptions or failures. A robust network is one that can withstand attacks, errors, and other types of disturbances and continue to operate normally. In today's interconnected world, where networks play a crucial role in many areas of society, from transportation to communication, the concept of network robustness is more important than ever. Ellens and Kooij (17) surveyed several measures for network robustness including classical measures (e.g., connectivity, distance, betweenness, clustering, reliability polynomials) and spectral measures (e.g., algebraic connectivity, number of spanning trees, and significant resistance).

One way to measure network robustness is to look at its resilience to failures. For example, if a transportation network is disrupted by an accident, how quickly can it recover and resume normal operations? If a communication network experiences a

power outage, how quickly can it be restored? These are the types of questions that researchers in network robustness are interested in answering.

Another important aspect of network robustness is its ability to resist attacks. In today's digital age, cyber-attacks on computer networks are becoming more frequent and sophisticated. A network that is not designed with robustness in mind can be easily compromised by these attacks, leading to data breaches, service disruptions, and other security issues.

To improve network robustness, there are several strategies that can be employed. One approach is to increase redundancy in the network. This means adding extra nodes or edges to the network so that if one fails, there are backup options available. For example, in a transportation network, having multiple routes to a destination can increase robustness. In a communication network, having multiple data centers or servers can provide redundancy.

Another strategy is to implement fault-tolerant mechanisms. This involves designing the network so that it can continue to operate even if some nodes or edges fail. For example, in a computer network, implementing load balancing can distribute the workload across multiple servers so that if one fails, the others can pick up the slack. In a transportation network, implementing dynamic rerouting can help vehicles avoid areas with congestion or accidents.

A third strategy is to improve the network's security. This can involve implementing encryption, firewalls, and other security measures to prevent unauthorized access or data breaches. It can also involve monitoring the network for suspicious activity and responding quickly to any threats that are detected.

In conclusion, network robustness is an important concept that is becoming increasingly important in today's interconnected world. A robust network can withstand

disruptions, failures, and attacks and continue to operate normally. To improve network robustness, strategies such as redundancy, fault tolerance, and security can be implemented. By designing networks with robustness in mind, we can ensure that they continue to provide the services and functionality that we depend on, even in the face of adversity.

To our best knowledge, we are the first to consider the measurement of network robustness by using causal node failures. Out contribution of this chapter is summarized as follows:

- We evaluate a network's robustness by computing the maximum number of causalities that can be applied while simultaneously ensuring that the modified network maintains a large giant component.
- We show that the decision problem for the above is NP-complete.
- We provide an optimization model for the above problem and propose a heuristic algorithm to solve it.
- We apply our algorithms to publicly available electric grid network data.

2.2 Problem Formulation

We consider the impact of causality on the connectivity of the network. First, we recall the definition of vertex-induced graphs and connected components in graph theory:

Definition 2.1. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, a vertex-induced sub-graph of $\mathcal{G}$ is another graph formed by a subset of $\mathcal{G}$ and all edges between any pair of vertices in this subset.*

Definition 2.2. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, a connected component is a vertex-induced subgraph where any pair of vertices are connected to each other, and which is not connected to any other vertex in the rest graph.*

As Definitions 2.1 and 2.2 show, vertex-induced subgraphs and connected components are not completely the same. For instance, in Fig.1, a subgraph with vertices $\{1, 8, 14\}$ is a vertex-induced subgraph, but vertex 1 is not connected to vertex 8 in this subgraph. Moreover, if we remove vertex 5, then we have a subgraph consisting of vertices $\{1, 2, 3\}$ and edges $\{(1, 2), (1, 3)\}$. This subgraph is a connected component because it is not connected to any vertex of the rest graph.

After applying causalities, the network might be a collection of connected components. To specify the connected components in terms of size, we have the following.

Definition 2.3. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$ and a constant $0 < \alpha < 1$, an α-giant component of graph $\mathcal{G}$ is a connected induced sub-graph where the number of nodes is at least $\alpha|\mathcal{V}|$.*

The giant component is a prominent characteristic of networks because it provides a critical threshold for random failures and can be used to quantify the robustness of networks. In Fig. 1.2, given $\alpha = 0.45$, applying causality $C_2 : \{5\} \Rightarrow \{11, 16\}$ results in a disconnected network with 3 connected components $\{1, 2, 3\}$, $\{4, 7\}$, $\{6, 8, 9, 10, 12, 13, 14, 15\}$, in which the largest has 8 nodes. Similarly, if we apply causality $C_1 : \{1, 4\} \Rightarrow \{9, 12\}$, we have a connected network with nodes 2, 3, 5, 6, 7, 8, 10, 11, 13, 14, 15, 16. Moreover, a connected subgraph with 13 nodes remains after causality $C_3 : \{3\} \Rightarrow \{7, 10\}$ is applied. These cases only apply one causality, but the connectivities of the remaining networks are different from each other. Additionally, the sizes of the giant components obtained from the application of C_2 or C_3 are larger in comparison with the size obtained after applying C_1.

In a power-grid system, connectivity plays a vital role. The number of nodes (e.g., generators) that are part of a connected network indicates the system's capacity. It also indicates how the excess capacity of one node can be transferred to another in the connected network. A giant component is the largest connected component. It informs us of the largest connected component (or the total combined capacity of a connected network) one could have in the face of causal failures. Therefore, we attempt to determine the maximum number of causal failures that can be applied yet maintain a certain level of total capacity (indicated by the parameterized giant component determination).

As such, we formulate the problem as follows.

Problem 2.1 (Giant Components)**.** *Given a network* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$*, where the set of causal failures* $\mathcal{C} = \{C_i : \mathcal{V}_{1,i} \Rightarrow \mathcal{V}_{2,i}, i = 1, 2, \cdots, h\}$*, and a fraction* $0 < \alpha < 1$*, the* Giant Component *problem seeks to find a maximum size set* $\hat{\mathcal{C}} \subseteq \mathcal{C}$ *of causal failures whose application leaves at least one* α*-giant component of the initial graph surviving in the remaining graph.*

Different from the problem in (13), problem 2.1 is not measuring robustness by looking for a subset of nodes whose failures cause the failure of a proportion of nodes. Instead, problem 2.1 identifies the maximum number of causalities that can be applied without breaking down all giant components of the network. It properly evaluates how strong the network is (i.e., its *robustness*) in the face of causal failures. Note that when $\alpha > 1/2$, there can be at most one α-giant component that survives. Hence, the above problem is rephrased (when $\alpha > 1/2$) to finding the maximum number (and a corresponding subset) of causalities whose application results in a remaining network with an α-giant component of the initial graph surviving.

2.3 NP-Hardness Proof

We demonstrate the NP-hardness of the problem in the previous section. We first define the decision version of Problem 2.1, denoted as *VfCL*: given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, and integers f and L, does there exist $\mathcal{C}' \subset \mathcal{C}$ with size at least f such that $\mathcal{G}$ has a component whose size is at least L after applying causalities in $\mathcal{C}'$?

First, we prove NP-completeness by presenting a polynomial-time reduction from the decision version of the MAX-2SAT problem ((18)) to this version of *VfCL*. The decision version of MAX-2SAT is described as: given 2-clauses $c_1, \ldots, c_p$, is there a truth assignment that satisfies at least f clauses? This problem is NP-complete.

Theorem 2.1. *VfCL is NP-complete.*

Proof. First, *VfCL* is in NP. Given a candidate certificate of *VfCL*, one could verify in polynomial time whether the remaining graph has a component with L vertices, by using a depth-first search.

Then we exhibit a polynomial-time reduction ϕ from an instance I of MAX-2SAT into an instance I' of *VfCL*:

- We create dummy nodes s and t. Suppose I has l boolean variables $\{x_1, \ldots, x_l\}$. Then we add $2l$ special nodes $X_a, \bar{X}_a, a = 1, \ldots, l$, all of which are connected to s. For each $a = 1, \ldots, l$, we also introduce a new large clique C_a on Y vertices which is joined by an edge to each of X_a and $\bar{X}_a$. Here, we assume that $Y := 2(2+l+5p)$.

- For clause $c_i = x_a \vee x_b$, we add nodes x_a, x_b, $c_{i,1}$, $c_{i,2}$, and a dummy node d_i, and edges (s, x_a), (s, x_b), (x_a, d_i), (x_b, d_i), $(c_{i,1}, d_i)$, $(c_{i,2}, d_i)$, $(c_{i,1}, t)$, and $(c_{i,2}, t)$.

- For clause $c_i = x_a \vee x_b$, we construct two causalities $C_{i,1} : x_a \Rightarrow \{X_a, c_{i,1}\}$ and $C_{i,1} : x_b \Rightarrow \{X_b, c_{i,2}\}$. If there is $\bar{x}_a$ in clause c_i, the corresponding causality will include $\bar{X}_a$.

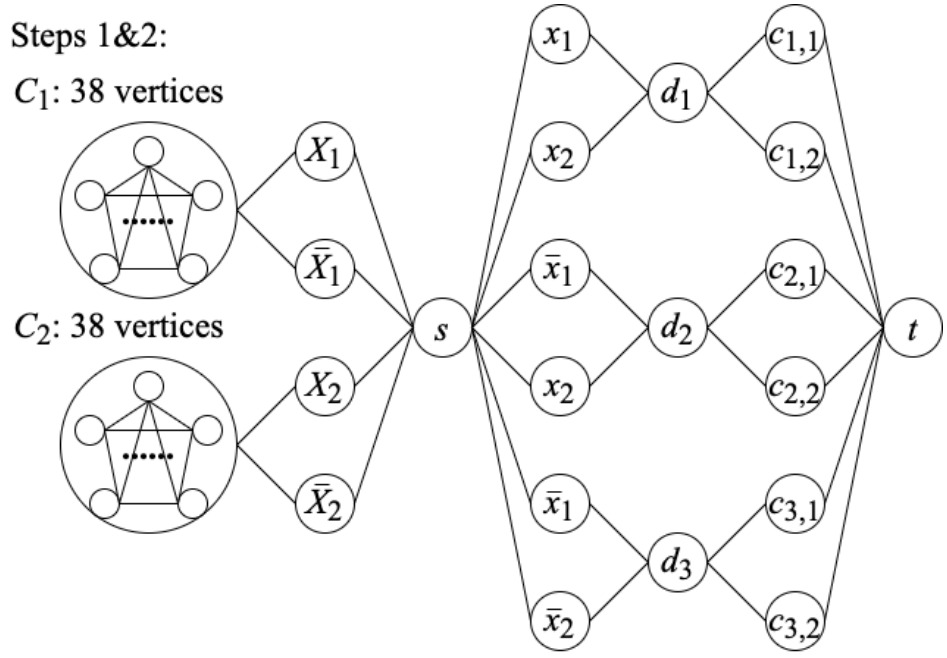
- We require that there are at least $L = lY$ nodes in the largest component. This requirement avoids that x_a and $\bar{x}_a$ would not be picked together, otherwise both nodes X_a and $\bar{X}_a$ would be removed and hence C_a would be isolated. Second, only one causality would be applied for the clause c_i, otherwise dummy node d_i is isolated.

Clearly, the size of I' is bounded by a polynomial in the size of I. Suppose that I is satisfiable. Then we have an assignment that satisfies exactly $f' \geq f$ clauses, so we pick one causality (corresponding to a literal becoming true) from each clause, and the largest component of the remaining network has at least $L' = 3f'+2+l+5(p-f')+lY \geq L$ nodes.

Suppose that every assignment of I satisfies at most $f-1$ clauses. Suppose further I' admits a set $\mathcal{C}'$ of $f' \geq f$ causalities whose application leaves a graph with some component on at least L vertices. We claim that there cannot be two causalities in $\mathcal{C}'$ whose LHSs contain respectively x_a and $\bar{x}_a$ for some a. If otherwise, the clique C_a would be isolated and hence any other component can have at most $2+2(l-1)+5p+(l-1)Y \leq L-Y+2l+5p < L$ nodes, contradicting our assumption. Hence, our claim holds true. Now consider the assignment ψ of Boolean values to $\{x_1, \ldots, x_l\}$ defined by $\psi(x_a) = T$ (or $\psi(x_a) = F$) depending on whether x_a (or $\bar{x}_a$) is in the LHS of some causality in $\mathcal{C}'$. For other variables, we arbitrarily set their values to one of T or F. In view of the claim above, this is a valid assignment that satisfies at least $f' \geq f$ clauses of I, thus contradicting our assumption. Hence, I' does not admit such a set $\mathcal{C}'$ of $f' \geq f$ causalities. This proves the theorem. □

Fig. 2.1 shows an example of the proof of Theorem 1. The MAX-2SAT instance is $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$ where $l = 2, p = 3$. Hence, we construct a graph with 97 nodes and 6 causalities. Cliques C_1 and C_2 both have $38 (= 2(2+2+3\times 5))$ nodes.

It is required that there are at least $L = lY = 2 \times 38 = 76$ nodes in the remaining network. We set $x_1 = T, x_2 = T$ and apply causalities related to x_2 only, then we have 2 causalities applied to the network and maintain 93 nodes.



Step 3: $x_1 \rightarrow \{X_1, c_{1,2}\}, x_2 \rightarrow \{X_2, c_{1,1}\}, \bar{x}_1 \rightarrow \{\bar{X}_1, c_{2,2}\},$
$x_2 \rightarrow \{X_2, c_{2,1}\}, \bar{x}_1 \rightarrow \{\bar{X}_1, c_{3,2}\}, \bar{x}_2 \rightarrow \{\bar{X}_2, c_{3,1}\}$

Step 4: x_1 = True, x_2 = True, pick x_2

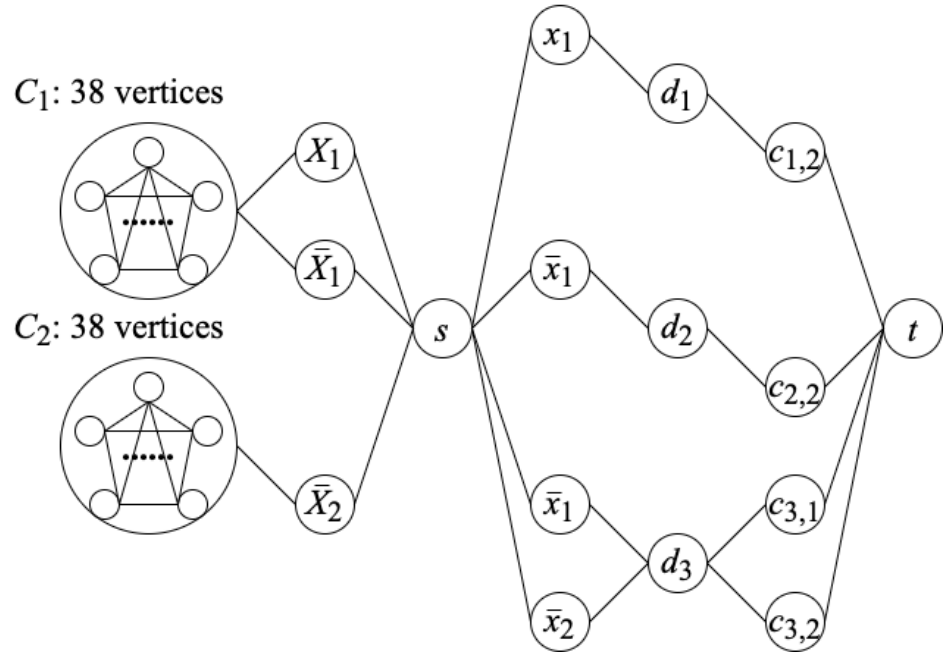


Figure 2.1: Example of proof of Theorem 2.1

2.4 Algorithms

2.4.1 MILP for Problem 2.1

We first provide a MILP formulation, denoted by MILP-GC, to seek an α-giant component. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$ be the undirected network and $\mathcal{DG} = (\mathcal{V}, \mathcal{A}, \mathcal{C})$ be its directed version obtained by replacing each edge of E with two directed edges in opposite directions. Let n denote $|\mathcal{V}|$ and $\mathbb{N}$ denote the set of nonnegative integers.

The formulation is inspired by a flow formulation (19) which aims to determine the largest connected component of an undirected graph. We set up a source node s and terminal node t, each of which has edges incident on all nodes in the network. In the newly constructed graph, x_{ij} denotes the flow from node i to node j. The description of the above formulation is listed as follows.

- Eqs. (2.1), (2.12): Binary variable z_C indicates that whether causality C is applied to the network. Problem 1 seeks the maximum number of applied causalities, therefore the objective function is the sum of z_C, which should be maximized.

- Eq. (2.2): This constraint ensures that there exists a giant component in the residual network. Correspondingly, the total amount of flow delivered to t is at least αn.

- Eqs. (2.3), (2.4): These constraints govern flow conservation (i.e., the total amount of flow from the source node s must be equal to the flow to the terminal node t, and the incoming flow is equal to the outgoing flow for each node $i \in \mathcal{V}$).

- Eqs. (2.5),(2.6): These constraints ensure that the entire flow from s is received directly by only one i, and this flow cannot exceed n.

- Eqs. (2.7), (2.8): The flow along each edge is an integer. Moreover, the capacity of each edge (i,t) is 1 to ensure that at most one unit reaches t from any i.

- Eqs. (2.9), (2.10), (2.11): If a causality C is applied to the network, no flow is allowed to pass through any node $i \in \mathcal{V}(C)$; otherwise, the flow amount is not greater than n.

$$
\begin{aligned}
maximize \quad & \sum_{C \in \mathcal{C}} z_C && (2.1)\\
s.t. \sum_{i \in \mathcal{V}} x_{it} &\geq \alpha n && (2.2)\\
\sum_{i \in \mathcal{V}} x_{si} &= \sum_{i \in \mathcal{V}} x_{it} && (2.3)\\
x_{si} + \sum_{j:(j,i) \in \mathcal{A}} x_{ji} &= x_{it} + \sum_{j:(i,j) \in \mathcal{A}} x_{ij}, i \in \mathcal{V} && (2.4)\\
\sum_{i \in \mathcal{V}} y_i &= 1 && (2.5)\\
x_{si} &\leq y_i n,\ i \in \mathcal{V} && (2.6)\\
y_i, x_{it} &\in \{0,1\},\ i \in \mathcal{V} && (2.7)\\
x_{si}, x_{ij} &\in \mathbb{N},\ i \in \mathcal{V}, (i,j) \in \mathcal{A} && (2.8)\\
x_{ij} &\leq (1 - z_C)n, C \in \mathcal{C}, \\
&\quad i \in \mathcal{V}(C), (i,j) \in \mathcal{A} && (2.9)\\
x_{ji} &\leq (1 - z_C)n, C \in \mathcal{C}, \\
&\quad i \in \mathcal{V}(C), (j,i) \in \mathcal{A} && (2.10)\\
x_{si}, x_{it} &\leq (1 - z_C)n, C \in \mathcal{C}, \\
&\quad i \in \mathcal{V}(C) && (2.11)\\
z_C &\in \{0,1\},\ C \in \mathcal{C} && (2.12)\\
x_{ij} &\in \mathbb{N}, (i,j) \in \mathcal{A} && (2.13)
\end{aligned}
$$

Complexity analysis: this formulation has $h + 2n$ binary variables, which are h variables (i.e., $z_C, C \in \mathcal{C}$) indicating whether a causality is applied, n variables (i.e. $x_{it}, i \in \mathcal{V}$) indicating whether a node $i \in \mathcal{V}$ has flow 1 shipping to t, n variables (i.e. $y_i, i \in \mathcal{V}$) indicating whether a node $i \in \mathcal{V}$ receives at most n unit of flow from s. In addition, there are $2m+n$ integral variables ranging from 1 to n, which are $2m$ variables (i.e., $x_{ij}, x_{ji}, ij \in \mathcal{E}$) showing the flow from nodes i to j or from j to i and $x_{si}, i \in \mathcal{V}$ representing the flow from s to node i. Hence, the overall computation complexity is $O(2^{n+h}n^{m+n})$.

2.4.2 A Heuristic Algorithm

In Problem 1, we hope to find the largest number of causalities that could be applied to one network while maintaining one giant component in the remaining network. Let $C_i : \mathcal{V}_{1,i} \Rightarrow \mathcal{V}_{2,i},\ i = 1, 2, \cdots, h$ be a set of causalities. First, we assume that $|\mathcal{V}_{1,i}| = |\mathcal{V}_{2,i}| = 1$, for $i = 1, 2, \cdots, h$.

Let $d_{\mathcal{G}}(v)$ be the degree of a node v in $\mathcal{G}$ and we denote by $\chi(C_i)$ as the number of edges each causality destroys:

$$\chi(C_i) = \begin{cases} d_{\mathcal{G}}(\mathcal{V}_{1,i}) + d_{\mathcal{G}}(\mathcal{V}_{2,i}), & if\ (\mathcal{V}_{1,i}, \mathcal{V}_{2,i}) \notin \mathcal{E} \\ d_{\mathcal{G}}(\mathcal{V}_{1,i}) + d_{\mathcal{G}}(\mathcal{V}_{2,i}) - 1, & \text{otherwise} \end{cases} \tag{2.14}$$

We have a greedy algorithm as follows.

We will now present a generalized version with the assumption that $|\mathcal{V}_{1,i}| \geq 1$ and $|\mathcal{V}_{2,i}| \geq 1$. Correspondingly, let

- $d_{\mathcal{G}}(\mathcal{V}_{p,i})$ be the sum of degrees of each node in set $\mathcal{V}_{p,i}, p = 1, 2$,
- $\mathcal{E}(\mathcal{L}, \mathcal{M})$ be the set of all edges $e = (u, v) \in \mathcal{E}$ such that $u \in \mathcal{L}$ and $v \in \mathcal{M}$,

Algorithm 1 GREEDY-MAX-GC

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, constant α;
Output: A set of causalities $\mathcal{MC}$, s.t. when all causalities in $\mathcal{MC}$ are applied to $\mathcal{G}$, there exists at least one giant component of $\mathcal{G}$ with size larger than $\alpha \times |\mathcal{V}|$.

1: Sort the set $\mathcal{C}$ based on $\chi(C)$ for all $C \in \mathcal{C}$ and let $\chi(C_1) \leq \chi(C_2) \leq \cdots \leq \chi(C_h)$;
2: $S \leftarrow |\mathcal{V}|$;
3: $\mathcal{G}^1 \leftarrow \mathcal{G}$;
4: $i \leftarrow 1$;
5: $\mathcal{MC} \leftarrow \varnothing$;
6: **while** $S > \alpha \times |\mathcal{V}|$ **do**
7: $\quad \mathcal{G}^1 \leftarrow$Apply C_i to $\mathcal{G}^1$;
8: $\quad S \leftarrow$size of the largest connected component of $\mathcal{G}^1$;
9: $\quad \mathcal{MC} \leftarrow \mathcal{MC} \cup \mathcal{C}'$;
10: $\quad i \leftarrow i + 1$;
11: **end while**
12: **return** $\mathcal{MC}$.

