Modeling, Analysis, and Control of Interdependent Networks

by

Jianan Zhang
B.Eng., Tsinghua University (2012)
S.M., Massachusetts Institute of Technology (2014)

Submitted to the Department of Aeronautics and Astronautics
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2018

© Massachusetts Institute of Technology 2018. All rights reserved.

Signature redacted

Author ....
Department of Aeronautics and Astronautics
July 1, 2018

Signature redacted

Certified by.... .....
Eytan Modiano
Professor
Thesis Supervisor

Signature redacted

Certified by..... .....
Edmund Yeh
Professor
Thesis Committee Member

Signature redacted

Certified by.... .....
Saurabh Amin
Associate Professor
Thesis Committee Member

Signature redacted

Accepted by . .....
Hamsa Balakrishnan
Associate Professor
Chair, Graduate Program Committee

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
OCT 09 2018
LIBRARIES

MIT Libraries

77 Massachusetts Avenue
Cambridge, MA 02139
http://libraries.mit.edu/ask

DISCLAIMER NOTICE

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available.

Thank you.

The images contained in this document are of the best quality available.

Modeling, Analysis, and Control of Interdependent Networks

by

Jianan Zhang

Submitted to the Department of Aeronautics and Astronautics
on July 1, 2018, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

The integrations of communication and physical networks facilitate network monitoring, operation and control, advancing the development of Internet of Things, smart power grids, and other cyber-physical systems. In these integrated networks, one network depends on another in order to be fully functional, leading to interdependence. The interdependence brings new challenges in the evaluation of network robustness under failures, and the design of control policies for mitigating failures, since failures may cascade from one network to another network that depends on it. We develop new models and analytical tools to study interdependent networks, with a focus on designing robust interdependent networks that can withstand failures and attacks.

We first model two interdependent networks of arbitrary topologies by layered graphs, where nodes in the demand layer depend on nodes in the supply layer. We study the supply node connectivity of the demand layer network: namely, the minimum number of supply node removals that would disconnect the demand network. We develop algorithms to evaluate the supply node connectivity given arbitrary network topologies and dependence between networks. We develop dependence assignment algorithms that maximize the supply node connectivity to enhance network robustness. We then study the robust routing problems: namely, delivering information or commodities through paths with high reliability. We develop algorithms to compute the path failure probability under correlated failures, and obtain the most reliable path for single-path routing and most reliable pair of paths for diverse routing between any pair of nodes in a network.

To study the formation and properties of large-scale interdependent networks, we develop an interdependent random geometric graph (RGG) model. The model represents two interdependent spatially embedded networks where interdependence exists between geographically nearby nodes in the two networks. We characterize the emergence of the giant mutual component in two interdependent RGGs as node densities increase, and obtain analytical bounds and confidence intervals for the percolation thresholds. This new model and analytical tools provide a framework for robustness evaluation of large-scale interdependent networks under uniform random node failures, geographical attacks, and degree-dependent failures that capture non-uniform

vulnerabilities of network components.

Finally, we consider two applications of interdependent networks. First, we consider interdependent power grid and communication network. We characterize the impact of communication failures on power grid control, and develop control policies for power grid frequency regulation and economic dispatch using limited communication. Second, we consider the robustness of distributed computing networks, where network flows depend on both communication and computation resources. We study the network robustness under the failure of network resources and solve network flow interdiction problems.

Thesis Supervisor: Eytan Modiano
Title: Professor

Thesis Committee Member: Edmund Yeh
Title: Professor

Thesis Committee Member: Saurabh Amin
Title: Associate Professor

Acknowledgments

First and foremost, I would like to thank my advisor Professor Eytan Modiano. His broad and deep knowledge, insight, experience, patience, caring and support helped me grow in the past six years and shaped my research philosophy. I could never overstate how grateful I am to have him as my advisor. The weekly meetings guided me to study fundamental problems and to overcome research obstacles. His insightful opinions would guide me through life.

I would like to thank my thesis committee members, Professor Edmund Yeh and Professor Saurabh Amin. Edmund introduced to me the field of network science. We had inspiring conversations and fruitful collaborations in the past four years. I am indebted to his guidance, kindness and encouragement. Saurabh advised me on game theory for interdiction. I would also thank him for his advice on my research.

I would like to thank the official thesis readers, Professor Hyang-Won Lee and Professor Sebastian Neumayer. They made invaluable comments to polish the thesis, and provided inspiring discussions for thesis extensions. Many thanks to Hyang-Won for the discussions and collaborations on research in the past year.