- $\mathcal{T}(\mathcal{L}, \mathcal{M})$ be the set of all nodes $v \in \mathcal{V}$ such that $(u_1, v) \in \mathcal{E}$ and $(u_2, v) \in \mathcal{E}$ with $u_1 \in \mathcal{L}$ and $u_2 \in \mathcal{M}$ where $\mathcal{L}, \mathcal{M} \subset \mathcal{V}$,

Then we have

$$\begin{aligned} \chi(C_i) = {} & d_{\mathcal{G}}(\mathcal{V}_{1,i}) + d_{\mathcal{G}}(\mathcal{V}_{2,i}) + |\mathcal{V}_{1,i}| + |\mathcal{V}_{2,i}| \\ & - 2|\mathcal{E}(\mathcal{V}_{1,i}, \mathcal{V}_{2,i})| - 2|\mathcal{E}(\mathcal{V}_{1,i}, \mathcal{V}_{1,i})| \\ & - 2|\mathcal{E}(\mathcal{V}_{2,i}, \mathcal{V}_{2,i})| - 2|\mathcal{T}(\mathcal{V}_{1,i}, \mathcal{V}_{2,i})| \end{aligned} \tag{2.15}$$

$\chi(C_i)$ denotes the number of edges the nodes in C_i are incident to. Eq. (2.15) ensures that all edges are counted once. Now we use the values of $\chi(C_i)$ in Step 2 of Algorithm 2 to find a large set of causalities.

Complexity analysis: We assume that the graph is represented as its adjacency matrix. Step 2 sorts the metric values of all causalities, so it takes $O(h \log h)$ time. The while loop (steps 6-11) does at most h operations on searching for connected components, which takes $O(n^2)$. Hence, the total time complexity is $O(max(h \log h, hn^2)) = O(hn^2)$.

Approximation ratio: algorithm 1 greedily looks for causalities to be applied based on the total degree metric χ. It can be envisioned that the extreme situation is that causalities are defined on low-degree cut vertices. Based on the greedy strategy, these causalities would be chosen first and might quickly make the size of the largest giant component less than $\alpha \times n$ after applying one causality. On the other hand, the optimal solution might be $h-1$, i.e. applying all causalities except the one that disconnects the giant component. Hence, we have $|\tilde{\mathcal{C}}|/|\mathcal{MC}| \leq h-1$.

Example 2.1. *Given causalities* $C_1 : \{1,4\} \Rightarrow \{9,12\}, C_2 : \{5\} \Rightarrow \{11,16\}, C_3 : \{3\} \Rightarrow \{7\}$ *and* $\alpha = 0.5$*, the maximum set of causalities is obtained as follows:*

1: $\chi(C_1) = 14, \chi(C_2) = 16, \chi(C_1) = 6$*, so sorted causalities are* C_3*,* C_1*,* C_2*;*

2: $S \leftarrow 16$*;*

3: $\mathcal{G}^1 \leftarrow \mathcal{G}$*;*

4: $i \leftarrow 1$*;*

5: $\mathcal{MC} \leftarrow \varnothing$*;*

6: (While loop:);

7: Apply C_3 *to* $\mathcal{G}^1$ *and* $S = 14 > 8$*;*

8: $\mathcal{MC} = \{C_3\}$*;*

9: $i = 2$*;*

10: Apply C_1 *to* $\mathcal{G}^1$ *and* $S = 10 > 8$*;*

11: $\mathcal{MC} = \{C_3, C_1\}$*;*

12: Apply C_2 *to* $\mathcal{G}^1$ *and* $S = 4 < 8$*;*

13: (End while);

14: Output: $\mathcal{MC} = \{C_3, C_1\}$

The output $\mathcal{MC} = \{C_3, C_1\}$ *illustrates that at most 2 causalities can be applied to the network in Fig. 1 while maintaining a giant component with at least eight nodes.*

2.5 Experimental Results

We present experiments to validate the performance of the proposed optimization models and algorithms, which are coded into a Python program and implemented on a PC with a 2.6 GHz 6-Core Intel Core i7 with a RAM of 16GB. Particularly, heuristic algorithms are coded with NetworkX (20), and optimization models are coded with Gurobi (21).

2.5.1 An Illustrative Example

The IEEE 300 bus system (22) is a popular test system (Fig. 2.2) and has been tested in many models, such as false data injection attacks (23), sequential power flow calculation (24), and probabilistic load margins (25), among others.

We use a subnetwork of the system and randomly generate a set of 25 disjoint causalities, where $\mathcal{V}_{i,1}$ and $\mathcal{V}_{i,2}$ $(i = 1, \ldots, 25)$ are both of size at most five. Fig.2.3 presents the residual network after applying 19 causalities when $\alpha = 0.5$. It can be seen that there is only one giant component remaining.

Table 2.1 shows the maximum number of causalities that can be applied to the network in terms of α. The results obtained from GREEDY-MAX-GC and MILP-GC are close, but GREEDY-MAX-GC takes less time. Especially for $\alpha = 0.5$, it takes 0.63s for GREEDY-MAX-GC to find 15 causalities, while MILP-GC spends 38.8s looking for all possible causalities to be applied.

2.5.2 Investigation of Giant Components of IEEE 300

Recalling the set-up that the number of nodes in $\mathcal{V}_1$ (or $\mathcal{V}_2$) in $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$ follows a uniform distribution between 1 and 5, we know that the expected number of nodes in

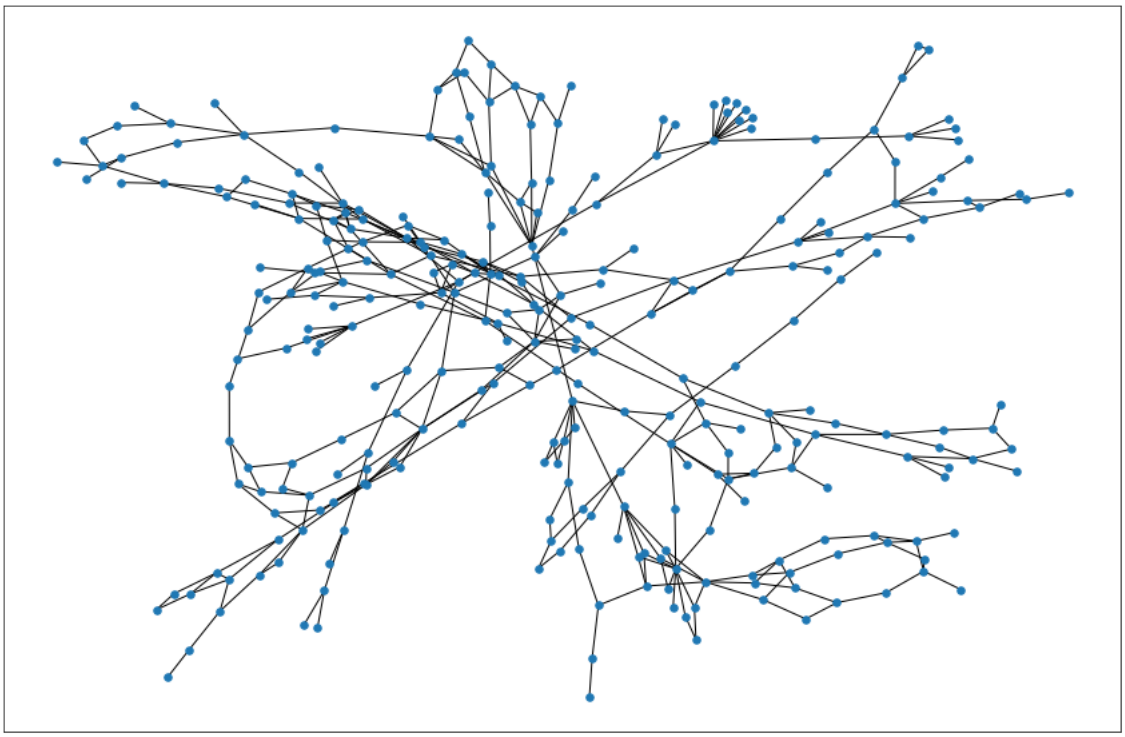


Figure 2.2: IEEE 300 bus system

Table 2.1: IEEE 300: first trial for the giant component problem

	GREEDY-MAX-GC		MILP-GC	
α	$\lvert\mathcal{MC}\rvert$	t_h(s)	opt	t_o(s)
0.1	25	0.55	25	0.78
0.2	24	0.54	24	0.77
0.3	19	0.53	22	3.10
0.4	18	0.54	21	12.35
0.5	15	0.63	19	38.80
0.6	15	0.67	17	12.43
0.7	13	0.67	14	6.53
0.8	9	0.67	11	1.84
0.9	5	0.61	7	0.97

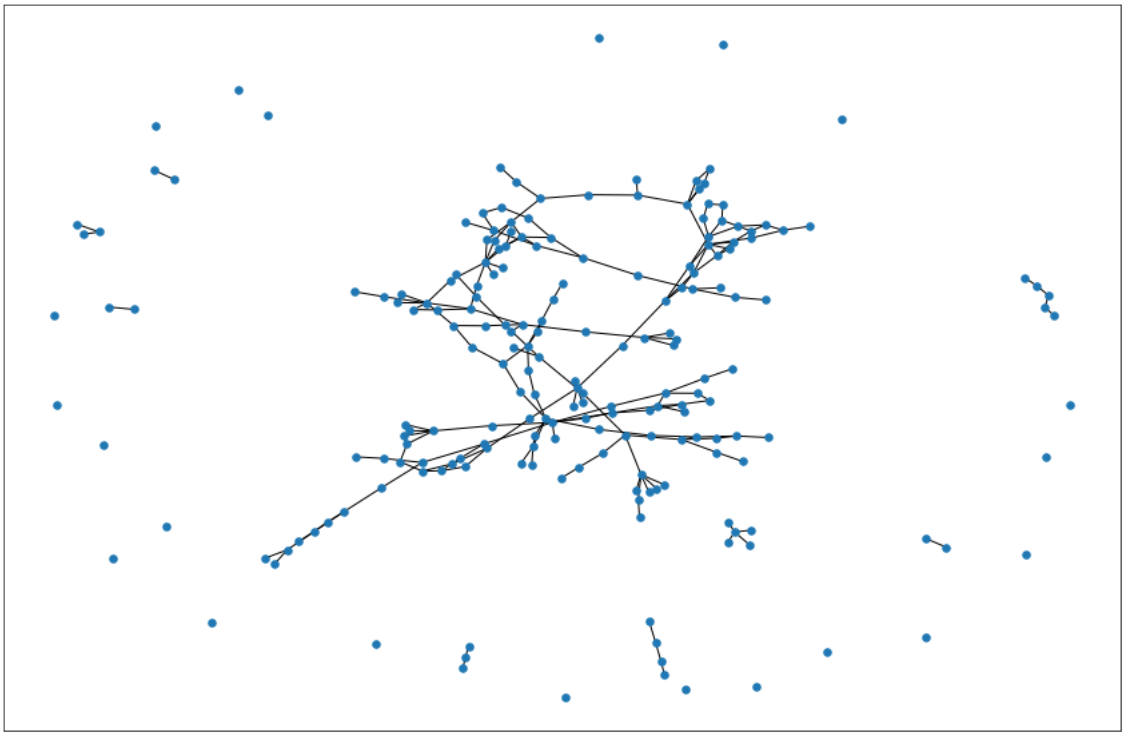


Figure 2.3: IEEE 300: remaining network containing a giant component with size at least 150 ($\alpha = 0.5$)

each causality is 6. To explore more properties of this system, we set up 10 experiments, each of which generates 30 causalities. Correspondingly, the expected number of nodes to be failed is 180. Given that it was demonstrated that GREEDY-MAX-GC works well in terms of accuracy, we use GREEDY-MAX-GC in the experiments, the results of which can be found in Table 2.2.

Table 2.2: IEEE 300: 30 causalities for the giant component problem with 180 total number of expected nodes failures

	Experiment No.										
α	1	2	3	4	5	6	7	8	9	10	mean
0.1	26	28	29	23	24	23	24	29	18	27	25.1
0.2	23	22	23	19	22	17	20	26	15	24	21.1
0.3	20	20	21	17	22	15	20	22	14	22	19.3
0.4	17	13	17	16	21	14	18	19	13	21	16.9
0.5	17	12	14	13	19	13	15	19	10	21	15.3
0.6	16	12	11	13	14	12	14	18	9	19	13.8
0.7	11	11	10	13	12	11	11	15	9	15	11.8
0.8	7	6	8	10	9	10	8	13	8	11	9
0.9	4	5	5	7	5	7	6	8	5	7	5.9

From Table 2.2, it can be observed that the greater the desired size of the giant component is, the fewer will be the number of causalities that need to be applied. More specifically, the number of applied causalities indicates the expected number of failed nodes, which have a close relationship with the size of the giant component. For instance, given $\alpha = 0.5$, the network allows 15.3 causalities to be applied, indicating that the number of failed nodes is expected to be $15.3 \times 6 \approx 92$. This means that to maintain a giant component with at least $300 \times 0.5 = 150$ nodes, on average 92 nodes can fail. Similarly, if there is a giant component with at least $300 \times 0.1 = 30$

nodes in the remaining network, on average we could let $25.1 \times 6 = 151$ nodes fail. In fact, the total number of nodes in the applied causalities is likely to be smaller than the expected number because the algorithm we use usually searches for the causality containing small numbers of nodes first. It is possible that even after we apply almost all causalities, we will still be able to get an α-giant component when the α is kept smaller. For example, in experiments 3 and 8, 29 causalities are applied to the network while we have a 0.1-giant component.

2.5.3 Scalability Analysis

To analyze the scalability of GREEDY-MAX-GC and MILP-GC, we use three network data sets, i.e., IEEE 300, power network 1138-bus (26), and Western US Power Grid (27) (written as power5000 in Table 2.3) with 300, 1138 and 4941 nodes respectively. In addition, we set up our experiments on these data sets as follows:

1. For each network, we randomly generate 10, 20, and 30 causalities;
2. For each causality $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$, $|\mathcal{V}_1| \leq 5$ and $|\mathcal{V}_2| \leq 5$ for IEEE 300, $|\mathcal{V}_1| \leq 20$ and $|\mathcal{V}_2| \leq 20$ for power network 1138-bus, $|\mathcal{V}_1| \leq 80$ and $|\mathcal{V}_2| \leq 80$ for Western US Power Grid;
3. $\alpha = 0.9$;

In Table 2.3, we could see that the numbers of applied causalities obtained by GREEDY-MAX-GC are the same as those obtained by executing MILP-GC. However, the time consumed by the heuristic algorithm is much less than that by MILP. For instance, given 30 predefined causalities for the Western US Power Grid, the heuristic and the MILP model consumed 5.18 seconds and 396.41 seconds, respectively, and produced 8 casualties that can be applied. Moreover, as the number of nodes increases,

Table 2.3: Scalability analysis for giant component

Number of causalities	GREEDY-MAX-GC						MILP					
	IEEE300		Power1138		power5000		IEEE300		Power1138		power5000	
	$\lvert\mathcal{MC}\rvert$	t_h(s)	$\lvert\mathcal{MC}\rvert$	t_h(s)	$\lvert\mathcal{MC}\rvert$	t_h(s)	opt	t_o(s)	opt	t_o(s)	opt	t_o(s)
10	5	0.08	5	0.26	5	1.07	5	0.77	5	8.63	5	151.22
20	5	0.18	5	0.60	6	2.74	5	1.28	5	14.61	6	537.42
30	7	0.30	5	0.84	8	5.18	7	1.10	5	49.46	8	396.41

the time taken by MILP increases exponentially. For example, when there are 10 predefined causalities, the running times for the three networks are 0.77 seconds, 8.63 seconds, and 151.22 seconds, respectively. This is in agreement with the computational complexity of MILP mentioned in section 2.4. By contrast, the running time of the heuristic algorithm is only 0.08 seconds, 0.26 seconds, and 1.07 seconds, respectively. Hence, GREEDY-MAX-GC when compared with the MILP formulation is comparable in terms of accuracy and efficiency in terms of running time.

2.6 Conclusion

In this chapter, we consider the impact of causal node failures on the robustness of one network, which is measured as the maximum number of applied causalities while the network maintains a giant component. We formally formulate the problem and prove the NP-hardness of the problem. Correspondingly, a mixed integer linear programming model is proposed to obtain the exact solution and a heuristic algorithm is designed to approximate the solution. Experimental results based on the IEEE 300 Bus, Power1138, and the Western US Power Grid system show that the heuristic algorithm can be considered as an alternative to the optimization model to substantially reduce the computational time.

Chapter 3

Causal Node Failures on the Vulnerability of a Network

3.1 Introduction

In today's increasingly interconnected world, networks are an essential part of modern life. They enable communication, data transfer, and access to information on a global scale. However, this interconnectedness also means that networks are vulnerable to a wide range of threats, both internal and external. In this essay, we will explore network vulnerability, its causes, and the steps that can be taken to mitigate the risks.

Network vulnerability refers to the susceptibility of a network to malicious attacks or unauthorized access. This can be caused by a variety of factors, including outdated software, weak passwords, unpatched systems, or human error. Cybercriminals can exploit these vulnerabilities to gain access to sensitive data, install malware, or launch attacks on other networks.

One of the primary causes of network vulnerability is outdated software. When software is not updated, it can contain security flaws that hackers can exploit. For example, the WannaCry ransomware attack in 2017 was able to infect hundreds of thousands of computers worldwide because they were running outdated versions of Microsoft Windows.

Another common cause of network vulnerability is weak passwords. Many people still use passwords that are easy to guess, such as "123456" or "password." This makes it easy for hackers to gain access to sensitive information. Similarly, when employees reuse the same passwords across multiple accounts, a single compromised password can lead to a domino effect of security breaches.

Human error is another significant cause of network vulnerability. Employees can accidentally download malware, click on phishing links, or leave their devices unlocked and unattended, providing an opportunity for cybercriminals to access the network. In some cases, employees may also intentionally engage in malicious behavior, such as stealing data or selling access to the network.

To mitigate the risks of network vulnerability, organizations must take proactive steps to secure their networks. This includes regularly updating software, implementing strong password policies, providing employee training on cybersecurity best practices, and conducting regular security audits. Additionally, organizations can implement security measures such as firewalls, intrusion detection systems, and access controls to limit access to sensitive data.

In conclusion, network vulnerability is a serious threat that can lead to data breaches, financial losses, and reputational damage. It is caused by a variety of factors, including outdated software, weak passwords, and human error. Organizations must take proactive steps to secure their networks and implement security measures to reduce the risks of cyberattacks. By staying vigilant and staying ahead of potential threats, organizations can protect their networks and the sensitive information they contain.

3.2 Problem Formulation

While the giant component in section 2.2 provides us with information on the capacity of the largest networked system amidst causal failures, we are also interested in the number of non-connected components (aka small components). The number of small components shows how much the network is disconnected given causal failures. We have parameterized this to see the minimum number of causal failures that can be applied that break the network into components. Hence, we have:

Problem 3.1 (Small Connected Components)**.** *Given a network* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$*, an integer* $k > 0$*, and causalities* $\mathcal{C} = \{C_i : \mathcal{V}_{1,i} \rightarrow \mathcal{V}_{2,i}, i = 1, \ldots, h\}$*, find a minimum size set* $\bar{\mathcal{C}} \subseteq \mathcal{C}$ *of causal failures whose applications leave a set of* **at least** k *connected components in the remaining graph.*

Problem 3.1 identifies causalities that would significantly damage the network, evaluating how weak the network is (i.e., its *vulnerability*). Naturally, if failed nodes result in a disconnected network with many components, there must be some vital nodes in the failed ones. Finding a minimum number of casualties is to find vital nodes that significantly affect the network's connectivity.

3.3 NP-Hardness Proof

We prove that Problem 3.1 is NP-hard by showing that its decision version is NP-complete. The decision version seeks to know, given $\mathcal{G}$, $\mathcal{C}$, k, and s, if there exists a $\mathcal{C}' \subseteq \mathcal{C}$ of size at most s such that the application of all causalities in $\mathcal{C}'$ leaves us a graph with at least k connected components. We refer to this decision version as the *VkCS* problem.

Theorem 3.1. *The decision problem VkCS is NP-complete.*

Proof. Clearly, the decision version is in NP. Then we consider *Vertex k-cut (VkS)* problem defined as: given an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a positive integer k, find a $\mathcal{C} \subseteq \mathcal{V}$ of minimum size $|\mathcal{C}|$ such that $\mathcal{G} \setminus \mathcal{C}$ has at least k connected components. We refer to such a $\mathcal{C}$ as a $k-$separator. This problem is known to be NP-hard (28) for fixed values of k. The decision version of this problem asks that given G, k and s, if there exists a $\mathcal{C} \subseteq \mathcal{V}$ satisfying (i) such that $|\mathcal{C}| \leq s$.

We exhibit a polynomial-time computable and answer-preserving reduction ϕ from instances of *VkS* into instances of *VkCS*. Given an instance $I = (\mathcal{G}, k, s)$ of *VkS*, ϕ maps I to an instance $I' = (\mathcal{G}', \mathcal{C}, k, s)$ of *VkCS* and is defined as follows. $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ where $\mathcal{V}' = \{u_1, u_2 : u \in \mathcal{V}\}$, $\mathcal{E}' = \{(u_1, v_1), (u_1, v_2), (u_2, v_1), (u_2, v_2) : (uv) \in \mathcal{E}\}$ and $\mathcal{C} = \{(u_1 \rightarrow u_2) : u \in \mathcal{V}\}$. It is easy to see that $\mathcal{G}$ has a k-separator of size at most s if and only if $\mathcal{G}'$ admits a subset $\mathcal{C}' \subseteq \mathcal{C}$ of size at most s whose application leaves us with a graph having at least k connected components. This reduction shows that *VkCS* is NP-complete. □

3.4 Algorithms

3.4.1 MILP for Problem

We provide two versions of the MILP formulation for the problem that seeks to minimize the number of causalities whose application leaves us with at least k connected components. The first one is listed as follows:

$$
\begin{aligned}
minimize \quad & \sum_{C \in \mathcal{C}} z_C && (3.1)\\
s.t. \sum_{i \in K} x_v^i &\leq 1, v \in \mathcal{V} && (3.2)\\
x_u^i + \sum_{j \in K \setminus \{i\}} x_v^j &\leq 1, i \in K, uv \in \mathcal{E} && (3.3)\\
\sum_{v \in \mathcal{V}} x_v^i &\geq 1, i \in K && (3.4)\\
\sum_{i \in K} x_u^i + z_C &\leq 1, C \in \mathcal{C}, u \in \mathcal{V}(C) && (3.5)\\
1 &\leq \sum_{i \in K} x_u^i + \sum_{C : u \in \mathcal{V}(C)} z_C, u \in \mathcal{V} && (3.6)\\
\sum_{i \in K} x_u^i &= 1, u \in \mathcal{V} \setminus \bigcup_{C \in \mathcal{C}} \mathcal{V}(C) && (3.7)\\
z_C, x_v^i &\in \{0,1\}, C \in \mathcal{C}, i \in K, v \in \mathcal{V} && (3.8)
\end{aligned}
$$

The above formulation is inspired by the work of Cornaz et al. (28) that aims to determine a vertex subset whose removal disconnects the graph in at least k components. Our focus is on minimizing the number of causalities. We use K to denote the set $\{1, \ldots, k\}$, representing k copies of all nodes in $\mathcal{V}$. As usual, $\mathcal{C}$ denotes the set of all causalities that are potentially effected. The description of the equations is listed as follows:

- Eq. (3.2): These constraints ensure that at most one copy of each node is selected.