I would like to thank my other research collaborators, Jaime Llorca, Antonia Tulino, and David Hay in the past few years. Special thanks to Professor Moshe Zukerman, for leading me to the networking field when I was an undergraduate student, and the caring along the way.

I would like to thank all members of the Communication and Networking Research Group (CNRG) and other friends. The countless hours spent together, hangouts and conversations, are memories that will never fade away. Life would not be the same without my friends.

The thesis work has been supported by DTRA grants HDTRA1-13-1-0021 and HDTRA1-14-1-0058, which I would like to acknowledge.

Last, I would like to thank my parents, Guoliang Zhang and Huaqi Qi, and my wife, Yi Wan, for their unconditional love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The development of smart cities and cyber-physical systems has brought interdependence between once isolated networks and systems. In interdependent networks, one network depends on another to be fully functional. Examples include smart power grids [1,2], transportation networks [3,4], and layered communication networks [5,6]. Failures in one network not only affect the network itself, but also may cascade to another network that depends on it. For example, in the Italy blackout in 2003, an initial failure in the power grid led to reduced functionality of the communication network, which led to further failures in the power grid due to loss of communication and control [1,7]. Thus, the robustness of a network relies on both its own topology and the interdependence between different networks.

Interdependent networks have been extensively studied in the statistical physics literature based on random graph models since the seminal work of [7]. Nodes in two random graphs are interdependent, and a node is functional if both itself and its interdependent node are in the largest component of their respective graphs. If a positive fraction of nodes are functional as the number of nodes approaches infinity, the interdependent networks percolate. The condition for percolation is a measure of the robustness of the interdependent networks. While these models are analytically tractable, percolation may not be a key indicator for the functionality of infrastructure networks. For example, a network would lose most of its functionality when a large fraction of nodes are removed, while the graph still remains percolated.

A few models have been proposed for specific applications to capture the dependence between networks, such as interdependent power grids and communication networks [2, 8], and IP-over-WDM networks [6, 9]. These models consider finite size, arbitrary network topology, and incorporate dynamics in real-world networks. Instead of percolation, more realistic metrics are used to capture the robustness of interdependent networks, such as the amount of satisfied power demand, or traffic demand. These models are able to capture important performance metrics in real-world networks, at the cost of more complicated modeling and analysis.

In this thesis, we develop analytically tractable models for interdependent networks. We analyze the robustness of interdependent networks, and develop control policies to mitigate failures and to recover the network functionality. We first develop a layered graph model that captures interdependence between networks, and study the connectivity and reliable routing problems under node failures. We then develop an interdependent random geometric graph model to study properties of large-scale interdependent networks. Finally, we study two applications of interdependent networks – smart power grids and distributed computing networks. We formulate the dependence between different functions in these networks, and study the robustness and network performance under failures.

1.1 Literature review

In this section, we review related studies on interdependent networks modeling and analysis, and network robustness under correlated failures.

Interdependent networks modeling and analysis

Cascading failures in interdependent networks have been extensively studied based on random graph models [7]. After initial node failures in the first graph, their interdependent nodes in the second graph fail. Thus, a connected component in the second graph may become disconnected, and the failures of the disconnected nodes cascade back to (their interdependent) nodes in the first graph. As a result of

the cascading failures, removing a small fraction of nodes in the first random graph potentially destroys the giant components of both graphs. The analytical models and techniques have been applied to study failures under targeted node attacks [10] and failures in a network of networks [10, 11].

To model spatially embedded networks, an interdependent lattice model was studied in [12]. Under this model, geographical attacks may cause significantly more severe cascading failures than random attacks. Removing nodes in a finite region (*i.e.*, an infinitely small fraction of nodes) may destroy the infinite clusters in both lattices [13]. The spatial proximity in interdependent networks was studied in [14].

If every node in one network is interdependent with multiple nodes in the other network, and a node is content to have at least one interdependent node, failures are less likely to cascade [4, 15]. Although the one-to-multiple interdependence exists in real-world spatially embedded interdependent networks (*e.g.*, a control center can be supported by the electric power generated by more than one power generator), it has not been previously studied using spatial graph models. Partial interdependence (*i.e.*, some nodes do not have to depend on other nodes) also reduces the likelihood of cascading failures, and was studied in [16, 17].

Other works on the robustness of interdependent networks based on network science models studied different factors that affect the robustness, including inter-similarity between interdependent nodes [18–20], directed edges that connect nodes within each network [21], overloading [22], and considerations of functional small components [23, 24]. In these models, the dynamics of failure cascades were characterized after initial node failures. The protection of interdependent networks was studied in [25–27].

Other models have been proposed for interdependent power grids and communication networks [2, 8], which considers power flows governed by physical laws in addition to connectivity. The interactions between network components and failure cascades in power grids were studied in [28, 29]. The role of communication in mitigating power grid failures was studied in [30].

Interdependent networks have also been studied in the system reliability literature.

Focused on specific applications, most papers study the effects of random failures and intentional attacks by running simulations on graph topologies representing real-world networks. For example, the operations of an oil network depend on the power grid [31]. Gas pipeline networks controlled under the Supervisory Control and Data Acquisition system depend on a secure and connected communication network [32]. Besides empirical approaches, agent based approaches, system dynamics based approaches, and input-output models have been used [33]. These studies require large amount of data of real systems and rely on assumptions on system dynamics. Moreover, these approaches focus on the interactions between systems and do not consider the component-level interactions in a system.

Network robustness under correlated failures

In interdependent networks, more than one node in one network may depend on the same node in another network. Independent node failures in one network may lead to correlated failures in another network, which complicates the analysis of network robustness. We review related works on the shared risk group model and layered communication networks, which tackle failure correlations to evaluate network performance.

In the shared risk group model [6, 34–36], a set of edges or nodes share the same risk and can be removed by a single failure event. The model is used to study the robustness in layered communication networks such as IP-over-WDM networks. Suppose that a demand node has multiple supply nodes, and is content to have at least one supply node. The interdependent networks can be viewed as a generalized shared risk group model, given that the occurrences of multiple risks, instead of one single risk, are required to remove a node in the interdependent networks.

The shared risk group model can be represented by a colored graph (or labeled graph), in which edges or nodes that share the same risk have the same color (or label) [36–38]. Complexity results and approximation algorithms have been developed to compute the minimum number of colors that appear in an *edge cut* that disconnects a colored graph [36, 39]. In interdependent networks, we study node failures due to

the removals of their supply nodes. Thus, our focus is on the *node cut* in a colored graph with colored nodes and regular edges. While most results for edge cuts that separate a pre-specified source-destination pair (*i.e.*, $s-t$ edge cuts) can be naturally extended to $s-t$ node cuts, the extension is not obvious when the global edge or node cuts of a graph are considered. Although it is possible to transform a node cut problem in an undirected graph into an edge cut problem in a directed graph, the nature and analysis of the problem in a directed graph are different from the problem in an undirected graph, when global cuts are considered [40,41]. Thus, new techniques need to be developed to study the global node cuts in a colored graph.

While most studies on the shared risk group model have focused on the evaluation of robustness metrics of a given network, there have also been previous works that take a *network design* approach to optimize the metrics. For example, in optical networks, where two logical links share the same risk if they are supported by the same physical link, previous research developed lightpath routing algorithms that maximize the number of tolerable physical link failures [6,9]. We study the interdependence assignment that maximizes the number of supply node failures that a network can tolerate (to stay connected). Instead of solving difficult integer programs as in most network design literature, we apply graph algorithms, *e.g.*, the vertex sampling and graph partitioning techniques [42, 43], to develop polynomial time algorithms that have provable performance guarantees. The vertex sampling techniques provide bounds on the probability that the graph is connected after random node removals. We build connections between the node removals in a single graph and the node failures in interdependent networks, and study the connectivity of interdependent networks.

Robust routing problems, such as the problems of computing path failure probability and computing the most reliable path, have been extensively studied under correlated failure scenarios. It is difficult to find a path with any performance guarantee in general [44]. In the shared risk group model, the most reliable path contains the smallest number of risks if all risks are equally likely to occur, and can be obtained by integer programming [35,37,45,46].

In addition to the most reliable path, a backup path can be used to further improve

reliability, through *diverse routing*. Diverse routing problems have been studied under correlated link failures. The correlation between a pair of logical links is obtained either by measurement [47] or by analysis of the underlay physical topology [48]. Heuristic algorithms have been developed to find multiple reliable paths, and their performance was evaluated by simulation [47, 49, 50]. In contrast, we develop diverse routing algorithms that have provable performance, by explicitly bounding the gap between the failure probabilities of paths obtained by the algorithm and the optimal paths.