- Eq. (3.3): For each edge (u, v), this inequality guarantees that for every $i \neq j$, the i-th copy of u and the j-th copy of v will not be chosen together.

- Eq. (3.4): These constraints ensure that each copy of the graph has at least one node in the final solution, thereby getting at least k connected components.

- Eq. (3.5): These constraints ensure that no copy of any vertex belonging to a chosen (for application) causality is chosen, thereby effectively forbidding that vertex from becoming a member of any connected component in the remaining graph.

- Eq. (3.6): These constraints ensure that for any vertex u either some causality containing u is chosen or that the vertex becomes a member of some connected component in the remaining graph.

- Eq. (3.7): These constraints ensure that those nodes that do not belong to any causality have to be in the remaining network.

Complexity analysis: this formulation contains h variables (i.e. $z_C, C \in \mathcal{C}$) indicating whether each causality is applied and nk variables (i.e. $x_v^i, i = 1, 2, \cdots, k, v \in \mathcal{V}$) indicating whether one node in one copy is selected. As all variables are binary, we have a solution space with 2^{h+nk} points. Hence, the overall computational complexity is $O(2^{h+nk})$.

We also provide another formulation for our problem that is based on the multi-source one-terminal flow network problem.

$$
\begin{aligned}
minimize \quad & n^2 \left(\sum_{C \in \mathcal{C}} z_C \right) - \sum_{i \in \mathcal{V}} z_i && (3.9)\\
s.t. \sum_{l \in K} \sum_{j \in \mathcal{V}} x_{s_l j} &= n - \sum_{i \in \mathcal{V}} z_i && (3.10)\\
\sum_{l \in K} \sum_{j \in \mathcal{V}} x_{s_l j} &= \sum_{j \in \mathcal{V}} x_{jt} && (3.11)\\
\sum_{j \in \mathcal{V}} y_{lj} &= 1,\ l \in K && (3.12)\\
\sum_{l \in K} x_{s_l i} + \sum_{j:(j,i) \in \mathcal{A}} x_{ji} &= \sum_{j:(i,j) \in \mathcal{A}} x_{ij} + x_{it},\ i \in \mathcal{V} && (3.13)\\
x_{s_l j} &\leq y_{lj} n,\ l \in K, j \in \mathcal{V} && (3.14)\\
y_{lj}, x_{jt} &\in \{0,1\},\ l \in K, j \in \mathcal{V} && (3.15)\\
x_{s_l j}, x_{ij} &\in \mathbb{N},\ l \in K, (i,j) \in \mathcal{A} && (3.16)\\
x_{ij} &\leq (1 - z_C) n,\ C \in \mathcal{C}, \\
& \quad i \in \mathcal{V}(C), (i,j) \in \mathcal{A} && (3.17)\\
x_{ji} &\leq (1 - z_C) n,\ C \in \mathcal{C}, \\
& \quad i \in \mathcal{V}(C), (j,i) \in \mathcal{A} && (3.18)\\
x_{s_l j}, x_{jt} &\leq (1 - z_C) n,\ C \in \mathcal{C}, \\
& \quad j \in \mathcal{V}(C), l \in K && (3.19)\\
z_i &\geq z_C,\ i \in \mathcal{V}(C),\ C \in \mathcal{C} && (3.20)\\
z_i, z_C &\in \{0,1\},\ i \in \mathcal{V},\ C \in \mathcal{C} && (3.21)
\end{aligned}
$$

The description of the above formulation is listed as follows:

- Eq. (3.9): problem 3.1 aims to find the minimum number of applied causalities, so the objective function is minimized;

- Eqs. (3.10), (3.20): these two equations ensure that each vertex that is part of a causality effected is not considered for the calculation of the remaining number of vertices. However, a vertex i that is not part of any applied causality can also be removed from consideration in (22). To avoid this, we include them as part of the objective function with a -1 coefficient. To make sure that only the number of effected causalities is determined, we multiply each z_C with a large ($= n^2$) multiplicative factor so that the optimum value is essentially determined by the number of applied causalities.

- Eq.(3.11): the flow going to the terminal t is equal to n minus the number of failed nodes;

- Eq.(3.12-3.19): similar to the formulation of Problem 2.1.

Complexity analysis: this formulation has $k + 2n + nk$ binary variables, i.e. h variables (i.e. $z_C, C \in \mathcal{C}$) indicating whether a causality is applied, n variables (i.e. $x_{it}, i \in \mathcal{V}$) indicating whether a node $i \in \mathcal{V}$ ships 1 unit of flow to t, n variables (i.e. $z_i, i \in \mathcal{V}$) indicating that whether each node i is in the remaining network and nk variables (i.e. $y_{lj}, l \in K, j \in \mathcal{V}$) indicating whether a node $j \in \mathcal{V}$ receives at most n unit of flow from s_{lk}. Moreover, there are $2m + nk$ integral variables ranging from 1 to n, which are $2m$ variables (i.e. $x_{ij}, x_{ji}, ij \in \mathcal{E}$) showing the flow from nodes i to j or from j to i and $x_{s_l j}, j \in \mathcal{V}$ representing the flow from s_l to node j. Hence, the overall computation complexity is $O(2^{nk+k} n^{m+nk})$.

Algorithm 2 GREEDY-MIN-SC

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, constant k;
Output: minimum set of causalities $\mathcal{MC}$, s.t. when all causalities in $\mathcal{MC}$ are applied to $\mathcal{G}$, there exists at least k connected components.

1: $\mathcal{H} \leftarrow \mathcal{G}$;
2: $S \leftarrow 1$//the number of connected components;
3: $\mathcal{MC} \leftarrow \emptyset$;
4: **while** $S < k$ **do**
5: Select $C \in \mathcal{C}$ to maximize S;
6: $\mathcal{MC} \leftarrow \mathcal{MC} \cup C$;
7: $\mathcal{C} \leftarrow \mathcal{C} - C$;
8: $\mathcal{H} \leftarrow$ apply C to $\mathcal{H}$;
9: $S \leftarrow$number of connected components of $\mathcal{H}$;
10: **end while**
11: **return** $\mathcal{MC}$;

3.4.2 A Heuristic Algorithm

Different from Problem 2.1, Problem 3.1 deals with determining the vital nodes whose failures have the greatest impact on the connectivity of other nodes. We design a greedy algorithm and provide an example as follows.

Complexity analysis: Step 6 takes $O(hn^2)$ after all causalities are checked. The worst case is that $S \geq k$ after applying all causalities. Hence, the total complexity is $h \cdot O(hn^2) = O(h^2n^2)$.

Approximation ratio: algorithm 2 greedily looks for the causality that causes a significant increase in the number of connected components. The extreme situation is that algorithm 2 finds $\geq k$ connected components by applying all causalities, i.e. $|\mathcal{MC}| = h$, while the optimal solution is $|\bar{\mathcal{C}}| = 2$. The reason is that if $|\bar{\mathcal{C}}| = 1$, then the specific causality would be found by algorithm 2. For this case, we have $|\mathcal{MC}|/|\bar{\mathcal{C}}| \leq h/2$.

Example 3.1. *Given causalities* $C_1 : \{1,4\} \rightarrow \{9,12\}, C_2 : \{5\} \rightarrow \{11,16\}, C_3 : \{3\} \rightarrow \{7\}$*, and* $k = 2$*, the minimum set of causalities for Fig. 1.1 is calculated as follows.*

1: $\mathcal{H} \leftarrow \mathcal{G}$*;*

2: $S \leftarrow 1$*;*

3: $\mathcal{MC} \leftarrow \emptyset$*;*

4: (While loop:);

5: Apply C_2 *to* $\mathcal{H}$ *and* $S = 3 > 2$*;*

6: (End while);

7: Output: $\mathcal{MC} = \{C_2\}$*.*

The output $\mathcal{MC} = \{C_2\}$ *reveals that applying one causality is enough to make the network disconnected with 3 components. Hence, nodes* $\{5, 11, 16\}$ *have a great impact on the connectivity of the network.*

3.5 Experimental Results

3.5.1 An Illustrative Example

We use the same experimental settings in section 2.6.1 and show the first trial in Fig. 3.1 and Table 3.1, which compares the performance of GREEDY-MIN-SC and the corresponding optimization model. In terms of the number of causalities, GREEDY-MIN-SC performs as well as the optimization model. The reason is that the very first causality found by GREEDY-MIN-SC contains cut vertices, and its application disconnects the network significantly. In addition, MILP costs much more time than GREEDY-MIN-SC, especially when $k = 10$.

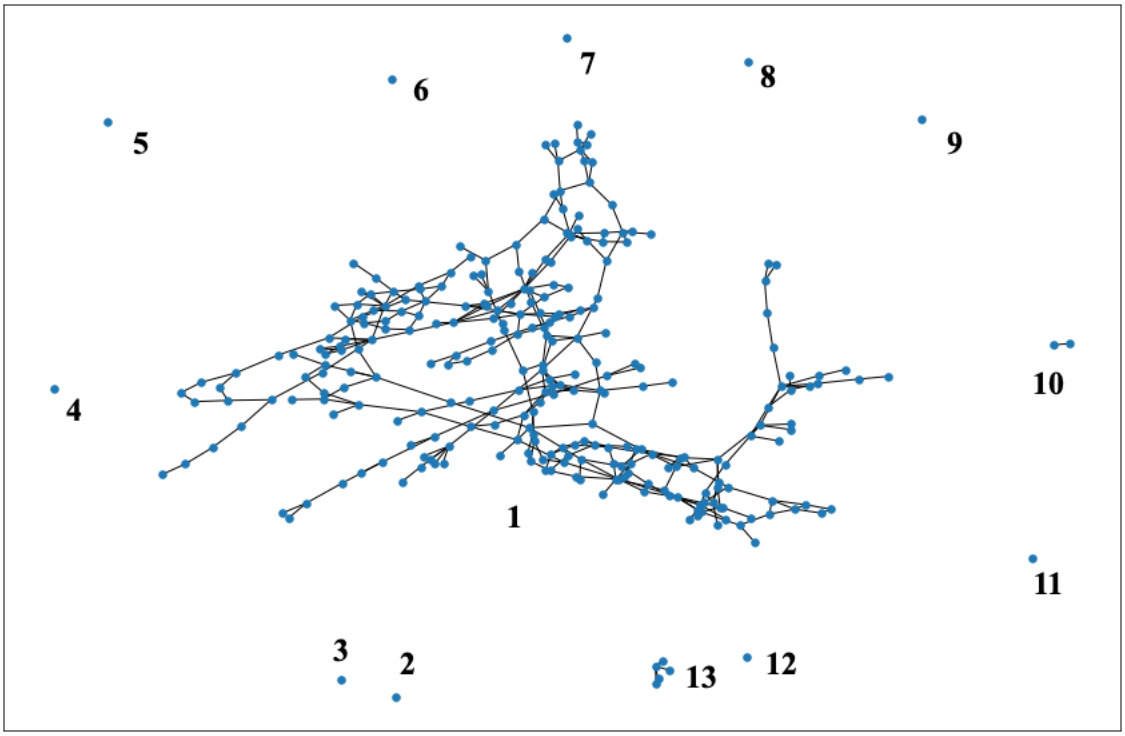


Figure 3.1: IEEE 300: remaining network with $k = 13$ components

Table 3.1: IEEE 300: first trial of small components problem

	GREEDY-MIN-SC		MILP	
k	$\lvert\mathcal{MC}\rvert$	t_h(s)	opt	t_o(s)
2	1	0.11	1	0.08
3	1	0.06	1	0.27
4	1	0.04	1	0.35
5	1	0.04	1	0.58
6	1	0.04	1	2.61
7	1	0.04	1	1.09
8	1	0.04	1	1.48
9	1	0.04	1	1.85
10	2	0.09	2	18.91

3.5.2 Investigation of Connected Components of IEEE 300

Table 3.2 shows the common trend that applying more causalities could generate more connected components. More specifically, the results in Table 3.2 are in agreement in that GREEDY-MIN-SC is able to detect the causalities that cause severe failures in the network. For example, 8 experiments have found one causality whose application disconnects the network into at least 10 connected components. From the perspective of the number of nodes, by randomly removing 6 nodes on average, we are able to disconnect the network into at least 10 connected components. GREEDY-MIN-SC is looking for causalities whose removal generates the greatest number of connected components greedily, so the number of nodes in the applied causality should be higher than the expected value.

Table 3.2: Comparison of networks for small components

k	IEEE300	Power1138	power5000
2	1	1	1
3	1	1	1
4	1	1	1
5	1	1	1
6	1	2	1
7	1	2	1
8	1	2	2
9	1	2	2
10	1	2	2

Table 3.3: Scalability analysis for small components

Number of causalities	GREEDY-MIN-SC						Optimization-2					
	IEEE300		Power1138		power5000		IEEE300		Power1138		power5000	
	$\|\mathcal{MC}\|$	t_h(s)	$\|\mathcal{MC}\|$	t_h(s)	$\|\mathcal{MC}\|$	t_h(s)	opt	t_o(s)	opt	t_o(s)	opt	t_o(s)
10	3	0.19	1	0.08	3	0.72	3	6.46	1	653.74	3	$>$ 3h
20	1	0.04	2	0.27	2	0.95	1	8.07	2	902.25	2	$>$ 3h
30	1	0.06	2	0.37	2	1.35	1	11.33	2	>2h	2	$>$ 3h

3.5.3 Scalability Analysis

To analyze the scalability of GREEDY-MIN-SC and MILP, we utilize the same experiment settings as section 2.6.3. Specifically, $k = 10$ for IEEE 300, $k = 40$ for power network 1138-bus, and $k = 80$ for Western US Power Grid. In Table 3.3, we can also see the advantage of GREEDY-MIN-SC over MILP formulation. For example, if the number of predefined causalities is 20, the running time of GREEDY-MIN-SC is 0.04 seconds, 0.27 seconds, and 0.95 seconds, respectively for the power network while MILP takes 8.07 seconds, 902.25 seconds, and > 3 hours given that both methods obtain the same results. An interesting observation between tables 2.3 and 3.3 is that MILP formulation for problem 3.1 takes much more time than MILP for problem 2.1. Recalling the computational complexity of both programs, we know that MILP for problem 2 generates k copies of all nodes in the network if we use the first formulation of MILP for problem 2. As a result, the number of variables is much more than that of MILP for problem 1. Therefore, by checking so many more variables, MILP for problem 2 costs much more time.

3.6 Conclusion

In this chapter, we consider the impact of causal node failures on the vulnerability of one network, which is measured as the minimum number of applied causalities while the network has at least k connected components. We formally formulate the problem and prove the NP-hardness of the problem. Correspondingly, a mixed integer linear programming model is proposed to obtain the exact solution and a heuristic algorithm is designed to approximate the solution. Experimental results based on the IEEE 300 Bus, Power1138, and the Western US Power Grid system show that the

heuristic algorithm can be considered as an alternative to the optimization model to substantially reduce the computational time.

Chapter 4

Length Constrained Shortest Paths under Cascading Causal Node Failures

4.1 Introduction

The shortest paths have a widespread application in engineering systems (29; 30; 31). For a power system, (32) proposed a new approach based on shortest paths to analyze whether a contingency creates a saturated cutoff in a power system when considering the problem of identifying transmission lines with a limited capacity if multiple disturbances occur in the system. For the singularity-induced voltage instability in a power network, (33) searched for the shortest path to its boundary. The arc length of the path is used to formulate the problem and is converted into a control framework to solve the shortest path manifold. (34) pointed out the importance of designing appropriate management, measurement, and communication infrastructures for a wide-area measurement system. Especially, the authors utilized dynamic multiobjective shortest-path programming for optimal communication infrastructure design. For the virtualization of network functions, to find a suitable service path that optimizes a specific target, (35) formulated a capacitated integer linear programming model based on the shortest path tour problem for service chaining and function placement, and showed that the performance of the proposed model is better than the current state model.

Constraints on the length of the path require that the distance between any pair of nodes be at most a given integer D. As we know, the diameter of a graph is the greatest distance between any pair of nodes, so a constraint on path length is also called a diameter constraint. A diameter constraint has been widely used in many networks, especially when evaluating network reliability. (36) introduced a diameter constraint on source-to-terminal reliability in a communication network, where each link fails with a given probability. They integrated the constraint with a factoring algorithm (37) and illustrated the computation gain in terms of topological reduction when computing the reliability. In a transportation network, which is often modeled as a flow network, (38) designed an approximation algorithm for multistate two-terminal reliability using a diameter constraint of st paths as a tuning parameter. Moreover, due to signal attenuation problems (39), wavelength division multiplexing networks often require a diameter constraint. Similarly, a hop constraint is also useful in virtual topology and routing design (40; 41; 42).

In this chapter, we mainly consider the impact of cascading causal failures on the paths between a set of source and terminal nodes. Our contributions are listed as follows:

- We formally define cascading causal failures. If the nodes in $\mathcal{V}_1 \subseteq \mathcal{V}$ (V is the network node set) fail, all the nodes in $\mathcal{V}_2 \subseteq \mathcal{V}$ will fail. The nodes in $\mathcal{V}_1$ do not need to be adjacent to the nodes in $\mathcal{V}_2$. Furthermore, if $\mathcal{V}_3 \subseteq \mathcal{V}_1 \cup \mathcal{V}_2$ and the failure of the nodes in $\mathcal{V}_3$ causes the failure of the nodes in $\mathcal{V}_4$, then the failure of $\mathcal{V}_1$ leads to cascading causal failures.

- We define the closure of a subset $\mathcal{U}$ of the node set as the set of nodes whose failures are triggered by the failure of $\mathcal{U}$.

- We consider the impact of causal node failures on the shortest paths of a pair of nodes s, t in the network $\mathcal{G}$ and formulate a problem to find the maximum number of causal failures applied by restricting the lengths of the paths st to at most $\alpha \times d_{\mathcal{G}}(s,t)$.

- We provide an NP-hardness proof of the formulated problem.

- We present an integer linear programming model to find the solution to the proposed problem. We also design a greedy algorithm to find an approximate solution and compare the performance of the greedy algorithm with the optimization model.

The rest of this chapter is organized as follows. Section 4.2 shows the system model of this paper. We provide an NP-hardness proof of the proposed problem in Section 4.3 and the corresponding integer linear programming model, and a heuristic algorithm in Section 4.4. We run the optimization model and the heuristic algorithm on several networks in Section 4.5. Finally, Section 4.6 concludes this chapter.

4.2 Problem Formulation

Example 4.1. *In Fig. 4.1(a), we use dotted circles and dotted squares to denote the nodes in V_1 and V_2 of a causality C, respectively. The given causalities are C_1 : $\{1,4\} \rightarrow \{9,12\}$, C_2 : $\{4,9\} \rightarrow \{15\}$, C_3 : $\{5\} \rightarrow \{11,16\}$, C_4 : $\{3\} \rightarrow \{7,10\}$, $C_5 : \{15\} \rightarrow \{13\}$.*

If we apply C_3 to the network, the failure of node 5 causes the failure of nodes 11 and 16. Fig. 4.1 (b) presents the remaining network with three connected components (i.e. $\{1,2,3\}$, $\{4,7\}$, and $\{6,8,9,10,12,13,14,15\}$).

If C_1 is applied to the network, then nodes $\{1, 4, 9, 12\}$ are removed from the network. As $\{4, 9\}$ is on the left side of C_2, node 15 fails. Furthermore, the failure of node 15 causes the failure of node 13, according to C_5. Therefore, we observe a cascade of causal failures C_1, C_2, C_5, and Fig. 4.1 (c) shows the remaining network consisting of two connected components (that is, $\{2, 3, 5, 6, 8, 10, 14, 16\}$ and $\{7, 11\}$) after applying C_1, C_2, C_5.

From Example 4.1, we observe cascading failures in the causal model. Therefore, we formally define cascading causal failures.

Definition 4.1. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, if the application of a causality $C \in \mathcal{C}$ triggers the subsequent application of other causalities, we say that cascading causal failures are applied to the network.*

Def. 4.1 generalizes causal failures to the cascading case if the information on the dependency between nodes is known. Naturally, given a group of failed nodes, it is necessary to obtain all failed nodes triggered by the failures of the group of nodes based on causal failures. Therefore, we define the closure of a group of nodes.

Definition 4.2. *Given a set of causalities $\mathcal{C}$, the closure of a set of nodes $\mathcal{U}$ with respect to $\mathcal{C}$, denoted by $\mathcal{U}_{\mathcal{C}}^{+}$, is the set of all nodes (including those in $\mathcal{U}$) whose failures are triggered by causalities in $\mathcal{C}$ upon removal of nodes in $\mathcal{U}$. The cascading set $\mathcal{C}(\mathcal{U})$ is the set of causalities in $\mathcal{C}$ that are automatically triggered (in a cascading fashion) as a result of the failure of the nodes in $\mathcal{U}$.*

The closure plays an important role in cascading causal failures assuming the time delays for cascading effects to be negligible while studying the performance of the network. An equivalent algorithmic definition of $\mathcal{U}_{\mathcal{C}}^{+}$ and $\mathcal{C}(\mathcal{U})$ is the output of the following algorithm, the computational complexity of which is $O(h^2)$.

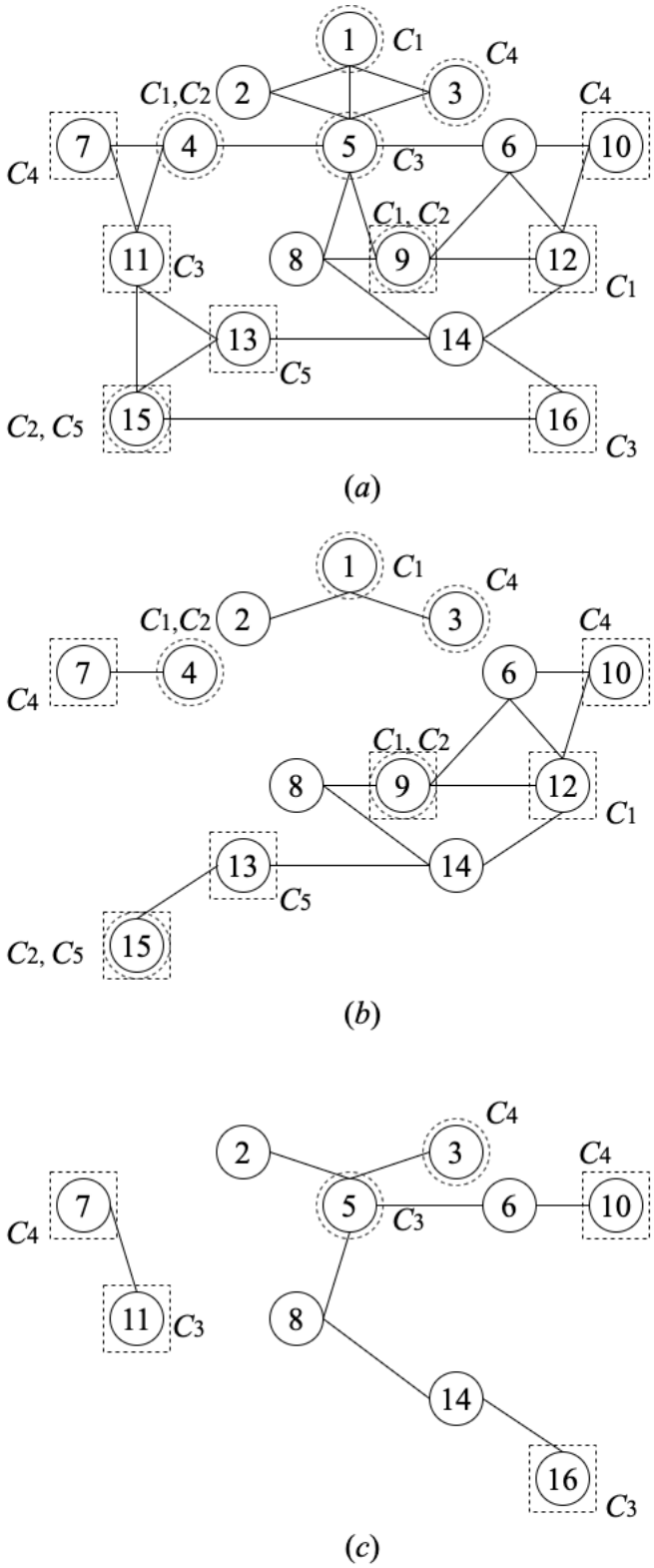


Figure 4.1: Remaining networks after applying different causalities.

Algorithm 3 Closure

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, $\mathcal{U} \subseteq \mathcal{V}$;
Output: $\mathcal{U}_{\mathcal{C}}^{+}$, $\mathcal{C}(\mathcal{U})$.

1: $\mathcal{U}' \leftarrow \mathcal{U}$, $\mathcal{S} \leftarrow \emptyset$, $\mathcal{D} \leftarrow \mathcal{C}$;
2: **while** $\exists C_i \in \mathcal{D}$ such that $\mathcal{V}_{1,i} \subseteq \mathcal{U}'$ **do**
3: $\quad \mathcal{U}' \leftarrow \mathcal{U}' \cup \mathcal{V}_{2,i}, \mathcal{S} \leftarrow \mathcal{S} \cup \{C_i\}, \mathcal{D} \leftarrow \mathcal{D} \setminus \{C_i\}$
4: **end while**
5: **return** $\mathcal{U}'$ and $\mathcal{S}$.

Given a source node s and a terminal node t, it is well known that node failure may increase the distance between them (38). More specifically, the more node failures the network has, the more likely the distance between s and t would increase.

Example 4.2. *In Fig. 1.1, let $s = 2$ and $t = 14$, then $d_{\mathcal{G}}(2, 14) = 3$. If causality C_1 is applied to $\mathcal{G}$, then nodes 1, 4, 9, 12, 15, and 13 will be removed from $\mathcal{G}$, but the distance between 2 and 14 is still 3. However, if we apply the causal failure C_3, then node 2 is not reachable from node 14 because node 5 is removed.*

Example 4.2 presents the difference in causalities with respect to their impact on connectivity. As alluded to in Section 1, the lengthening of the shortest paths can adversely affect certain networks (e.g., the distance between nodes affects the efficiency of power transmission in the power network). Therefore, it is interesting to investigate how the number of causal failures applied affects the shortest path from s to t. Let $D_0 = d_{\mathcal{G}}(s, t)$, then we have the following problem.

Problem 4.1. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, where causality set $\mathcal{C} = \{C_i : \mathcal{V}_{1,i} \Rightarrow \mathcal{V}_{2,i}, i = 1, 2, \cdots, h\}$ (containing possible cascading failures), $s, t \in \mathcal{V}$, and a constant α, find a* **maximum** *subset of causalities whose application maintains the distance between s and t to be at most $\alpha \times D_0$.*

Problem 4.1 investigates the impact of cascading causal failures on the distance between a given pair of nodes. This is a good metric to measure the connectivity of a

pair of nodes in terms of the maximum number of causalities they can withstand. Let $D = \alpha \times D_0$, and we shall use D and α interchangeably below.

4.3 NP-Hardness Proof

The problem 4.1 is also applicable to edge-weighted networks because it focuses on the length of *st* paths. To prove its hardness, we start with the case where the edges are not weighted and there are no cascading failures and then extend the unweighted case to the weighted case.

We consider the decision version of Problem 4.1 with unweighted edges and no cascading failures, denoted as *VfCD*: given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, where causalities $\mathcal{C} = \{C_i : V_{1,i} \Rightarrow V_{2,i}, i = 1, 2, \ldots, h\}$, $s, t \in \mathcal{V}$, $D_0 = d_{\mathcal{G}}(s, t)$, and a constant α, let $D = \lfloor \alpha \times D_0 \rfloor$, whether there exists $\mathcal{C}' \subseteq \mathcal{C}$ with size at least f such that applying causalities in $\mathcal{C}'$ maintains $d_{\mathcal{G}}(s, t) \leq D$.

To prove that this version is NP-complete, we utilize the decision version of the MAX-2SAT problem (18), which is described as: given 2-clauses $c_1, c_2, \ldots, c_p$, whether there is a truth assignment satisfying at least f clauses. This problem is NP-complete.

Theorem 4.1. *VfCD is NP-complete.*

Proof. First, *VfCD* is in NP because one could verify whether $d_{\mathcal{G}}(s, t) \geq D$ in polynomial time given a certificate of *VfCD*.

Then we perform a polynomial-time reduction from an instance I of MAX-SAT to an instance I' of *VfCD* as follows:

1. We create the source node s. For each clause $c_i = x_a \vee x_b, i = 1, 2, \ldots, h$, we create nodes x_a, x_b, three nodes $c_i, c_{i,1}, c_{i,2}$ and edges (x_a, c_i), $(x_a, c_{i,1})$, (x_b, c_i), $(x_b, c_{i,2})$, $(c_i, c_{i,1})$, $(c_i, c_{i,2})$. Moreover, we add edges (s, x_a) and (s, x_b) if x_a and x_b are in clause c_1.

2. For each literal $x_j, j = 1, 2, \ldots, l$, nodes X_j, $\bar{X}_j$, and dummy node d_j are created, with edges (d_j, X_j), $(d_j, \bar{X}_j)$, (X_j, d_{j+1}), $(\bar{X}_j, d_{j+1})$. In particular, when $j = 1$, we add edges $(c_{h,1}, d_1)$, $(c_{h,2}, d_1)$; when $j = l$, let $t = d_{l+1}$.

3. For each clause $c_i = x_a \vee x_b$, causalities are defined as $C_a : \{x_a\} \Rightarrow \{X_a, c_{i,2}\}$ and $C_b : \{x_b\} \Rightarrow \{X_b, c_{i,1}\}$.

4. $D = 3p + 2l + 1$.

Suppose that I is satisfiable for a given assignment, then we have at least f satisfied clauses. We pick one "True" literal for each satisfied clause and apply the corresponding causality in the constructed graph. The distance between s and t is at most D. Therefore, I' is also satisfiable.

Suppose that I is not satisfiable for all possible assignments, which means that at most $f-1$ clauses are satisfied. We prove that there are at most $f-1$ causalities applied by contradiction. Suppose that at least f causalities are applied to the network; then, based on the assumption of the construction method, the applied causalities must be of different causalities; otherwise, s is not connected to t. Furthermore, the applied causalities should not contain both x and be negative; otherwise, s is not connected to t. Therefore, the causalities applied to $\geq f$ would lead to the fact that s is not connected to t, which means that the distance between s and t is infinity ($\geq D$). Therefore, at most $f - 1$, causalities should be applied to the network. □

Fig. 4.2 illustrates the construction method in the proof of Theorem 4.1. The instance of MAX-2SAT is $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$. The corresponding instance of *VfCD* without cascading effects is a graph with 23 nodes and 6 causalities.

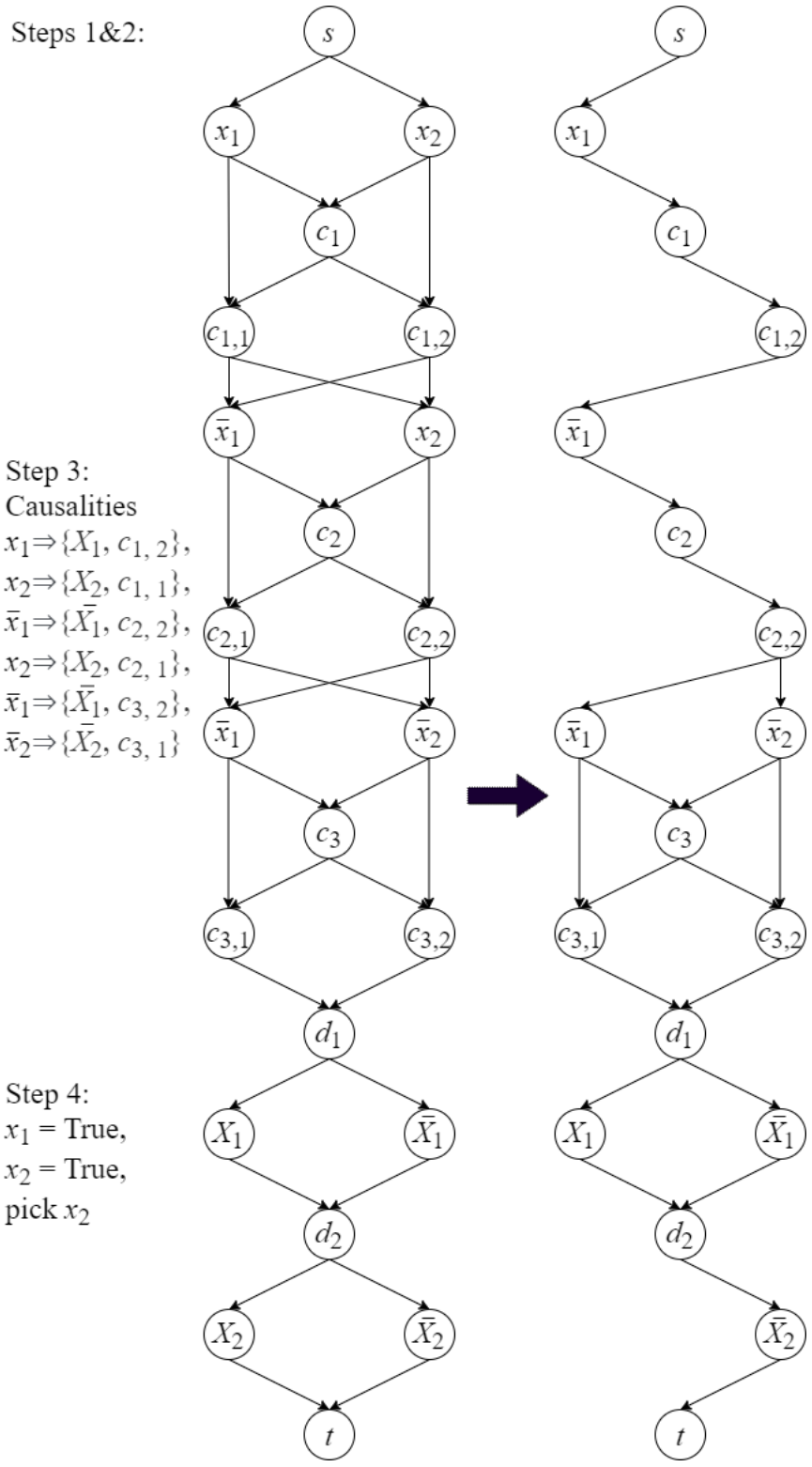


Figure 4.2: Example depiction of Theorem 4.1.

Theorem 4.1 presents the hardness of problem 4.1 with unweighted edges. We can consider unweighted edges with a uniform weight (e.g., 1). As a result, *VfCD* is a special situation for the edge-weighted case. Then we could easily know:

Corollary 4.1. *VfCD with weighted edges is NP-complete.*

Furthermore, Theorem 1 and Corollary 1 establish the NP-completeness of a restricted version of *VfCD* where there are no cascading failures. This establishes the NP-completeness of the decision versions of Problem 1 with cascading failures. Therefore, we have the following:

Corollary 4.2. *VfCD with weighted edges and cascading failures is NP-complete.*

4.4 Algorithms

4.4.1 MILP for the Problem

We propose a mixed integer linear programming (MILP) formulation to find the maximum number of causal failures that result in the shortest path of at most $D = \alpha \times D_0$. We consider the directed version $\mathcal{DG} = (\mathcal{V}, \mathcal{A}, \mathcal{C})$ of the undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, where $\mathcal{A}$ is generated by replacing each edge in $\mathcal{E}$ with two directed arcs in opposite

directions. Given the causal failures set $\mathcal{C}$ and the length constraint D, we have the formulation in Eqs. (4.1-4.9).

$$Maximize \quad \sum_{C \in \mathcal{C}} z_C \tag{4.1}$$

$$\sum_{ij \in \mathcal{A}} w_{ij} x_{ij} \leq D \tag{4.2}$$

$$x_{ij} \leq 1 - z_C, \quad C \in \mathcal{C}, \quad i \in \mathcal{V}_{1,C} \cup \mathcal{V}_{2,C}, (i,j) \in \mathcal{A} \tag{4.3}$$

$$x_{ji} \leq 1 - z_C, \quad C \in \mathcal{C}, \quad i \in \mathcal{V}_{1,C} \cup \mathcal{V}_{2,C}, (j,i) \in \mathcal{A} \tag{4.4}$$

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & if\ i = s \\ -1, & if\ i = t \\ 0, & otherwise \end{cases} \tag{4.5}$$

$$\sum_i x_{ij} \leq 1, \quad j \in \mathcal{V}, i \in \mathcal{V} - \{s,t\} \tag{4.6}$$

$$x_{ij}, x_{ji} \in \{0,1\}, \quad ij \in \mathcal{A} \tag{4.7}$$

$$z_C \in \{0,1\}, \quad C \in \mathcal{C} \tag{4.8}$$

$$1 - z_C \leq \sum_{i \in V_{1,C}} \sum_{ij \in \mathcal{A}} x_{ij}, C \in \mathcal{C} \tag{4.9}$$

This formulation aims to transmit one unit flow from source s to terminal t. Therefore, in Eq. (4.5), for the source, s, the sum of the outgoing flow is 1 higher than the incoming flow, while the total incoming flow for t is 1 higher than its total outgoing flow. Other nodes follow the flow conservation constraint. Because only one unit of flow is transmitted in the network, the total outgoing flow of each node $v \in V - \{s,t\}$ is at most 1 according to Eqs. (4.6) and (4.7). Regarding causality, C, the binary

variable $z_C = 1$ implies that causality C is applied to the network. Once C is applied, each edge incident on any node in C would not carry any flow, as governed by Eqs. (4.3) and (4.4). Consequently, the state of each edge would be 0 or 1, and the edges with state 1 definitely form an st path. Let w_{ij} be the weight of the edge $(i, j) \in \mathcal{E}$, and we require the length of the path to be at most D (in Eq. (4.2)) while maximizing the number of applied causalities (in Eq. (4.1)). Finally, Eq. (4.9) is the application of cascading causal failures, that is, causality C should be applied automatically if the sum of the flow of all nodes in $\mathcal{V}_{1,C}$ is 0.

4.4.2 A Heuristic Algorithm

Algorithm 4 Greedy-Max-Length

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, length constraint D, source node s and terminal node t;
Output: maximal set of causalities $\mathcal{MC}$, such that when all causalities in $\mathcal{MC}$ are applied to $\mathcal{G}$, there exists at least one st-path whose length is not greater than D.

1: Sort the set $\mathcal{C}$ based on $\theta(C)$ for all $C \in \mathcal{C}$ and let $\theta(C_1) \leq \theta(C_2) \leq \cdots \leq \theta(C_h)$ and update $\mathcal{C}$ as $\{C_1, C_2, \cdots, C_h\}$;
2: $i \leftarrow 1$, $\mathcal{C}' \leftarrow \varnothing$;
3: **while** $i \leq |\mathcal{C}|$ **do**
4: $\quad \mathcal{MC} \leftarrow \mathcal{MC} \cup C_i$;
5: $\quad$ Run Algorithm 1 to get the closure of $\mathcal{V}(\mathcal{MC})$, i.e., causality set $\mathcal{S}$ and nodes in $\mathcal{S}$, denoted as $\mathcal{U}$.
6: $\quad$ **if** s, t are not connected or $d_{\mathcal{G}}(s, t) > D$ in $\mathcal{G} - \mathcal{U}$ **then**
7: $\quad\quad \mathcal{MC} \leftarrow \mathcal{MC} - C_i$
8: $\quad\quad i \leftarrow i + 1$
9: $\quad\quad$ **Continue**
10: $\quad$ **else**
11: $\quad\quad \mathcal{G} \leftarrow \mathcal{G} - \mathcal{U}$
12: $\quad\quad \mathcal{C} \leftarrow \mathcal{C} - \mathcal{MC}$
13: $\quad$ **end if**
14: **end while**
15: **return** $\mathcal{MC}$

Given the NP-hardness of the formulated problem, we design a greedy strategy to solve the problem approximately in polynomial time. Let $\theta(C)$ be the distance between

s and t if we apply the causality C with its closure to $\mathcal{G}$, then we sort the causalities based on the corresponding values of θ in ascending order. In the sorted causalities, we add one causality and check the closure of the removed nodes and the distance between s and t. If the distance between s and t is less than or equal to D, we add the applied causalities to the result; otherwise, we ignore the present causality and continue to check the next one. The details of the algorithm are listed below.

Complexity Analysis: Step 1 actually computes the shortest st path that passes each node in each causality. In the while loop (Steps 3-13), the algorithm calculates the shortest path between s and t after iteratively applying the causalities. Therefore, the time complexity is $O(n^3)$.

Approximation Ratio: we consider the worst case. The first causality applied maintains $d_{\mathcal{G}}(s,t) \leq D$ while applying any of the other causalities would make $d_{\mathcal{G}}(s,t) > D$. The result of the algorithm would be 1. On the other hand, the optimal solution might apply all other causalities except the first one found by the proposed algorithm. Therefore, the optimal solution for the worst case is $h-1$.

4.5 Experimental Results

In this section, we demonstrate the performance of the proposed algorithm and the optimization model in experiments involving real-world networks. All programs are coded in Python and implemented on a Macbook with CPU M1 Pro and 16 GB memory. The graph-related programs refer to the NetworkX library (20), and the optimization model uses the Gurobi library (21).

4.5.1 Case Study I: IEEE 300 Network

We explore the connectivity of the IEEE 300 bus system, the topology of which is found in Fig. 4.3. We randomly generate 25 causalities as shown in Table 4.1. In each causality $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$, the number of nodes in $\mathcal{V}_1$ or $\mathcal{V}_2$ follows a discrete uniform distribution in $\{1, 2, 3, 4, 5\}$.

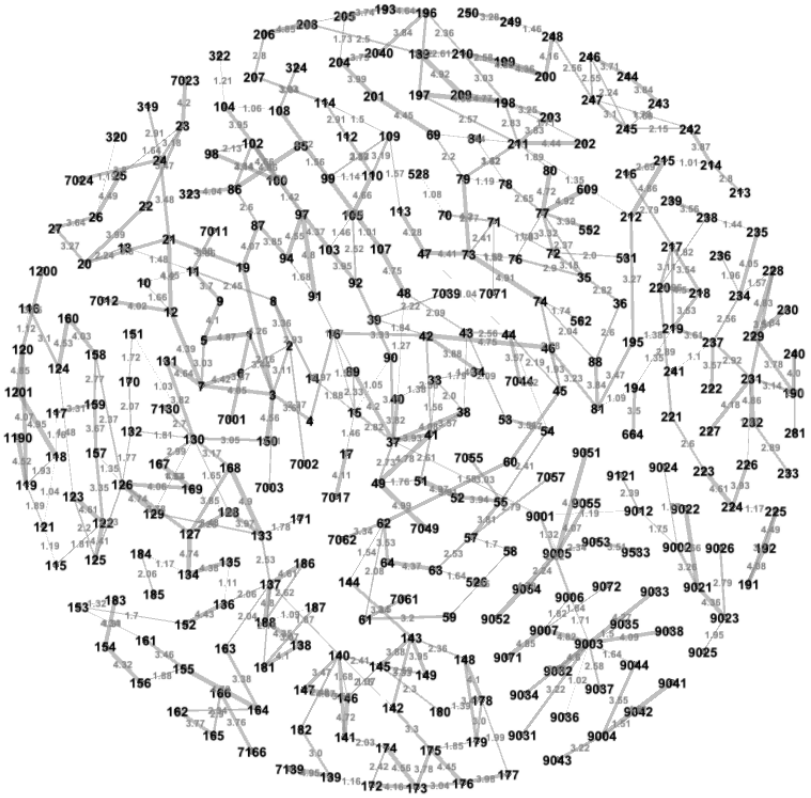


Figure 4.3: IEEE 300 network with synthetic edge weights.

4.5.1.1 Unweighted Network

We pick 4 pairs of terminal nodes, compare the performance of the proposed greedy algorithm with the optimization model and show the results in Table 4.2, where t_h and t_o denote the time consumed by the greedy algorithm and the optimization model,

Table 4.1: Causality definition generated randomly for the IEEE 300 network, where numbers in the causality sets represent node labels from the original network data set.

No.	$\mathcal{V}_1$	$\mathcal{V}_2$
1	136, 127	113
2	27, 90, 175	148, 107, 102
3	181	89, 137, 52
4	86, 9044	5, 171
5	7130, 42, 9036	25, 79, 198, 240
6	61, 9042, 188	552, 201, 41
7	226, 53, 104, 124	9044, 241, 228
8	243, 88, 203, 248, 176	526, 44, 102, 129
9	196, 117, 48, 39, 235	145
10	9055, 72	9053, 241, 224, 9026, 201
11	121, 126, 161, 178, 9037	9038, 122, 125, 72
12	526, 153, 9055	16, 225, 5, 109
13	108, 72, 179, 1201	188
14	160, 9035, 142, 60, 7139	140, 9023
15	214, 116, 22	103, 188, 9052, 182, 156
16	198, 9071, 237, 23, 166	9022
17	322, 214, 103, 205, 9037	2040, 9071
18	20	130, 72, 218, 128
19	4, 203, 90	64, 62, 204, 202
20	135	151
21	103, 211	53, 9023, 129, 115, 73
22	226, 7055, 169, 121	141, 229, 227, 120
23	25	9002
24	14, 158, 34, 27	224, 609
25	114	168, 156, 48, 9022, 45

respectively. In terms of running time, the greedy algorithm takes only 20% of the time cost by the optimization model to obtain the solution. However, for the IEEE 300 network, the running time of the optimization model is still acceptable, because the average would be around 0.14 seconds. For the number of causalities applied, we can see that the results of the greedy algorithm are close to those of the optimization model. For example, when $s = 3$ and $t = 7039$, both methods found the same number of causalities applied for $D = 6$. Moreover, for $s = 7166, t = 9533, D = 24$, the greedy algorithm found 19 applied causalities, 2 fewer than the optimization models. Interestingly, the results obtained from the optimization model are almost unchanged, although D is increasing. The reason is that the optimization model always tries to keep the shortest path from s to t unchanged and applies all causalities that do not contain nodes on the shortest st-path. However, this does not indicate that the problem formulated can be solved in polynomial time, unless we have prior knowledge that there is only one shortest path from s to t.

From the perspective of the removed nodes, the number of causalities applied indicates the expected number. Based on the assumption mentioned at the beginning of this subsection, the expected number of nodes in each causality is 6. For $s = 199, t = 9033, D = 20$, the number of causalities applied is 22, which means that removing $22 \times 6 = 132$ nodes on average could guarantee that the distance between node 199 and node 9033 is at most 20.

One might be interested in the difference between individual node failures and causal failures. Suppose that we have an IEEE 300 network with 132 nodes that will fail; we hope to find the maximum number of failed nodes such that the distance between s and t is at most D. Clearly, we need to check cases 2^{132} in the worst case. Taking into account causal failures, we know that the expected number of causalities is

Table 4.2: Experiments for unweighted IEEE 300

$\{s,t\}$	D_0	D	$\lvert\mathcal{MC}\rvert$	t_h(s)	OPT	t_o(s)
$\{3,7039\}$	5	6	21	0.03	21	0.17
		10			22	0.13
		18			22	0.13
$\{2,7061\}$	10	11	19	0.03	22	0.16
		13			23	0.13
		17			23	0.13
$\{199,9003\}$	13	16	20	0.03	22	0.16
		20				
		24				
$\{7166,9533\}$	19	20	19	0.03	21	0.16
		24				0.13
		28				0.13

$132/6 = 22$. In other words, we need to scan 2^{22} cases to obtain the maximum number of causalities in the proposed problem. This is a substantial computation gain if we consider failures as causalities.

4.5.1.2 Weighted Network

For the same network, we sample a real number between 1 and 5 for each edge and use the same source and terminal nodes as the unweighted network. Instead of setting D directly, we pick three values of α for each pair of source and terminal nodes. The results in Table 4.3 again illustrate the fact that the time spent by the optimization model is almost 5 times the time cost of the greedy algorithm. For example, given $s = 2, t = 7061, \alpha = 2.5$, the optimization model takes 0.16 seconds, while the greedy algorithm takes 0.03 seconds. Similar cases occur with $s = 3, t = 7039, \alpha = 1.5$ and

$s = 7166, t = 9533, \alpha = 1.5$. Furthermore, compared to Table 4.2, we could see that for the same terminal nodes, the running time of the unweighted version is close to that of the weighted version for both the greedy algorithm and the optimization model. This fact indicates that whether the edges are weighted does not affect the running time of both algorithms.

Table 4.3: Experiments for edge-weighted IEEE 300

$\{s, t\}$	D_0	α	$\lvert\mathcal{MC}\rvert$	t_h	OPT	t_o
$\{3, 7039\}$	13.97	1.5	21	0.03	22	0.15
		2.5				0.13
		3.5				0.13
$\{2, 7061\}$	32.89	1.5	19	0.06	23	0.15
		2.5		0.03		0.16
		4		0.03		0.13
$\{199, 9003\}$	35.65	1.5	20	0.04	22	0.17
		2.5		0.03		0.16
		4		0.03		0.16
$\{7166, 9533\}$	51.66	1.5	19	0.03	21	0.15
		2.5				0.13
		3.5				0.13

4.5.2 Case Study II: Power 5000

A power network with 4945 nodes and 6570 edges is generated using the method in (43) and introduced to investigate the scalability of the proposed algorithm. We also randomly generate 40 causalities. For each causality, $C : V_1 \Rightarrow V_2$, the number of nodes in V_1 or V_2 is sampled from a discrete uniform distribution between 1 and 60.

4.5.2.1 Unweighted Network

Each edge is weighted as 1 in this case. Four pairs of terminals are selected for experiments, and the comparison is shown in Table 4.4.

Table 4.4: Experiments for Unweighted Power 5000

$\{s,t\}$	D_0	D	$\lvert\mathcal{MC}\rvert$	t_h(s)	OPT	t_o(s)
$\{184,187\}$	8	10	36	0.70	36	20.37
		12		0.67	36	20.37
		16		0.70	37	20.10
$\{10,14\}$	13	14	37	0.73	37	20.73
		18		0.78		19.94
		22		0.70		20.67
$\{709,1314\}$	19	22	10	0.83	33	20.70
		26	12	0.84	33	20.47
		28	21	0.84	34	20.72
$\{1035,1200\}$	25	26	29	0.84	34	34.43
		30		0.80	34	31.02
		34		0.81	35	30.33

In terms of the number of causalities applied, we also observe the efficacy of the greedy algorithm. For example, given $s = 10, t = 14, D = 18$, both the greedy algorithm and the optimization model obtain 37 causalities. Similarly, given $s = 184, t = 187, D = 12$, we have 36 causalities applied using the greedy algorithm and the optimization model. An interesting situation is that the number of causalities applied remains unchanged regardless of the change of D in some experiments. For example, for $D = 14, 18, 22, s = 10, t = 14$, the number of causalities applied remains at 37. Obviously, most causalities do not affect the shortest path from node 10 to node 14.

From Table 4.4, we also observe the uncertainty of the proposed greedy algorithm. For $s = 709, t = 1314, D = 22$, the optimization model obtains 33 applied causalities, while the greedy algorithm finds 10. As we mentioned in Theorem 2, applying any causality except for the causalities found by the greedy algorithm causes the distance between node 709 and node 1314 to be greater than 22. However, this is very rare. From the above experiments, the greedy algorithm works in most cases.

From the perspective of running time, we can see that the greedy algorithm takes much less time than the optimization model. For example, for $s = 1035, t - 1200, D = 26$, the greedy algorithm spends 0.84 seconds, while the optimization model spends 34.43 seconds. Similarly, for $s = 184, t = 187, D = 10$, the running time of the optimization model is almost 20 times that of the greedy algorithm. However, for a network with 5000 nodes, the optimization model takes only 1 minutes to obtain the solution, probably reasonable if we want the exact solution.

4.5.2.2 Weighted Network

We randomly assign weights from 1 to 5 to each edge and repeat the experiments for the terminal nodes above. We choose $\alpha = 1.5, 2.5, 3.5$ and compare the greedy algorithm and the optimization model, the results of which are listed in Table 4.5.

The results in Table 4.5 are consistent with the observations that we have obtained in previous experiments. However, the efficacy for $s = 709, t = 1314$ is much higher than the corresponding unweighted version. The reason is that for a pair of terminal nodes in the weighted version, there exist several paths with lengths that are close to each other. As a result, the network could apply more causalities without making the distance between s and t exceed the length constraint.

Table 4.5: Experiments for Weighted Power 5000

$\{s,t\}$	D_0	α	$\lvert\mathcal{MC}\rvert$	t_h(s)	OPT	t_o(s)
$\{184,187\}$	26.80	1.5	36	0.71	37	20.05
		2.5		0.77		20.44
		3.5		0.68		20.47
$\{10,14\}$	38.80	1.5	37	0.72	37	20.58
		2.5		0.74		20.22
		3.5		0.75		20.56
$\{709,1314\}$	76.30	1.5	35	0.83	36	20.64
		2.5		0.95		20.74
		3.5		0.93		20.66
$\{1035,1200\}$	110.59	1.5	29	0.88	35	30.61
		2.5				30.58
		3.5				30.29

4.5.2.3 Scalability Analysis

By comparing the results of the IEEE 300 network with the Power 5000 network, we observe the scalability of the proposed algorithms. Looking at the run time, it takes < 1 seconds for the optimization model for the IEEE 300 network, but the time is greater than 20 seconds for most experiments in the Power 5000 network. One might imagine that the running time would be much higher as the size of the network increases. However, the time taken by the greedy algorithm in the Power 5000 network is almost twice that of the time in the IEEE 300 network, while the greedy algorithm preserves the efficiency to a certain degree. If there is no other specific requirement for the number of causalities, the greedy algorithm is a good alternative with respect to the optimization model.

4.6 Conclusion

In this chapter, we consider the impact of causal failures on the shortest paths between pairs of nodes. We formulate a problem to find the maximum number of applied causalities since the length of the st-path is at most D. The formulated problem is NP-hard, and a mixed integer linear programming model is built correspondingly. In comparison, we propose a greedy algorithm to approximately obtain the number of applied causalities. The experiments demonstrate that the greedy algorithm works well in most cases and that the optimization model is acceptable in terms of running time.

For future work, we will look at the impact of causal node failures on st-reliability in a network. That is, considering the probability that each node is functioning or not, we will construct the model of probabilistic causal failures and investigate the probability that the source node s is connected to the terminal node t, that is, st-reliability, in the presence of causal failures.

Chapter 5

Causal Edge Falires on the Maximum Flow in a Network

5.1 Introduction

Network flow models have been applied to real-world applications, such as power transmission networks (44), supply chain networks (45), and transportation networks (46). The failure of edges affects the flow transmission in these networks and might trigger failures of other edges or a situation in which no flow is transmitted via other edges. More specifically, we consider the cases of the networks mentioned above as follows.

Power Networks

A power transmission network consists of generators, substations, and transmission towers, which are all modeled as nodes, and transmission lines, which are modeled as edges connecting those nodes. Each transmission line consists of several physical lines, and each physical line supplies a specific capacity. Therefore, each transmission line has multiple capacities, depending on the number of physical lines operating. For example, if there are x functioning physical lines for an edge, then we say that the capacity or state is x. Furthermore, if a transmission line fails, the load will be redistributed and other transmission lines might have to receive additional loads. It is possible that the extra load due to load redistribution exceeds the capacity of other transmission

lines, and thus causes complete or partial failure of these lines if the load of the failed transmission line has a relatively large load. This is an example in which the failure of an edge causes the failure of other non-adjacent edges.

Supply Chain Networks

A standard supply chain network consists of suppliers, plants, warehouses, and retailers (47), and it is represented by nodes found within levels labeled 1, 2, 3, and 4, respectively. The transportation edges are placed between levels l and $l+1$, where $l = 1, 2, 3$. Each transportation link has its own flow capacity, representing the maximum number of products that could be shipped between levels. The demand originates from retailers, and the products are shipped through the network to satisfy that demand. If an edge from a supplier fails due to a disruption (e.g., a natural disaster such as a flood or earthquake), the amount of each product delivered to the warehouses connected to the retailer will be reduced. As a result, the flow transmitted to the retailer could be smaller, or even 0. Let us consider that the flow of one edge is 0 to be an edge failure because of the initial edge failure. We could observe from this example that an edge failure in a supply chain network can trigger failures of other edges, regardless of whether these edges share common nodes.

Transportation Networks

A transportation network is a network in a geographical space, such as a highway or airline network. Such a network is generally modeled as a graph in which a node represents a location (e.g., a station, airport, or municipality), and an edge is a connection between two locations. For example, in a road network, an edge is a one-way road between two stations, and the weight of the edge indicates the number of vehicles that a particular road can handle in a specific time period. Vehicles can bypass

an edge representing a closed road segment by finding alternate roads but which can cause severe traffic congestion. To avoid such congestion, a traffic authority can shut down some low-capacity roads, forcing vehicles to choose high-capacity roads. From the perspective of failures, if we consider the closed road segment to be a failed edge, then causal edge failures are a common phenomenon in road networks.

The flow network model is important in representing engineering applications. For example, the assignment problem, a fundamental problem in operations research, is widely applied in real life. Generally, it is defined as assigning k' agents to k jobs to minimize the corresponding cost. Hu et al. (48) constructed an integer programming model based on network flow and proposed a dynamic algorithm to solve the generalized assignment problem. In operator networks, network function virtualization technology helps manage network service, the requests of which are usually deployed as a service function chain. However, load imbalance has been an issue in such a network. Han et al. (49) proposed a method for implementing the service function chain using network flow theory to solve the load imbalance. For coupled electricity and gas networks, Mhanna et al. (50) formulated the multi-period optimal electricity and gas flow problem, which aims to find a dispatch such that the costs of power generators and nodes in the gas network are minimized and solve the problem by constructing algorithms based on network flow and linear programming.

Node/edge failures affect the maximum flow and reliability between the source node s and the terminal node t in such flow networks. Specifically, the failure of one node triggers a malfunction of edges emanating from the node. As a result, flow cannot be delivered through these malfunctioning edges. Furthermore, the probability that d units of flow could be delivered from s to t is also lower because the failure of a node can reduce the number of minimal paths. Lin (51) considered a stochastic flow network where each edge/node has several capacities and may fail, and proposed an algorithm

to compute the probability that the maximum flow from s to t is at least a given integer d by finding all d-MPs and using the inclusion-exclusion principle. Similarly, Yeh (52) proposed an algorithm based on d-MC to compute multi-state two-terminal reliability based on the same network assumption. Both works show that node/edge failures play a vital role in the reliability of a flow network.

These works do not mention the dependence of edges in a network, regardless of whether the edges share common nodes. We address this deficiency in the literature by exploring causal edge failures and their impact on flow. And we formalize the study of such causal edge failures with the following contributions.

- We formally define causal edge failures in a directed flow network.
- We formulate a problem to find the maximum number of causal edge failures while preserving d units of flow to transmit from source node s to terminal node t.
- We prove the decision version of the formulated problem to be NP-complete and present a mixed linear programming model to find the exact solution to the problem.
- We design a greedy algorithm to approximate the solution to the optimization problem to deal with its computational complexity.

The rest of this paper is organized as follows. Section 5.2 builds the mathematical model of causal edge failures and formulates the problem. In Section 5.3, we show the hardness of the proposed problem and present a corresponding optimization model. Section 5.4 provides a genetic algorithm and an approximation algorithm to solve the problem. We demonstrate multiple experiments in Section 5.5, comparing the performance of the designed algorithms, and conclude this chapter in Section 5.6.

5.2 Problem Formulation

We consider a directed flow network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{C})$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ is the set of edges, $\mathcal{W}$ is the set of edge capacities and $\mathcal{C}$ is the set of causalities. The network has $|\mathcal{V}| = n$ nodes and $|\mathcal{E}| = m$ edges. Each edge $e \in \mathcal{E}$ can also be represented as an ordered pair of nodes $e = (u, v)$, where $u, v \in \mathcal{V}$. Furthermore, the capacity of edge $e = (u, v)$ is indicated by w_e or $w_{u,v}$, which is the maximum number of units of flow that can be transmitted through this edge.

Fig. 5.1 depicts an example of a flow network that has 6 nodes $\{1, 2, 3, 4, 5, 6\}$ and 9 edges $\{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9\}$, the capacities of which are $\{3, 2, 1, 2, 1, 3, 1, 1, 2\}$. For example, the capacity of edge e_1 is 3, such that edge e_1 can transmit at most 3 units of flow from node 1 to node 2. edges are correlated in this network. We consider

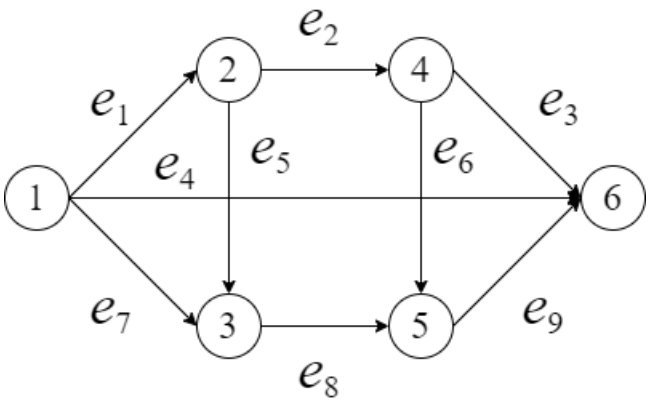


Figure 5.1: An example of a flow network.

the edge dependence model as follows:

Definition 5.1. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{C})$, edges $e_i, e_j \in \mathcal{E}$, if the failure edge e_i triggers the failure of the edge e_j, we say that causal edge failures denoted $C : e_i \Rightarrow e_j$, are applied to $\mathcal{G}$.*

Definition 5.2. *A causality $C : e_i \Rightarrow e_j$ is* applied *implies that when the edge e_i is removed, e_j is removed.*

The removal or failure of an edge is equivalent to the capacity of the edge being set to 0. Furthermore, we generalize causal edge failures to the case of edge sets as follows.

Definition 5.3. *Given a network $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{C})$, $\mathcal{E}_1, \mathcal{E}_2 \subseteq \mathcal{E}$, if the failure of edges in $\mathcal{E}_1$ cause the failure of* all *edges in $\mathcal{E}_2$, we call such failure as a* causal failure (edge) *from $\mathcal{E}_1$ to $\mathcal{E}_2$. We refer to this as* failure of $\mathcal{E}_1$ causes failure of $\mathcal{E}_2$. *The causality is denoted as $C : \mathcal{E}_1 \Rightarrow \mathcal{E}_2$.*

Example 5.1. *Given the causality $C_1 : \{e_1\} \Rightarrow \{e_2\}$ in Fig. 5.1, if we apply C_1 to the network, then we remove the edges e_1 and e_2 or set the capacities of both edges to 0 simultaneously. As a result, nodes 2 and 4 do not have incoming flow, and the edges e_3, e_5, and e_6 are unrelated to the flow transmission from node 1 to node 6.*

It should be noted that causal edge failures are an extension of the Shared Risk Links Group, a set of links that share a common resource, which affects all links in the set if the common resource fails (53). Causal edge failures can describe all kinds of edge dependence from the network topology perspective.

For the flow network, the flow from source node s to terminal node t is affected once we apply causal edge failures.

Maximum flow is perhaps the most popular way to characterize flow in such networks, defined as the maximum units that can be delivered from source node s to terminal node t. From the example in Fig. 5.1, the maximum flow from source node 1 to terminal node 6 is 5. Given the causalities, $C_1 : \{e_1\} \Rightarrow \{e_2\}$, $C_2 : \{e_6\} \Rightarrow \{e_7\}$, and $C_3 : \{e_5\} \Rightarrow \{e_3\}$, applying them, respectively, reduces the maximum flow to 2, 4, and 4. As such, it is also interesting to explore the impact of causal failures on maximum

flow, namely in finding the maximum subset of causalities that, when applied, enable the flow network to maintain some desired level of maximum flow.

Problem 5.1. *Given a flow network* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{C})$*, causalities* $C_i : \mathcal{E}_{1,i} \Rightarrow \mathcal{E}_{2,i}, i = 1, 2, \cdots, h$*, two nodes* $s, t \in \mathcal{V}$*, and a constant* d*, with a maximum flow* $f_{s,t}$*, find a* **maximum** *subset of causalities whose application maintains* $f_{s,t} \geq d$*.*

Problem 5.1 evaluates the robustness of the network when faced with edge failures. It uses the number of causalities of edges as a metric instead of the number of edges.

Intuitively, one could check each possible combination of all causalities and obtain the solution to Problem 5.1. However, the number of combinations to be checked is 2^h, where h is the number of causalities, indicating the hardness of Problem 5.1.

5.3 NP-Hardness Proof

We consider the decision version of the problem formulated, denoted as *VrCd*: Given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{C})$, source node s, and terminal node t, is there $\mathcal{C}' \subseteq \mathcal{C}$ with size at least r such that applying $\mathcal{C}'$ maintains $f_{s,t} \geq d$?

To prove *VrCd* is NP-complete, we refer to the decision version of the maximum 2-satisfiability (MAX-2SAT) problem: given a Conjunctive Normal Form with p clauses $c_1, c_2, \ldots, c_p$, is there a true/false assignment of all literals such that there are at least r clauses that are satisfied? MAX-2SAT is NP-complete (54). Then we have the following.

Theorem 5.1. *VrCd is NP-complete.*

Proof. First, it is straightforward to see that *VrCd* is in NP because we could verify a certificate of *VrCd* in polynomial time by using a known algorithm to find the maximum flow.

Then we perform a polynomial-time reduction from an instance I of MAX-2SAT to an instance I' of *VrCd* as follows:

- We identify source node s and terminal node t. Suppose that there are l literals $x_1, x_2, \ldots, x_l$ and/or their negatives $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_l$. For each literal x_i, we have nodes X_i, their negatives $\bar{X}_i$, and a dummy node D_i. In addition, we have edges (s, X_i), $(s, \bar{X}_i)$, (X_i, D_i), $(\bar{X}_i, D_i)$ and (D_i, t). The capacities of all the edges here are set to p.

- For each clause $c_j = x_a \vee x_b, j = 1, 2, \ldots, p$, we construct nodes x_a, x_b, c_j and edges (s, x_a), (s, x_b), (x_a, c_j), (x_b, c_j), (c_j, t). The capacities of all the edges here are set to p. In particular, if x_a appears in more than one clause, we do not construct duplicate nodes for x_a, that is, only one x_a is enough in the graph.

- If x_a is in clause $c_j, j = 1, 2, \ldots, p$, causality is defined as $C_a : \{(s, x_a)\} \Rightarrow \{(x_a, c_j), (s, X_a)\}$.

- Let $d = pl + p^2 - p + 1$ and E_a be applied if x_a is assigned True in clause c_j.

Suppose I is satisfied, then at least r clauses are satisfied. Then we choose one true literal from each satisfied clause, and consequently, at least r causalities are applied. The flow from s to t is $pl + p$, which meets the requirement that at least d units are transmitted from s to t.

Suppose I is not satisfied; then, at most, $r - 1$ clauses are satisfied. Further, we have the following.

Claim: It is not possible that causalities r are applied to the network.

Suppose there are r applied causalities. If each comes from one clause, then we have r clauses that are satisfied according to the above construction. Therefore, two of them come from the same clause c_j, leading to the situation that no flow from s to t will flow

through node c_j. As a result, the flow would be at most $p(l-1)+p^2 = pl+p^2-p < d$. Alternatively, two of the applied causalities come from x_i and $\bar{x}_i$, then the flow from s to t through node D_i is 0. Consequently, we have $f_{s,t} \leq pl+p(p-1) = pl+p^2-p < d$. Therefore, the claim holds and the theorem is proved. □

For another explanation of the proof of theorem 5.1, we provide an example in Fig. 5.2 where the instance of MAX-2SAT is $(x_1 \vee x_2) \wedge (\bar{x}_1 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_2)$. Consequently, six causalities are constructed in the graph, and, finally, two causalities are applied to the network according to the assignment of literals.

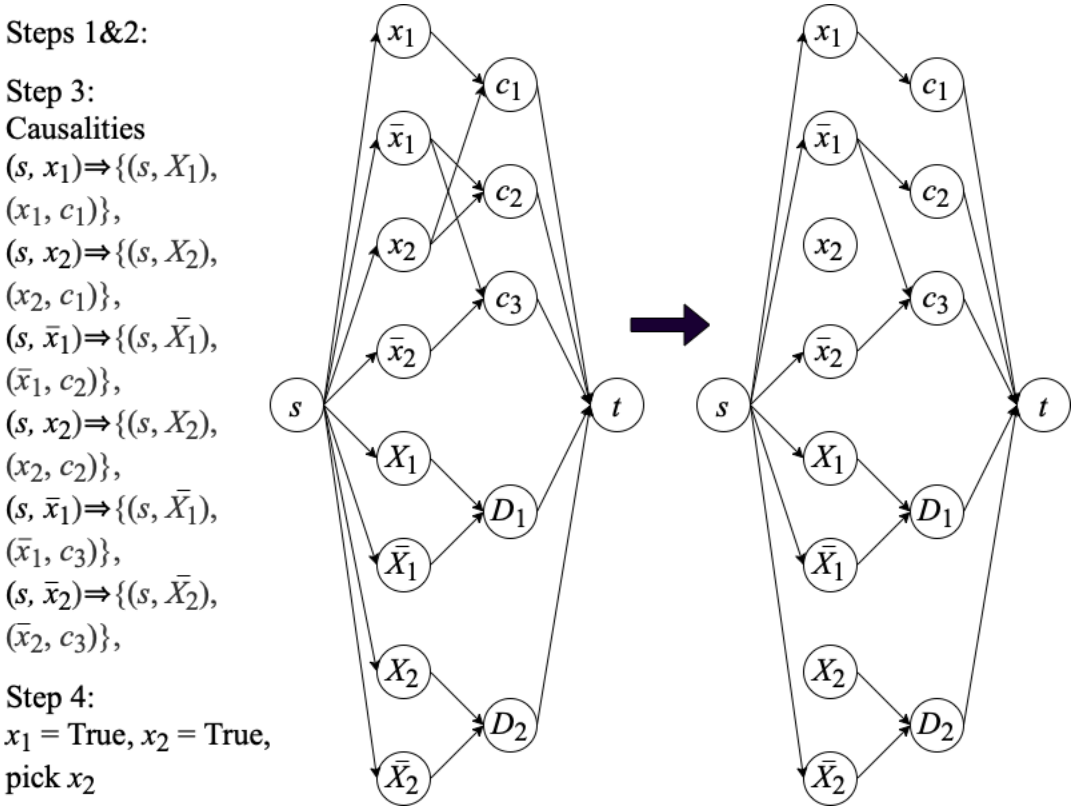


Figure 5.2: Explanation of proof for Theorem 5.1

.

5.4 Algorithms

5.4.1 Optimization model

We provide a mixed-integer linear programming (MILP) model to solve *VrCd*. In this model, an additional edge is added from node t to s, (t, s), with infinite capacity, and z_C is a binary variable that indicates whether causality C is applied: $z_C = 1$ indicates that causality C is applied, and $z_C = 0$ indicates that causality C is not applied.

$$
\begin{aligned}
maximize \quad & \sum_{C \in \mathcal{C}} z_C && (5.1)\\
s.t.\ f_{t,s} \quad & \geq \quad d && (5.2)\\
\sum_{u \in \mathcal{V}} f_{v,u} \quad & = \quad \sum_{u \in \mathcal{V}} f_{u,v}, v \in \mathcal{V} && (5.3)\\
f_{u,v} \quad & \leq \quad w_{u,v}, (u,v) \in \mathcal{E} && (5.4)\\
f_{u,v} \quad & \geq \quad 0, (u,v) \in \mathcal{E} && (5.5)\\
f_{u,v} \quad & \leq \quad w_{u,v}(1 - z_C), C \in \mathcal{C}, (u,v) \in \mathcal{E}(C) && (5.6)\\
z_C \quad & \in \quad \{0,1\}, C \in \mathcal{C} && (5.7)
\end{aligned}
$$

This formulation aims to find the maximum number of causalities with Eq. (5.1) while preserving the flow transmitted from s to t is at least d, governed by Eq. (5.2). We assume that there is no learning effect or deterioration effect in the network, so for each node, the sum of all coming flows equals the sum of all outgoing flows, as dictated by Eq. (5.3). This is called the flow conservation law. The flow passing through each edge does not exceed the capacity of the edge and must be nonnegative, ensured by Eqs. (5.4) and (5.5), respectively. For the edges of causality E, Eq. (5.6) requires that the flow must be 0 if causality E is applied, according to Eq. (5.7).

5.4.2 A Heuristic Algorithm

To find the solution more efficiently, we propose an approximation algorithm considering causalities. As we know, applying one causality might reduce the maximum flow from the source to the terminal. If the maximum flow drops sharply after removing edges from one causality, then this causality should not be applied to the network first. Therefore, for causality $C \in \mathcal{C}$, we consider the maximum flow in the graph $\mathcal{G} - C$, denoted as $\theta(C)$, and classify the causalities based on $\theta(C)$ in descending order. Moreover, we use a stack to store the applied causalities while checking whether the maximum flow is at least d. Details of the algorithm are listed below.

Algorithm 5 GREEDY-MAX-FLOW

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{C})$, flow demand d, source s, terminal t;
Output: approximated maximum number of applied causalities.

1: Sort all causalities based on θ values in ascending order, and let $\theta(C_1) \geq \theta(C_2) \geq \cdots \geq \theta(C_h)$;
2: $\mathcal{MC} = \emptyset$; *//initialization*
3: **for** i in $1 : h$ **do**
4: **if** the maximum flow of $\mathcal{G} - C_i$ is at least d **then**
5: $\mathcal{MC} \leftarrow \mathcal{MC} + C_i$; *//updating MC*
6: $\mathcal{G} \leftarrow \mathcal{G} - C_i$;
7: **else**
8: Continue;
9: **end if**
10: **end for**
11: **return** $\mathcal{MC}$.

Computational Complexity: Algorithm 1 only needs to compute the maximum flow for $2h$ times in the worst case. Hence, the total computational complexity is $O(hnm^2)$.

Approximation Ratio: After sorting, the first causality can be applied to the network but applying each of the remaining causalities would reduce the maximum

flow to less than d, while applying all other causalities is the solution. Therefore, this worst-case indicates that the tightest approximation ratio should be $h - 1$.

5.5 Experimental Results

In this section, multiple experiments are presented to compare the performance of the proposed algorithms and the optimization model. All algorithms, coded in Python, are implemented on a desktop with a 2.6 GHz 6-Core Intel Core i5 with a RAM of 16GB. Specifically, the optimization model utilizes the Gurobi library (21) and the NetworkX library (20) to help code the greedy algorithm and the genetic algorithm.

5.5.1 Comparison of Performance of Algorithms

In this subsection, we use a power network, denoted power2000, with 1,723 nodes and 2,394 edges from Rossi and Ahmed (26) with the following settings.

- We assign the capacity of each edge by randomly sampling an integer from 1 to w_{max}. In this experiment, $w_{max} = 20$.
- For each causality $C : \mathcal{E}_1 \Rightarrow \mathcal{E}_2$, we randomly generate m_1' and m_2' edges for $\mathcal{E}_1$ and $\mathcal{E}_2$, respectively, where $m_1', m_2' \in [1, m_{max}]$. In this experiment, $m_{max} = 30, h = 40$. In addition, to avoid cascading failures, we ensure that all causalities are disjoint.
- When running the optimization model, we convert this undirected network to a directed network by replacing each edge uv with two edges (u, v) and (v, u), the capacities of which are the same.

We choose different pairs of s, t, compute the number of causalities, and list the results in Table 5.1.

Table 5.1: Performance comparison of optimization model and greedy algorithm for power2000.

s, t	max-flow	d	MILP	GREEDY-MAX-FLOW			
			opt	$t_o(s)$	$\|\mathcal{MC}\|$	t_h(s)	$\|\mathcal{MC}\|/opt$
$\{40, 310\}$	5	1	36	6.10	36	4.75	1.00
		2	36	5.88	35	4.72	0.97
		3	35	6.09	35	5.17	1.00
		4	34	5.95	34	4.96	1.00
		5	33	6.09	33	5.10	1.00
$\{20, 1000\}$	11	1	36	5.81	36	6.80	1.00
		3	36	5.97	36	6.72	1.00
		5	36	5.85	36	6.64	1.00
		7	34	5.82	31	6.53	0.91
		9	32	5.91	30	7.12	0.94
$\{500, 1500\}$	18	1	38	5.81	37	8.70	0.97
		4	38	5.84	38	8.60	1.00
		7	37	6.04	35	8.79	0.94
		10	35	5.97	31	9.94	0.88
		16	29	6.17	28	9.68	0.96

From Table 5.1, we observe the efficacy of the greedy algorithm. The greedy algorithm could achieve almost at least 90% of the solution. For example, the optimization model and the greedy algorithm obtained 36 causalities for $s = 40, t = 310, d = 1$. Similarly, for $s = 20$, $t = 1000$, and $d = 9$, the number of causalities applied as obtained by the greedy algorithm is only 1 less than that obtained by the optimization model. However, in some cases, the greedy algorithm spent more time getting the results than the optimization model. For example, it took the optimization model 6.17 seconds to get the solution for $s = 500, t = 1500, d = 16$, while the greedy algorithm consumed 9.68 seconds. Similarly, for $s = 20, t = 1000, d = 9$, the greedy algorithm spent about 1 second more than the optimization model. This indicates that the greedy algorithm does not necessarily provide an advantage relative to the optimization model in terms of the running time for power2000.

From the perspective of the relationship between the number of edges and the number of causalities, we could observe the advantage of causalities from the experiments. Consider a network with m edges where we are looking for the maximum number of failed edges such that there exists flow d from s to t. This problem is very hard, requiring $O(2^m)$ time in the worst case because one needs to check each possible combination of all edges. However, based on causalities, we know that the expected number of edges in one causality is $m_{max} + 1$ and the expected number of causalities is $\frac{m}{m_{max}+1}$ if all edges belong to a specific causality. Consequently, we need to check the possible combination of causalities, which takes much less time than $O(2^m)$.

5.5.2 More Examples

To investigate the scalability of the optimization model and the greedy algorithm, we performed the experiments on a road network with 2642 nodes and 3303 edges and a power network with 4941 nodes and 6954 edges, denoted by road3000 and power5000,

respectively(26). We use the same experimental setting as power2000 except that $n_{max} = 40$ for road3000 and $n_{max} = 60$ for power5000.

Table 5.2: Performance comparison of algorithms for road3000.

s, t	max-flow	d	MILP		GREEDY-MAX-FLOW		
			opt	$t_o(s)$	$\|\mathcal{MC}\|$	t_h(s)	$\|\mathcal{MC}\|/opt$
$\{200, 2500\}$	5	1	29	99.53	26	9.24	0.90
		2	26	91.20	22	9.37	0.85
		3	25	65.12	20	10.61	0.80
		4	24	24.02	17	12.16	0.71
		5	22	27.34	17	11.91	0.77
$\{1000, 2000\}$	10	2	29	129.19	26	11.88	0.90
		4	27	146.57	20	13.32	0.74
		6	25	185.92	21	14.79	0.84
		8	22	130.07	18	15.55	0.82
		10	21	68.96	15	16.68	0.71
$\{1500, 1600\}$	12	2	37	13.55	37	7.71	1.00
		4	35	14.12	30	9.22	0.86
		8	35	13.43	35	8.54	1.00
		10	32	14.08	31	8.66	0.97
		12	28	13.85	28	8.39	1.00

Table 5.2 illustrates the efficacy of the greedy algorithm for road3000. The greedy algorithm achieves at least 70% of the solution obtained by the optimization model. Also, the greedy algorithm took much less time to get the solution. For example, the optimization model consumed 129.19 seconds to get 29 applied causalities for $s = 1000, t = 2000, d = 2$, while it only took 11.88 seconds for the greedy algorithm to get 26 applied causalities. The greedy algorithm only spent 10% of the time consumed by

the optimization model to obtain 90% of the solution. Similarly, the greedy algorithm only spent 12% of the time consumed by the optimization model to obtain 82% of the solution for $s = 1000, t = 2000, d = 8$.

Note that different choices of s, t could result in a significant difference in running time. For example, for $s = 1000, t = 2000$, the running time of the optimization model is almost 10 times that of the optimization model for $s = 1500, t = 1600$. The reason is that when searching for the MILP solution if the quality of the initial feasible solution is good, the search will stop quickly.

Table 5.3: Performance comparison of algorithms for power5000.

s, t	max-flow	d	MILP		GREEDY-MAX-FLOW		
			opt	$t_o(s)$	$\|\mathcal{MC}\|$	t_h(s)	$\|\mathcal{MC}\|/opt$
$\{1, 5\}$	11	5	37	46.49	37	18.27	1.00
		6	36	47.17	36	18.37	1.00
		8	36	47.36	35	18.73	0.97
		9	36	46.81	34	18.02	0.94
		10	36	46.74	34	18.05	0.94
$\{2100, 4600\}$	19	1	38	47.11	38	23.09	1.00
		5	36	46.66	36	23.29	1.00
		9	35	46.61	34	23.19	0.97
		13	33	47.10	33	22.52	1.00
		17	31	45.80	31	26.68	1.00
$\{1200, 2000\}$	28	6	37	47.79	37	17.14	1.00
		11	36	46.41	36	18.01	1.00
		16	35	46.85	34	17.13	0.97
		21	32	49.02	31	18.28	0.97
		26	30	48.13	29	18.62	0.97

Table 5.3 compares the performance of the optimization model with the greedy algorithm. The greedy algorithm again shows its efficacy and efficiency. For example, the greedy algorithm only spent 1/3 of the time cost by the optimization model to obtain the solution for $s = 2100, t = 4600, d = 6$. Similarly, the greedy algorithm achieved 97% of the solution using almost half the time consumed by optimization.

Looking at Tables 5.1, 5.2, and 5.3, we can see that the execution time of the algorithms depends on the size of the network and the choices of the terminal nodes s, t. We can see that for a pair of s, t in road3000, the run-time of the algorithms might be much more than that of an experiment in power5000. The case $s = 1000, t = 2000$ on road3000 and the case $s = 1200, t = 2000$ on power5000 illustrate that. In addition, both algorithms are scalable. The running time of the optimization model is not more than 4 minutes for all experiments. We are confident that both algorithms would be efficient in larger networks.

5.6 Conclusion

This chapter defines the causal edge failures and their impact on the flow network. We formulate an optimization problem to investigate the robustness of the flow network and design a greedy algorithm to search for the solution exactly and heuristically. The experiments show the efficiency and efficacy of the greedy algorithm for large-scale networks.

In future work, we will investigate probabilistic causal failures. More specifically, we plan to define the probability of applying one causality based on the likelihood that each edge is functioning. Then the reliability of a flow network facing causal failures will be defined. Moreover, we will also look at cascading causal edge failures in flow networks.

Chapter 6

Cost-Effective Network Augmentation Facing Causal Node Augmentation

6.1 Introduction

As a basic topic in graph theory, augmentation for network connectivity adds edges with minimum weights to meet some connectivity requirements (55). Based on connectivity metrics, network augmentation could be categorized into two cases: edge connectivity augmentation and node connectivity augmentation. For the former, connectivity augmentation is typically performed according to k-edge-connectivity, such that the network remains connected whenever at most k edges are removed. k-edge-connectivity measures how robust a network is, and the corresponding weighted augmentation problem is denoted as W-k-ECA. To determine whether a network is k-edge-connected, it is suggested to check whether the maximum flow between any pair of nodes is greater than k when setting the capacity of each edge to be 1 (as a maximum flow $\geq k$ means that there are at least k disjoint paths between a pair of nodes). Frederickson et al. (56) proved the NP-hardness of W-k-ECA even for $k = 2$. Watanabe et al. (57) designed approximation algorithms for W-k-ECA and pointed out that W-k-ECA has a variant: given a k-edge-connected graph, find the minimum weighted edges to be added such that the graph is $(k + 1)$-edge-connected. For this variant, Dinitz et al. (58) showed that if k is odd, then the problem could be converted to the tree augmentation problem;

otherwise, it is equivalent to the cactus augmentation problem. Lately, researchers have made significant progress in analyzing the performance of the approximation algorithm for this variant of W-k-ECA. Byrka et al. (59) presented a polynomial-time (< 1.91) approximation for the cactus augmentation problem. Furthermore, Cecchetto et al. (60) obtained a 1.393 approximation algorithm for the cactus augmentation problem. The augmentation problems on edge connectivity could be directly converted to the corresponding augmentation problems on node connectivity.

The above works focused on edge-weighted augmentation requiring that the network be connected after augmentation. There is no such work mentioning network augmentation facing causal failures. In addition, it is not necessary to always keep the network connected because the failures might cause severe disconnectivity to the network and the augmentation would be costly. Instead, we could maintain a connected component with a specific size.

Current research lacks the means to make connectivity augmentation decisions to maintain a connected component of a given size when faced with causal failures. Therefore, we consider this problem in this chapter: weighted edge connectivity augmentation under the threat of causal node failures. Our contributions are as follows:

- We propose a new metric called k-causality-α-robust to measure the robustness of a network.
- We formulate a problem looking for the minimum cost of adding edges such that the network has a giant component after the application of a given set of causalities.
- We prove that the formulated problem is NP-hard and present the corresponding MILP model.

- We propose an efficient greedy algorithm to solve the problem approximately and study its effectiveness in the experiments.

The remainder of this chapter is organized as follows. The system model and the problem are formulated in Section 6.2. Section 6.3 proves that the proposed problem is NP-hard. Section 6.4 proposes the corresponding mixed integer linear programming model to find the exact solution and a heuristic algorithm to solve the problem approximately. Section 6.5 verifies the performance of the proposed algorithms using several examples of random networks. Finally, Section 6.6 concludes this paper.

6.2 Problem Formulation

Fig. 6.1 shows the topology of a network $\mathcal{G}$ consisting of 11 nodes and 10 edges, along with their causal failures $C_1 : \{2\} \Rightarrow \{6\}, C_2 : \{4\} \Rightarrow \{10\}, C_3 : \{5\} \Rightarrow \{8\}$. If causality C_1 is applied to the network, then we have the removal of nodes 2 and 6 at the same time because the failure of node 2 causes the failure of node 6. As a result, we have 5 connected components in the remaining network (i.e. $\{1\}$, $\{3\}$, $\{4, 5, 9, 10\}$, $\{7\}$, $\{8, 11\}$). Similarly, if we apply causalities C_2 or C_3 to the network individually, then we could obtain 3 and 4 connected components, respectively. Therefore, applications of different causalities have different effects on the connectivity of the network.

Definition 6.1. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, causality-$\alpha$-robustness, denoted as $cr_\alpha(\mathcal{G})$, is the minimum number of causalities whose application leaves no α-giant component.*

Definition 6.1 proposes a new metric based on the existence of a α-giant component to measure the robustness of a network when several causalities are defined. In Fig. 6.1, given $\alpha = 0.4$, applying causality C_1 would result in no connected components with a size of at least 5. Therefore, $cr_{0.4}(\mathcal{G}) = 1$.

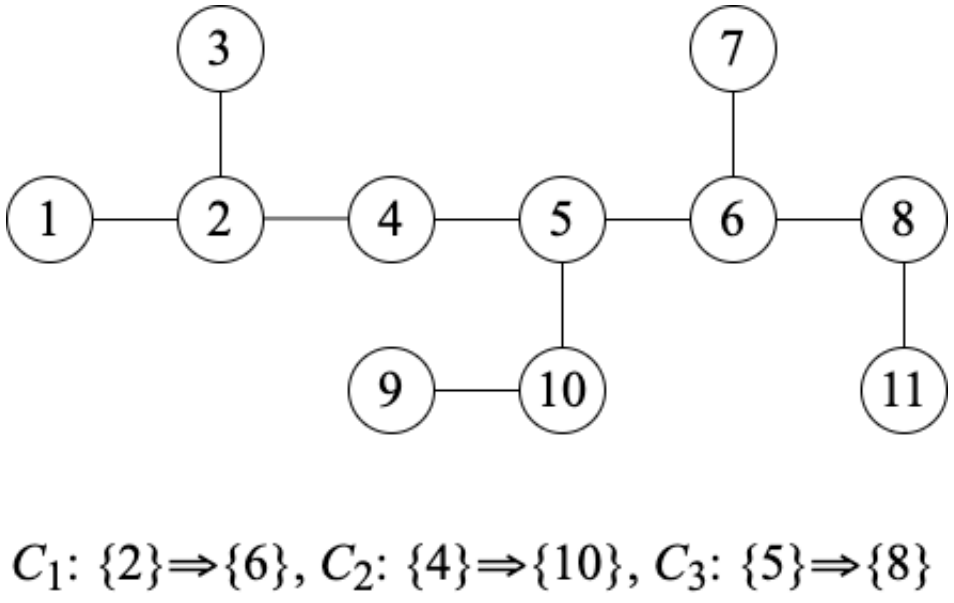


Figure 6.1: Network example with causalities.

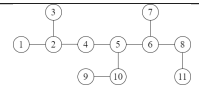
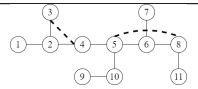
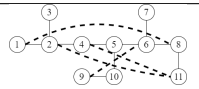
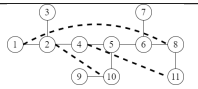
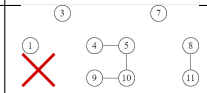
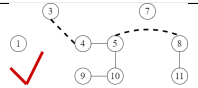
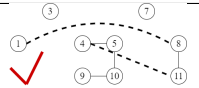
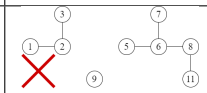
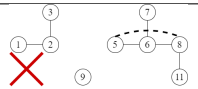
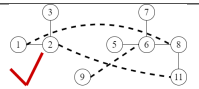
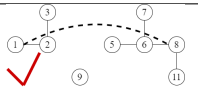
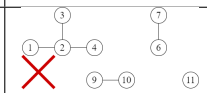
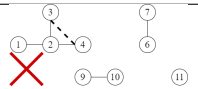
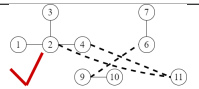
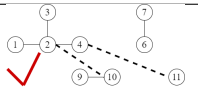
Definition 6.1 is actually looking for critical causalities of the network. Based on the number of critical causalities, we propose the following definition.

Definition 6.2. *A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$ is called k-causality-α-robust if $cr_\alpha(\mathcal{G}) \geq k$.*

Definitions 6.1 and 6.2 offer a means of measuring the robustness of a network based on the relationship between the number of causalities applied and the size of a giant component. In particular, with Definition 6.2, we can suggest that a network is robust if any $< k$ causalities could be applied to the network while preserving a α-giant component.

Generally, a network might not be as robust with respect to k and α when considering causal failures. As a result, it is necessary to add some edges to obtain the robustness of the network. In Table 6.1 we have illustrated three augmentation strategies for the network in Fig. 6.1 with three causal failures. There are two augmentations

Table 6.1: Augmentation on the network in Fig.6.1 given $\alpha = 0.6$. We assume, for illustration, that all the edge costs are one. The ones that have the 0.6-giant component after the application of the causal failure are given in the check marks. The augmented edges are in dashed format.

Augmentation Strategy	None	Add (3, 4), (5, 8)	Add (1, 8), (2, 11), (4, 11), (6, 9)	Add (1, 8), (2, 10), (4, 11)
Augmented Networks				
Apply $C_1 : \{2\} \Rightarrow \{6\}$				
Apply $C_2 : \{4\} \Rightarrow \{10\}$				
Apply $C_3 : \{5\} \Rightarrow \{8\}$				

in which all causal failures produce a 0.6-giant component and the one with the fewest edge augmentations adds three edges.

The small experiment above raises a significant problem: how to maintain a α-giant component regardless of which causality is applied to the network. To be more practical, we consider the fact that the costs of adding edges are not the same. Mathematically, we denote by $\mathcal{G}_c$ the complete graph of $\mathcal{G}$, that is, $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ where $\mathcal{E}_c = \{uv | u, v \in \mathcal{V}\}$. Consequently, $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{E}})$ is the complementary graph of $\mathcal{G}$ with respect to $\mathcal{G}_c$ and $\bar{\mathcal{E}} = \mathcal{E}_c - \mathcal{E}$. Furthermore, we define a cost function $c : \bar{\mathcal{E}} \to \mathbb{R}$ that indicates the cost to add an edge $e \in \bar{\mathcal{E}}$. The problem is formally formulated as follows.

Problem 6.1. *Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, together with a cost function $c : \bar{\mathcal{E}} \to R^+$ in $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{E}})$, and constants α and k, find a minimum-cost set of edges $\tilde{\mathcal{E}} \subseteq \bar{\mathcal{E}}$ such that $cr_\alpha((\mathcal{V}, \mathcal{E} \cup \tilde{\mathcal{E}})) \geq 2$.*

Different from the traditional node/edge connectivity augmentation problems, Problem 6.1 emphasizes that only nodes in causalities can fail, therefore nodes that are not in causalities are considered to be perfectly functioning. Given that the number of nodes in each causality is not fixed, the problem 6.1 is to find an augmentation strategy for the network to keep an α-giant component when random (causal) failures occur.

It is worth noting that causality is related to a group of nodes. For any fixed k, any combination of causalities of size k can be considered as a larger group of nodes if we combine the causalities in the combination as one causality. Therefore, we consider $k = 2$ in this work, which means that we are looking for a minimum-cost network where the application of any causality would maintain a connected component of size at least $\alpha \times n$.

6.3 NP-Hardness Proof

Before we move forward to the proof of the hardness of the formulated problem, we introduce some concepts in graph theory.

Definition 6.3. *The edge connectivity of a graph $\mathcal{G}$ denoted as $ec(\mathcal{G})$, is the minimum number of edges whose removal disconnects the graph.*

For example, we only need to remove edge (1, 2) to disconnect node 1 of the network in Fig. 6.1 from the remaining nodes, so the minimum number of edges to disconnect the network is 1. In other words, the edge connectivity of the network is 1, $ec(\mathcal{G}) = 1$.

Definition 6.4. *A connected graph $\mathcal{G}$ is called k-edge-connected if $ec(\mathcal{G}) \geq k$.*

Similarly to Definition 6.2, Definition 6.4 indicates that if removing fewer than k edges from the network $\mathcal{G}$ it remains connected, then $\mathcal{G}$ is k-edge-connected. Therefore, Definition 6.4 enables a metric for the robustness of a network in terms of the number of edges that could be removed. For example, the network in Fig. 6.1 is 1-edge-connected because removing any edge from the network would not disconnect it. It should be noted that the degree of node 1 in the network is 1, which is also the minimum degree, indicated by $\delta(\mathcal{G})$, of the network. From the above examples, we could observe that if a network $\mathcal{G}$ is k-edge-connected, then $k \leq \delta(\mathcal{G})$.

We have the following NP-complete weighted edge connectivity augmentation problem (57), denoted as W-2-ECA:

Problem 6.2. *Given a graph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$ together with a cost function $c : \bar{\mathcal{E}} \to R^+$ in $\bar{\mathcal{G}}' = (\mathcal{V}', \bar{\mathcal{E}}')$, find a minimum cost of the edges $\tilde{\mathcal{E}}' \subseteq \bar{\mathcal{E}}'$ such that $ec((\mathcal{V}', \mathcal{E}' \cup \tilde{\mathcal{E}}')) \geq 2$.*

Suppose that we have an instance I' of problem 6.2 and an instance I of problem 6.1, denoted by W-2-CRA, we prove the NP-hardness of problem 6.1 by constructing a polynomial-time reduction from I' to I.

Theorem 6.1. *W-2-CRA is NP-hard.*

Proof. Clearly, W-2-CRA is in NP because one could easily verify the causality-α-robustness given a certificate of I.

We perform the following steps to reduce I' to I:

- For each edge $uv \in \mathcal{E}'$, we add two nodes u^*, v^*, remove the edge uv and add the edges uu^*, u^*v, uv^*, v^*v. Then we set the causality $C : u^* \Rightarrow v^*$. Therefore, we have the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ of instance I where $|\mathcal{V}| = |\mathcal{V}'| + 2|\mathcal{E}'|$ and $|\mathcal{E}| = 4|\mathcal{E}'|$, and the causality set $\mathcal{C}$, which contains $|\mathcal{E}'|$ causalities.

- We denote by $c'_{u',v'}$ and $c_{u,v}$, respectively, for $u'v' \in \bar{\mathcal{E}}'$ and $uv \in \bar{\mathcal{E}}$. If $u, v \in \mathcal{V}'$, then $c_{u,v} = c'_{u,v}$; otherwise, $c_{u,v} = \infty$. Furthermore, if (u, v) is in $\mathcal{G}'$, then $c_{u,v} = \infty$.

- $\alpha = 1 - \frac{2}{|\mathcal{V}|}$.

If I' is satisfied, then $ec((\mathcal{V}', \mathcal{E}' \cup \tilde{\mathcal{E}}')) \geq 2$ after adding edges in $\tilde{\mathcal{E}}'$ with minimum cost, that is, after augmentation, the removal of any edge would preserve the connectivity of $\mathcal{G}'$. Using the same augmentation strategy, we could always have one connected component of a size of at least $|\mathcal{V}| - 2$ whenever any causality is applied. Therefore, I is also satisfied. On the reverse side, if I is satisfied, then the connected component with a size of at least $|\mathcal{V}| - 2$ indicates the connectivity of the network $\mathcal{G}'$ because each causality containing two nodes corresponds to an edge in I', so I' is also satisfied.

If I' is not satisfied, then after augmentation, there exists an edge such that network $\mathcal{G}'$ is not connected without this edge. By adding the same edges as $\mathcal{G}'$ to $\mathcal{G}$, we apply the corresponding causalities to network $\mathcal{G}$, because $\mathcal{G}'$ is not connected whichever edge we remove and then the size of the largest connected component of $\mathcal{G}'$ is less than $|\mathcal{V}'| - 1$. Correspondingly, the size of each connected component is obviously less than

$|\mathcal{V}|-2$. Therefore, I is not satisfied. On the reverse side, if I is not satisfied, then there exists a causality whose application results in at least two connected components. The network $\mathcal{G}'$ is not connected if the corresponding edges are removed. Therefore, I' is also not satisfied. □

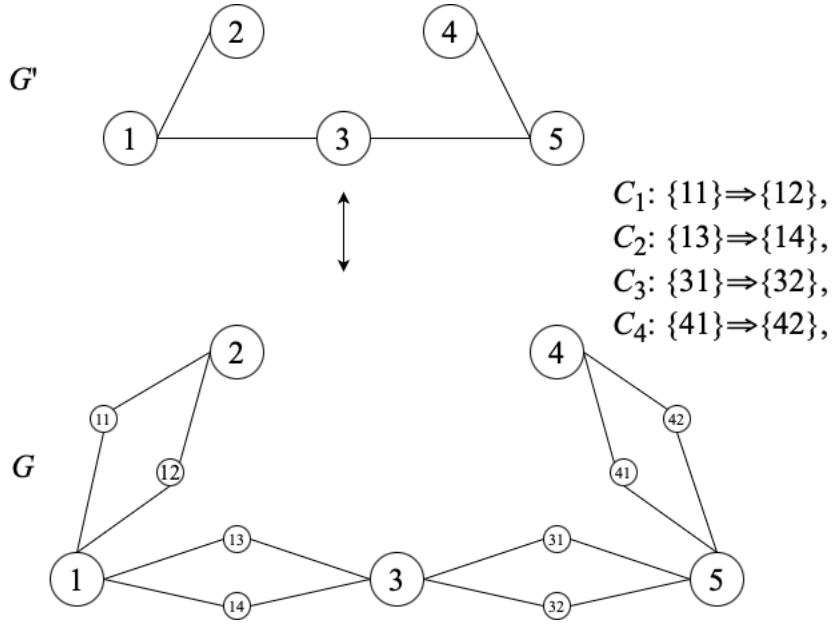


Figure 6.2: An example of the proof of Theorem 1

For another explanation, we use the example in Fig. 6.2 to illustrate the reduction. Given a graph $\mathcal{G}'$ with 5 nodes and 4 edges, it is obvious that removing any edge from the graph would disconnect the graph. As Fig. 6.2 shows, we replace each edge of $\mathcal{G}'$ with 2 nodes and 4 edges and construct 4 causalities. Regarding the cost of edges, we have $c_{u,v} = \infty$ if $(u, v) \in \{(1, 2), (1, 3), (3, 5), (4, 5)\}$ or if one of u, v is in $\{11, 12, 13, 14, 31, 32, 41, 42\}$. For the rest, $c_{u,v} = c'_{u,v}$.

6.4 Algorithms

6.4.1 MILP for the Problem

We introduce the idea of building a flow network to find an α-giant component from (61). For a given graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$, Zhang et al. (61) converted $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$ to its directed version $\mathcal{G} = (\mathcal{V}, \mathcal{A}, \mathcal{C})$ by replacing each edge $uv \in \mathcal{E}$ with directed arcs $(u, v), (v, u)$. They added auxiliary nodes s, t and arcs $(s, u), (u, t)$ to $\mathcal{G}$ for each node $u \in V$. Each edge connected to node s has an index that indicates whether the flow from s passes through this edge. The model requires that only one node in $\mathcal{G}$ receives the flow from s. As a result, the node that receives the flow from s would be the node in the largest connected component. By properly setting the capacities of all the arcs, the remaining network has a α-giant component if and only if the flow from s to t is at least $\alpha \times n$.

For our problem, we need to ensure that the augmentation strategy, which is the set of edges to be added, is applicable to all causalities. Hence, we create h copies, each of which is for each causality, respectively. All copies share one augmentation strategy, that is, adding edges from $\bar{\mathcal{G}}$ to $\mathcal{G}$. We denote the directed versions of $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{E}})$ and $\mathcal{G}_c = (\mathcal{V}, \mathcal{E}_c)$ by $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{A}})$ and $\mathcal{G}_c = (\mathcal{V}, \mathcal{A}_c)$, respectively, and present our optimization model with the corresponding explanation as follows:

First, the objective function Eq. (6.1) minimizes the cost of the augmentation strategy. Constraints (6.2) and (6.3) determine whether the edge uv is added. That is, if edge uv is added, $x_{uv} = x_{vu} = 1$; otherwise, $x_{uv} = x_{vu} = 0$.

Second, for copy $i \in H = \{1, 2, \cdots .h\}$, an extra edge (t, s) is added so that we just need to guarantee the flow from t to s is at least $\alpha \times n$ (Eq. (6.4)) while ensuring flow conservation (Eq. (6.5)). Constraints (6.6) and (6.7) guarantee that only one arc from the source node s has positive flow with capacity at most n. Constraints (6.8)-(6.11)

require that all flow values going to and coming from the nodes in C_i are 0, indicating that the nodes in causality C_i are removed. Each edge (u, v) in $\bar{\mathcal{G}}$ also has its flow value $f_{i,uv}$ in copy i, and its capacity is determined by whether this edge is added (Eqs. (6.12-6.13)).

$$\min_{uv \in \bar{\mathcal{E}}} \quad c_{uv} x_{uv} \tag{6.1}$$

$$s.t. \quad x_{uv}, x_{vu} \in \{0,1\}, \quad uv \in \bar{\mathcal{E}} \tag{6.2}$$

$$x_{uv} = x_{vu}, \quad uv \in \bar{\mathcal{E}} \tag{6.3}$$

$$f_{i,ts} \geq \alpha n, \quad i \in H \tag{6.4}$$

$$\sum_{u:(u,v)\in\mathcal{A}_c} f_{i,uv} = \sum_{u:(v,u)\in\mathcal{A}_c} f_{i,vu}, \quad i \in H \tag{6.5}$$

$$\sum_{u \in \mathcal{V}} y_{i,u} = 1, \quad i \in H \tag{6.6}$$

$$f_{i,su} \leq y_{i,u} n, \quad u \in \mathcal{V}, i \in H \tag{6.7}$$

$$f_{i,uv} = 0, \quad u \in \mathcal{V}(C_i), (u,v) \in \mathcal{A}, i \in H \tag{6.8}$$

$$f_{i,vu} = 0, \quad u \in \mathcal{V}(C_i), (v,u) \in \mathcal{A}, i \in H \tag{6.9}$$

$$f_{i,su} = 0, \quad u \in \mathcal{V}(C_i), i \in H \tag{6.10}$$

$$f_{i,ut} = 0, \quad u \in \mathcal{V}(C_i), i \in H \tag{6.11}$$

$$f_{i,uv} \leq n x_{uv}, \quad (u,v) \in \bar{\mathcal{A}}, i \in H \tag{6.12}$$

$$f_{i,vu} \leq n x_{vu}, \quad (v,u) \in \bar{\mathcal{A}}, i \in H \tag{6.13}$$

$$y_{i,u}, f_{i,ut} \in \{0,1\}, \quad u \in \mathcal{V}, i \in H \tag{6.14}$$

$$f_{i,su} \in \mathbb{N}, \quad u \in \mathcal{V}, i \in H \tag{6.15}$$

$$f_{i,uv}, f_{i,vu} \in \mathbb{N}, \quad uv \in \mathcal{E} \cup \bar{\mathcal{E}}, i \in H \tag{6.16}$$

Third, this formulation creates $\binom{n}{2} - n$ binary variables (Eq. (6.3)) to indicate whether an edge is added, and for each copy, $\binom{n}{2} + n$ integer variables (Eqs. (6.15)-(6.16)) and binary variables $2n$ (Eq. (6.14)) are created to ensure that the flow from s to t is at least $\alpha \times n$. Therefore, the time complexity of this formulation is $O(2^{n^2-n+hn} n^{n^2+n})$.

6.4.2 A Heuristic Algorithm

Given that it takes exponential time to run the optimization model, it is necessary to design a polynomial-time heuristic algorithm to approximate the solution for practical purposes. To make the designed algorithm work for each causality, we naively check the application of each causality, respectively, to the network and augment the connectivity of the network based on the remaining network, where there are clearly several connected components. We start from the largest connected component and look for possible connections to other connected components. A straightforward issue is how to choose from the rest connected components. As mentioned above, every edge in $\bar{E}$ has a cost associated with its addition. Hence, we define the cost to connect two connected components based on that:

Definition 6.5. *The cost between two connected components CC_i and CC_j $(i \neq j)$ is the smallest $c(uv)$ where $u \in CC_i$ and $v \in CC_j$.*

Definition 6.5 defines the cost of adding an edge between two connected components. By using Definition 6.5, the remaining network could be converted to a complete network with weighted nodes and edges. More specifically, suppose that there are p connected components in the remaining network after applying one causality, then we have the following rules:

R1 Each connected component is converted to a supernode. The weight of the super-node is the number of nodes in the connected component.

R2 The weight of each edge between any pair of super-nodes is the cost of adding that edge by Definition 6.5.

Therefore, the augmentation could be performed on the completed network constructed with p nodes and $p(p-1)/2$ edges.

The next step is to choose the edges. Intuitively, we start from the super-node with the largest weight and add its minimum-weighted edge, resulting in one giant component containing two super-nodes. Then we update the cost between other super-nodes and this giant component and add the minimum-weighted edge and so on. It is worth noting that added edges should compose a tree of super-nodes because an edge resulting in a cycle increases the cost but not the number of nodes. Therefore, when searching for and adding edges, we need to ignore those that would generate a cycle. A solution is to remove this edge and look for the next edge at a minimum cost. The algorithm, named as GREEDY-MIN-CRA, terminates when we obtain an α-giant component for the original network. The details are listed below.

Complexity analysis: In line 2, it takes $O(n^2)$ to find the size of the largest connected component when applying one causality and $O(h \log h)$ to sort the causalities. For lines 3-24, when considering each causality, it takes $O(n^2)$ to check all possible edges to be added. Therefore, in the worst case, the time complexity of the algorithm is $O(hn^2)$.

Algorithm 6 GREEDY-MIN-CRA

Input: graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{C})$ and its complement graph $\bar{\mathcal{G}} = (\mathcal{V}, \bar{\mathcal{E}})$ with edge costs;
Output: minimum cost of adding edges from $\bar{\mathcal{G}}$ to $\mathcal{G}$, s.t. when any causality is applied to $\mathcal{G}$, there exists at least one α-giant component.

$cost \leftarrow 0$; //the initial cost
Sort causalities based on the size of the largest connected component when applying each of them individually to the network
for C in (sorted) $\mathcal{C}$ **do**
    $\mathcal{H} \leftarrow \mathcal{G}$; // Copy $\mathcal{G}$
    $\mathcal{H} \leftarrow \mathcal{H} - C$; // Apply C to $\mathcal{H}$
    $S \leftarrow$ the size of the largest connected component;
    if $S > \alpha \times |\mathcal{V}|$ **then**
        continue;
    else
        Convert $\mathcal{H}$ to the corresponding complete graph $\mathcal{H}_c$ by using **R1** and **R2**;
        $v_c \leftarrow$ the super-node with size S;
        while $S < \alpha \times |\mathcal{V}|$ **do**
            $e \leftarrow \arg\min_{e \in \mathcal{E}(v_c)} c(e)$;
            if $v_c + e$ has a cycle in $\mathcal{H}_c$ **then**
                continue;
            else
                $v_c \leftarrow v_c + e$;
                $cost \leftarrow cost + c(e)$;
                $\mathcal{H} \leftarrow \mathcal{H} + e$;
                $\mathcal{G} \leftarrow \mathcal{G} + e$;
            end if
        end while
    end if
end for
return $cost$;

6.5 Experimental Results

In this section, we conduct experiments on synthetic networks to test the performance of the MILP model and the approximation algorithm. The Erdős–Rényi model (62) is used to generate the networks and all algorithms are coded in Python. The Gurobi (21) and NetworkX (20) libraries are used specifically for the MILP and greedy algorithms. All programs were run on a laptop with 16GB RAM and an Apple M1 Pro chip.

6.5.1 Comparison of MILP and GREEDY-MIN-CRA

We randomly generate an Erdős–Rényi network with 150 nodes. The probability that each pair of nodes has an edge is set to 0.02. The generated network has 223 edges. We also generate 20 causalities. For each causality $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$, the number of nodes in $\mathcal{V}_1$ or $\mathcal{V}_2$ is sampled from a discrete uniform distribution between 0 and 5. After determining the number of nodes in each causality, we randomly assign nodes to each causality from all nodes in the network. To be specific, the numbers of nodes in the causalities are 4, 8, 9, 5, 8, 3, 5, 6, 8, 6, 5, 5, 8, 6, 6, 8, 5, 10, 5, and 4, and the numbers of nodes in the largest connected component of the remaining networks after applying each causality individually are 137, 131, 128, 137, 134, 136, 136, 133, 131, 134, 132, 136, 133, 133, 136, 132, 136, 128, 136, and 137, respectively. Therefore, the range of α is between $0.86 (= 128/150)$ and $0.93 (= (150 - 10)/150)$. It is worth noting that if $\alpha > 0.93$, whatever edges we add to the network, the network would not always have a connected component with at least 140 nodes because one causality removes 10 nodes from the network. Moreover, the cost of each edge in the complement graph of the generated network is sampled from a continuous uniform distribution between 0 and 100.

Table 6.2: Comparison of MILP and GREEDY-MIN-CRA

α	MILP			GREEDY-MIN-CRA			Comparison	
	No. of added edges	Cost of added edges	t_o (s)	No. of added edges	Cost of added edges	t_h (s)	No. of edges found by both	Approximation ratio
0.86	1	0.001	11.078	1	0.001	0.342	1	1
0.87	3	0.085	11.497	4	0.140	0.344	3	1.642
0.88	5	0.432	11.368	5	0.432	0.345	5	1
0.89	4	1.012	11.705	8	1.455	0.344	3	1.438
0.90	6	1.358	11.548	9	2.118	0.345	5	1.560
0.91	8	2.434	12.162	10	2.776	0.347	7	1.141
0.92	10	3.396	13.069	12	4.144	0.368	9	1.220
0.93	12	6.659	11.747	15	9.512	0.345	11	1.428

Table 6.3: Approximation Ratios of Multiple Experiments on Networks with 150 Nodes

α	1	2	3	4	5	6	7	8	9	10	Avg.
0.89	1	1.015	1	1.101	1.236	1	1	1.330	1.830	1.039	1.155
0.90	1	1.001	1	1.087	1.286	1	1.364	1.252	1.579	1.034	1.160
0.91	1	1	1.056	1	1.256	1	1.008	1.240	1.606	1.283	1.145
0.92	1.028	1.097	1.043	1	1.162	1	1	1.121	1.479	1.203	1.113
0.93	1	1.040	1.027	1	1.288	1.003	1	1	1.349	1	1.071

Table 6.2 compares the performance of the heuristic algorithm with the optimization model.

First, the time consumed by the heuristic algorithm is much less than that of the optimization model. For example, for $\alpha = 0.86$, the heuristic algorithm only spends 0.342 seconds looking for the edge to be added, while it takes over 11 seconds for the optimization model to find the added edge. Similarly, for $\alpha = 0.88$, the heuristic algorithm succeeds in finding the same solution as the optimization model, but the time it spends is only 3% of the time the optimization model consumes.

Second, it is not always true that the larger the α-giant component specified, the more edges the networks need to add. The results for $\alpha = 0.88, 0.89$ are in agreement with this claim. For $\alpha = 0.88$, the optimization model finds five edges to be added while it finds only four edges to be added for $\alpha = 0.89$. However, it is generally true that a larger connected component leads to more added edges using the heuristic algorithm. We could see this trend from the *No. of added edges* column for the GREEDY-MIN-CRA algorithm.

Third, the edges found by the heuristic algorithm are almost the same as those obtained from the optimization model. As illustrated above, both algorithms return the same results for $\alpha = 0.86, 0.88$. For $\alpha = 0.91$, the heuristic algorithm finds seven edges that are also found by the optimization model. The difference is that the heuristic returns three extra edges, while the optimization model gives another edge.

Fourth, the heuristic ratios with respect to the cost of the edges to be added are less than two. From the approximation ratio (cost) in the column, the highest is 1.642 for $\alpha = 0.87$. Furthermore, we perform the same experiment 10 more times and show the corresponding approximation ratios with respect to the values of α in Table 6.3. From Table 6.3, we observe that a higher α generally leads to a lower approximation ratio on

average. For α ranging from 0.89 to 0.93, the approximation ratio almost drops from 1.155 to 1.072 on average.

Combining Tables 6.2 and 6.3, we could see the efficiency of the heuristic algorithm in terms of running time and the efficacy of the heuristic algorithm in terms of the approximation ratio.

6.5.2 Scalability Analysis

In this subsection, we generate Erdős–Rényi networks with 300, 500, and 1000 nodes to test the scalability of the proposed algorithms. The cost of each edge in the complement graph of each network is sampled from a continuous uniform distribution ranging from 0 to 100. Other network settings are listed as follows:

- For a network with 300 nodes, the probability that there exists an edge between a pair of nodes is 0.01 and the network has 452 edges. For each causality $C : V_1 \Rightarrow V_2$, $|V_1|$ or $|V_2|$ is sampled from a discrete uniform distribution from 1 to 10. We generate 20 causalities and the number of nodes in the causalities is 11, 17, 15, 17, 11, 10, 11, 9, 18, 7, 20, 15, 11, 11, 16, 15, 14, 15, 6, and 4. Correspondingly, the number of nodes in the largest connected component after applying each causality individually is 262, 260, 261, 256, 257, 267, 266, 267, 255, 269, 252, 258, 263, 261, 261, 257, 264, 259, 271, and 272, respectively. Hence, the value of α is between 0.86 and 0.93. We choose $\alpha = 0.86, 0.88, 0.90, 0.92$ in this experiment.

- For a network with 500 nodes, the probability that two nodes are connected is 0.0075. The generated network has 953 edges. For each causality $C : V_1 \Rightarrow V_2$, $|V_1|$ or $|V_2|$ is sampled from a discrete uniform distribution from 1 to 20. We also generate 20 causalities with 21, 16, 16, 18, 9, 16, 13, 16, 17, 13, 5, 21, 23, 19,

Table 6.4: Scalability Analysis: three networks with 300, 500, 1000 nodes respectively

Network size	α	MILP			GREEDY-MIN-CRA			Comparison	
		No. of added edges	Cost of added edges	t_o (s)	No. of added edges	Cost of added edges	t_h (s)	No. of edges found by both	Approximation ratio
300	0.86	2	0.183	54.031	6	0.361	1.391	1	1.973
	0.88	8	0.556	58.031	9	0.659	1.397	6	1.185
	0.90	13	1.271	57.737	14	1.496	1.402	11	1.177
	0.92	19	3.374	62.761	20	3.444	1.408	17	1.021
500	0.91	1	0.019	120.794	2	0.028	3.877	1	1.474
	0.92	5	0.158	124.342	5	0.158	3.888	5	1
	0.93	10	0.708	123.870	10	0.708	3.883	10	1
1000	0.91	10	0.120	707.223	10	0.126	17.505	8	1.050
	0.92	19	0.395	943.203	20	0.437	17.662	18	1.106
	0.93	28	0.967	791.400	29	0.972	17.688	27	1.005
	0.94	39	1.870	801.893	41	2.132	19.729	39	1.140

14, 27, 15, 26, 30, and 21 nodes. After applying them individually, we have 468, 472, 473, 470, 481, 473, 477, 477, 474, 476, 485, 468, 467, 472, 477, 461, 474, 460, 453, and 468 nodes in the largest connected component, respectively. Therefore, we choose $\alpha = 0.91, 0.92, 0.93$.

- For the network with 1000 nodes, we set the probability to 0.0035, and the random network has 1688 edges. Similarly, 20 causalities are generated since $|V_1|$ or $|V_2|$ is sampled from a discrete uniform distribution from 1 to 30. The number of nodes in the causalities is 23, 6, 26, 24, 32, 30, 19, 21, 14, 40, 32, 32, 23, 45, 29, 28, 32, 25, 18, and 29. The number of nodes in the largest connected component is 918, 943, 917, 925, 913, 920, 934, 930, 936, 903, 916, 916, 928, 900, 915, 918, 914, 924, 929, and 917, respectively, if each causality is applied to the network individually. Therefore, the values of α are set to 0.91, 0.92, 0.93, and 0.94.

Table 6.4 shows the results of the above three networks. From the perspective of running time, it can be seen that with increasing network size, the time consumed by the optimization model grows exponentially. For example, for $\alpha = 0.92$, the optimization model spends 62.761, 124.342, and 943.203 seconds looking for the solution for the three network sizes, respectively. Taking into account the time it spends on the 150-node network for the same α in Table 6.2, which is 13.069 seconds, we could easily observe the trend. On the other hand, it takes much less time for the heuristic algorithm to obtain the solution. For $\alpha = 0.92$, the time spent by the heuristic algorithm is only $2.8\%, 2.2\%, 3.1\%$, and 1.9% of the time consumed by the optimization model for the four networks.

Meanwhile, the heuristic algorithm maintains the approximation at less than two for the experiments in Table 6.4. The highest is 1.973 for $\alpha = 0.86$ and the network

with 300 nodes, because the heuristic algorithm returns six edges to be added, while the optimization model shows only two edges.

Moreover, from the perspective of the number of edges to be added, we could see that the closer the numbers of edges found by both algorithms are, the lower the approximation ratio will be. For example, for $\alpha = 0.91$ in a network with 1000 nodes, both algorithms find 10 edges, but only 8 edges are found by both. Similarly, we have an approximate ratio of 1.050. For $\alpha = 0.92$ in the network with 300 nodes, the heuristic algorithm finds 20 edges, while the optimization returns 19 edges. In this case, the approximation ratio of 1.021 is also as low as expected.

6.6 Conclusion

In this chapter, we consider the augmentation of weighted-edge network connectivity in the face of causal failures. Given a graph and predefined causalities, the proposed problem aims to find a set of edges added with the minimum cost such that the augmented network maintains at least an α-giant component if each causality is applied to the network individually. We provide the NP-hardness proof of the proposed problem and construct a mixed-integer linear programming model based on the flow network to find the solution to the problem. Given that the optimization model takes exponential time, we design a heuristic algorithm to approximate the solution. By performing multiple experiments on synthetic networks, we observe that the heuristic algorithm achieves an approximation ratio of less than 2 and takes significantly less time compared to the optimization model.

For future work, we would like to consider the network augmentation strategy where the network could maintain a α-giant component if any $< k$ causalities are applied to the network. In addition, we would like to introduce probability to causal failures (i.e.,

the failure of one node causes the failure of another node with some probability) and construct a probabilistic model for causal failures and network augmentation.

Chapter 7

Conclusions

7.1 Summary

This dissertation proposes a new failure model in one network. We formally define causal failures and investigate the impact of causal node failures on the robustness by measuring the maximum number of applied causalities when maintaining an α-giant component (Chapter 2), vulnerability by measuring the minimum number of causalities to obtain at least k connected components (Chapter 3), and shortest path by measuring the maximum number of applied causalities to keep the distance between node s and node t at most a given number(Chapter 4), the impact of causal edge failures on the maximum flow by measuring the maximum number of applied causalities to maintain the flow from s to t at least d(Chapter 5), and network augmentation when facing causal node failures (Chapter 6).

In each chapter, we propose a discrete optimization problem, prove the NP-hardness of the problem, construct the corresponding MILP model, and design a heuristic algorithm. The experiments show the efficiency and efficacy of the heuristic algorithm when compared with the MILP model in each chapter.

7.2 Future Work

Causal failures can be extended to more versions and problems. In future work, we consider more problems related to deterministic and probabilistic causal failures.

For deterministic causal failures, we will investigate the following problems:

- An α-giant component indicates a significant proportion of nodes that are included in one connected component. The existence of one α-giant component guarantees a significant proportion of network performance, especially when there are cascading failures in the network. Further, node failures may increase the distance between any pair of nodes. For example, node failures might increase the power loss in the power network and affect the quality of communication in a communication network, etc. It is not enough to guarantee the existence of an α-giant component. Moreover, we hope the diameter of the giant component is not too large so that all nodes in the giant component could be "close" to each other.

- Service degradation means impacted customers' use of the service. It will occur when the service is not meeting the criteria for Measured Throughput and Backup Service Availability. We assume that each node $v \in \mathcal{V}$ has service level 100 initially. The degradation of the service of a node is triggered by that of another node. For instance, a causality $C : u \Rightarrow v$ with a service degradation mapping $f : [0, 100] \Rightarrow [0, 100]$ (e.g. $f(50) = 20$ means that 50 service level of node u can only maintain 20 of node v). Generally, if the causality is defined on two node sets, i.e. $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$, the service degradation mapping is denoted as $f : [0, 100]^{|\mathcal{V}_1|} \Rightarrow [0, 100]^{|\mathcal{V}_2|}$. Hence, we would have causal partial failures in networks and consider new versions of giant components and connected component problems.

- Failures of $\mathcal{V}_{i,1}$ in causality $C_i : \mathcal{V}_{i,1} \Rightarrow \mathcal{V}_{i,2}$ do not trigger failures of nodes in $\mathcal{V}_{i,2}$ at once. It takes some time for nodes in $\mathcal{V}_{i,2}$ to fail completely. As a result, we will consider causal failures with time effects.

For random cases, we will consider causal failures by introducing functioning probability to each node. More specifically, we suppose that $p = 1 - q(0 < p < 1)$ is the probability that each node $v \in \mathcal{V}$ is functioning. For short, we denote by s_v the state of vertex v and $s(v) = 1(0)$ means vertex v is functioning (failed). As a result, we have $Pr(s(v) = 1) = p, Pr(s(v) = 0) = 1 - p = q$. For each causality, say $C : \mathcal{V}_1 \Rightarrow \mathcal{V}_2$, where $\mathcal{V}_1 = \{v_{11}, v_{12}, \cdots, v_{1n_1}\}$, $n_1 = |\mathcal{V}_1|$ is the number of nodes in $\mathcal{V}_1$ and $\mathcal{V}_2 = \{v_{21}, v_{22}, \cdots, v_{2n_2}\}$, $n_2 = |\mathcal{V}_2|$ is the number of nodes in V_2. Then the probability that a causality C is applied is $Pr(s(C) = 0) = q^{n_1} = (1 - p)^{n_1}$. Hence, we consider the probability that there exists an α-giant component when facing probabilistic causal failures.

Reference List

[1] Y.-K. Lin, P.-C. Chang, and L. Fiondella, "A study of correlated failures on the network reliability of power transmission systems," *International Journal of Electrical Power & Energy Systems*, vol. 43, no. 1, pp. 954 – 960, 2012.

[2] M. Sedaghat, E. Wadbro, J. Wilkes, S. D. Luna, O. Seleznjev, and E. Elmroth, "Diehard: Reliable scheduling to survive correlated failures in cloud data centers," in *2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 52–59, May 2016.

[3] M. Ganjalizadeh, P. D. Marco, J. Kronander, J. Sachs, and M. Petrova, "Impact of correlated failures in 5G dual connectivity architectures for URLLC applications," 2019.

[4] Y.-K. Lin, L. Fiondella, and P.-C. Chang, "Quantifying the impact of correlated failures on system reliability by a simulation approach," *Reliability Engineering & System Safety*, vol. 109, pp. 32 – 40, 2013.

[5] M. Rahnamay-Naeini, J. E. Pezoa, G. Azar, N. Ghani, and M. M. Hayat, "Modeling stochastic correlated failures and their effects on network reliability," in *2011 Proceedings of 20th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–6, 2011.

[6] S. D. S. Reis, Y. Hu, A. Babino, J. S. A. Jr, S. Canals, M. Sigman, and H. A. Makse, "Avoiding catastrophic failure in correlated networks of networks," *Nature Physics*, vol. 10, pp. 762–767, 2014.

[7] S. Yang, S. Trajanovski, and F. A. Kuipers, "Optimization problems in correlated networks," *CoRR*, vol. abs/1502.06820, 2015.

[8] S. Nath, H. Yu, P. Gibbons, and S. Seshan, "Subtleties in tolerating correlated failures in wide-area storage systems.," *Proc. Symp. Networked Systems Design and Implementation (NSDI'06)*, 01 2006.

[9] M. Bakkaloglu, J. Wylie, C. Wang, and G. Ganger, "Modeling correlated failures in survivable storage systems," *Fast Abstract at International Conference on Dependable Systems & Networks*, 06 2002.

[10] D. V. Lindley and N. D. Singpurwalla, "On exchangeable, causal and cascading failures," *Statistical Sciences*, vol. 17, no. 2, pp. 209–219, 2002.

[11] A. Sen, A. Mazumder, J. Banerjee, A. Das, and R. Compton, "Identification of k most vulnerable nodes in multi-layered network using a new model of interdependency," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 831–836, 2014.

[12] J. Banerjee, A. Das, C. Zhou, A. Mazumder, and A. Sen, "On the entity hardening problem in multi-layered interdependent networks," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 648–653, 2015.

[13] J. Banerjee, C. Zhou, A. Das, and A. Sen, "On robustness in multilayer interdependent networks," in *Critical Information Infrastructures Security* (E. Rome, M. Theocharidou, and S. Wolthusen, eds.), (Cham), pp. 247–250, Springer International Publishing, 2016.

[14] Y. Almoghathawi and K. Barker, "Restoring community structures in interdependent infrastructure networks," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1355–1367, 2020.

[15] S. Hosseini, K. Barker, and J. E. Ramirez-Marquez, "A review of definitions and measures of system resilience," *Reliability Engineering & System Safety*, vol. 145, pp. 47 – 61, 2016.

[16] Z. Zhang, W. An, and F. Shao, "Cascading failures on reliability in cyber-physical system," *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1745–1754, 2016.

[17] W. Ellens and R. E. Kooij, "Graph measures and network robustness," *CoRR*, vol. abs/1311.5064, 2013.

[18] M. Garey, D. Johnson, and L. Stockmeyer, "Some simplified np-complete graph problems," *Theoretical Computer Science*, vol. 1, no. 3, pp. 237 – 267, 1976.

[19] A. Schrijver, "On the history of the transportation and maximum flow problems," *Mathematical Programming*, vol. 91, no. 3, pp. 437–445, 2002.

[20] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx," in *Proceedings of the 7th Python in Science Conference* (G. Varoquaux, T. Vaught, and J. Millman, eds.), (Pasadena, CA USA), pp. 11 – 15, 2008.

[21] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2020.

[22] University of Washington, ""Power systems test case archive -UWEE"," 1993.

[23] R. Deng, P. Zhuang, and H. Liang, "False data injection attacks against state estimation in power distribution systems," *IEEE Transactions on Smart Grid*, vol. 10, no. 3, pp. 2871–2881, 2019.

[24] W. Feng, J. Wu, C. Yuan, G. Liu, R. Dai, Q. Shi, and F. Li, "A graph computation based sequential power flow calculation for large-scale ac/dc systems," in *2019 IEEE Power Energy Society General Meeting (PESGM)*, pp. 1–5, 2019.

[25] J. Liu, J. Shyu, and C. Chu, "Probabilistic load margins of power systems embedded with wind farms," in *2013 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1540–1543, 2013.

[26] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *AAAI*, 2015.

[27] D. J. Watts and S. H. Strogatz, "Collective dynamics of small-world networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[28] D. Cornaz, F. Furini, M. Lacroix, E. Malaguti, A. R. Mahjoub, and S. Martin, "The vertex k-cut problem," *Discrete Optimization*, vol. 31, pp. 8 – 28, 2019.

[29] H. Cancela, F. Robledo, G. Rubino, and P. Sartor, "Efficient estimation of distance-dependent metrics in edge-failing networks," *International Transactions in Operational Research*, vol. 21, no. 2, pp. 199–213, 2014.

[30] A. Olivera, F. R. Amoza, and C. E. Testuri, "A grasp algorithm for a capacitated, fixed charge, multicommodity network flow problem with uncertain demand and survivability constraints," *International Transactions in Operational Research*, vol. 17, no. 6, pp. 765–776, 2010.

[31] A. Sedeño Noda and C. González-Martín, "Shortest path simplex algorithm with a multiple pivot rule: A comparative study," *Asia-Pacific Journal of Operational Research*, vol. 27, no. 06, pp. 677–691, 2010.

[32] R. S. Biswas, A. Pal, T. Werho, and V. Vittal, "A graph theoretic approach to power system vulnerability identification," *IEEE Transactions on Power Systems*, vol. 36, no. 2, pp. 923–935, 2021.

[33] D. Wu, F.-E. Wolter, B. Wang, and L. Xie, "Searching for the shortest path to voltage instability boundary: From euclidean space to algebraic manifold," *International Journal of Electrical Power & Energy Systems*, vol. 131, p. 107127, 2021.

[34] A. Ghasemkhani, H. Monsef, A. Rahimi-Kian, and A. Anvari-Moghaddam, "Optimal design of a wide area measurement system for improvement of power network monitoring using a dynamic multiobjective shortest path algorithm," *IEEE Systems Journal*, vol. 11, no. 4, pp. 2303–2314, 2017.

[35] M. Sasabe and T. Hara, "Capacitated shortest path tour problem-based integer linear programming for service chaining and function placement in nfv networks,"

IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp. 104–117, 2021.

[36] H. Cancela, M. El Khadiri, and L. A. Petingi, "Polynomial-time topological reductions that preserve the diameter constrained reliability of a communication network," *IEEE Transactions on Reliability*, vol. 60, no. 4, pp. 845–851, 2011.

[37] A. Satyanarayana and M. K. Chang, "Network reliability and the factoring theorem," *Networks*, vol. 13, no. 1, pp. 107–120, 1983.

[38] Z. Zhang and F. Shao, "A diameter-constrained approximation algorithm of multistate two-terminal reliability," *IEEE Transactions on Reliability*, vol. 67, no. 3, pp. 1249–1260, 2018.

[39] E. Mutafungwa, "Optical hop number limits imposed by various 2×2 cross-connect node designs," *Opt. Express*, vol. 9, pp. 400–410, Oct 2001.

[40] N. Skorin-Kapov, "Wdm optical networks planning using greedy algorithms," in *Greedy Algorithms* (W. Bednorz, ed.), ch. 30, Rijeka: IntechOpen, 2008.

[41] P. Leesutthipornchai, N. Wattanapongsakorn, and C. Charnsripinyo, "Multi-objective routing wavelength assignment in wdm network using spea2 approach," in *2009 9th International Symposium on Communications and Information Technology*, pp. 1057–1062, 2009.

[42] P. Leesutthipornchai, C. Charnsripinyo, and N. Wattanapongsakorn, "Solving multi-objective routing and wavelength assignment in wdm network using hybrid evolutionary computation approach," *Computer Communications*, vol. 33, no. 18, pp. 2246 – 2259, 2010.

[43] S. Soltan, A. Loh, and G. Zussman, "A learning-based method for generating synthetic power grids," *IEEE Systems Journal*, vol. 13, no. 1, pp. 625–634, 2019.

[44] J.-W. Wang and L.-L. Rong, "Robustness of the western united states power grid under edge attack strategies due to cascading failures," *Safety Science*, vol. 49, no. 6, pp. 807–812, 2011.

[45] B. Adenso-Díaz, J. Mar-Ortiz, and S. Lozano, "Assessing supply chain robustness to links failure," *International Journal of Production Research*, vol. 56, no. 15, pp. 5104–5117, 2018.

[46] S. Dong, X. Gao, A. Mostafavi, and J. Gao, "Modest flooding can trigger catastrophic road network collapse due to compound failure," *Communications Earth & Environment*, vol. 3, no. 1, p. 38, 2022.

[47] E. H. Sabri and B. M. Beamon, "A multi-objective approach to simultaneous strategic and operational planning in supply chain design," *Omega*, vol. 28, no. 5, pp. 581–598, 2000.

[48] Y. Hu and Q. Liu, "A network flow algorithm for solving generalized assignment problem," *Mathematical Problems in Engineering*, vol. 2021, p. 5803092, 2021.

[49] X. Han, X. Meng, Z. Yu, Q. Kang, and Y. Zhao, "A service function chain deployment method based on network flow theory for load balance in operator networks," *IEEE Access*, vol. 8, pp. 93187–93199, 2020.

[50] S. Mhanna, I. Saedi, and P. Mancarella, "Iterative lp-based methods for the multiperiod optimal electricity and gas flow problem," *IEEE Transactions on Power Systems*, pp. 1–1, 2021.

[51] Y.-K. Lin, "A simple algorithm for reliability evaluation of a stochastic-flow network with node failure," *Computers & Operations Research*, vol. 28, no. 13, pp. 1277–1285, 2001.

[52] W.-C. Yeh, "A simple mc-based algorithm for evaluating reliability of stochastic-flow network with unreliable nodes," *Reliability Engineering & System Safety*, vol. 83, no. 1, pp. 47–55, 2004.

[53] J. Tapolcai, L. Rónyai, B. Vass, and L. Gyimóthi, "List of shared risk link groups representing regional failures with limited size," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, 2017.

[54] M. W. Krentel, "The complexity of optimization problems," *Journal of Computer and System Sciences*, vol. 36, no. 3, pp. 490–509, 1988.

[55] K. P. Eswaran and R. E. Tarjan, "Augmentation problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 653–665, 1976.

[56] G. N. Frederickson and J. Ja'Ja', "Approximation algorithms for several graph augmentation problems," *SIAM Journal on Computing*, vol. 10, no. 2, pp. 270–283, 1981.

[57] T. Watanabe, T. Mashima, and S. Taoka, "The k-edge-connectivity augmentation problem of weighted graphs," in *Algorithms and Computation* (T. Ibaraki, Y. Inagaki, K. Iwama, T. Nishizeki, and M. Yamashita, eds.), (Berlin, Heidelberg), pp. 31–40, Springer Berlin Heidelberg, 1992.

[58] E. A. Dinitz, A. V. Karzanov, and M. V. Lomonosov, "On the structure of the system of minimum edge cuts of a graph," *Studies in Discrete Optimization*, pp. 290–306, 1976.

[59] J. Byrka, F. Grandoni, and A. J. Ameli, "Breaching the 2-approximation barrier for connectivity augmentation: A reduction to steiner tree," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2020, (New York, NY, USA), p. 815–825, Association for Computing Machinery, 2020.

[60] F. Cecchetto, V. Traub, and R. Zenklusen, "Bridging the gap between tree and connectivity augmentation: Unified and stronger approaches," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2021, (New York, NY, USA), p. 370–383, Association for Computing Machinery, 2021.

[61] Z. Zhang, S. Radhakrishnan, C. Subramanian, K. Barker, and A. D. González, "Causal node failures and computation of giant and small components in networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 3048–3060, 2021.

[62] M. E. Newman *et al.*, "Random graphs as models of networks," *Handbook of graphs and networks*, vol. 1, pp. 35–68, 2003.