1.2 Contributions

We developed new models and analytical tools to study interdependent networks, with a focus on performance evaluation and design of robust interdependent networks that can withstand failures and attacks. We develop metrics that measure the robustness of interdependent networks, by generalizing canonical metrics for the robustness of a single network. The models are simple enough to allow for the evaluation of the robustness of interdependent networks, and to obtain insights and principles for designing robust interdependent networks. We consider two applications of interdependent networks – smart grid and distributed computing networks. We develop control policies for power grid under communication failures, and study the interdiction problems in computing networks, where network flows depend on both communication and computation resources.

Network connectivity

In Chapter 2, we develop a layered graph model to represent two interdependent networks, where demand nodes are supported by supply nodes. We add a minimal ingredient to the classical graph model to capture interdependence, and define supply node connectivity as a robustness metric for our model, analogous to the widely accepted cut metric (node connectivity) for the classical graph model. We prove the complexity, and develop integer programs to evaluate the supply node connectivity,

both for a given pair of nodes and for the entire network. Moreover, we propose a polynomial time algorithm that computes the supply node connectivity for a special class of problems, based on which we develop an approximation algorithm for the general problem.

In addition, we study the network design problem of improving the robustness of interdependent networks by assigning interdependence between two networks. We propose a simple assignment algorithm that maximizes the supply node connectivity of an $s - t$ pair, by assigning node-disjoint paths with different supply nodes while allowing nodes in the same path to have the same supply node. Based on a similar idea and considering disjoint connected dominating sets, we develop an assignment algorithm that approximates the optimal global supply node connectivity to within a polylogarithmic factor. Finally, we propose a random assignment algorithm under which, with high probability, the global supply node connectivity is within a constant factor from the optimal in most cases, and at worst is within a logarithmic factor from the optimal.

Reliable routing

In Chapter 3, we study robust routing problems in interdependent networks, by characterizing the effects of failures in one network on the other network. Given that a demand node fails if it loses all of its supply nodes and that a supply node may support multiple demand nodes, independent supply node failures may lead to correlated demand node failures. We develop techniques to tackle the failure correlation, and study robust routing problems in interdependent networks.

We prove the complexity of computing path failure probability in interdependent networks, and develop a polynomial time randomized approximation scheme to evaluate the path failure probability. We develop algorithms to find the most reliable path between a pair of nodes for single-path routing. In addition, we study the diverse routing problem in interdependent networks. We prove the complexity, and develop approximation algorithms to compute the failure probability of two paths, and to compute a pair of reliable paths whose failure probability is minimized.

Percolation of large-scale interdependent networks

In Chapter 4, we propose an interdependent random geometric graph (RGG) model for two interdependent networks, to study the properties of large-scale interdependent networks. Compared with existing network science models for interdependent networks, the interdependent RGG model captures the differences in the scales of two networks as well as the one-to-multiple interdependence in spatially embedded networks. We derive the first analytical upper bounds on the percolation thresholds of the interdependent RGGs, above which a positive fraction of nodes are functional. We obtain 99% confidence intervals for the percolation thresholds, by mapping the percolation of interdependent RGGs to the percolation of a square lattice where the probability that a bond in the square lattice is open is evaluated by simulation.

We characterize sufficient conditions for the interdependent RGGs to percolate under random failures and geographical attacks. In particular, if the node densities are above any upper bound on the percolation threshold obtained in this chapter, the interdependent RGGs remain percolated after a geographical attack. This is in contrast with the cascading failures after a geographical attack, observed in the interdependent lattice model with one-to-one interdependence.

Finally, we extend our techniques to study models with more general interdependence requirement (*e.g.*, a node in one network requires more than one supply node from the other network).

Power grid frequency control with limited communication

In Chapter 5, we study the control for smart grid under failures, as an application of interdependent networks. Power grid frequency regulation and economic dispatch is traditionally implemented in a hierarchical architecture, and require communication to exchange information between generators. We study the impact of loss of communication on power grid control.

We study the performance of a decentralized integral controller with properly designed controller gains, which does not require any communication. We quantify the

gap between the cost under the decentralized control and the minimum possible cost, and derive conditions for joint frequency regulation and economic dispatch, based on the DC power flow model. We study the tradeoff between the cost and the convergence time, by changing the parameters of the controller. We also study the benefit of reducing the convergence time from communication, and quantify the importance of each individual communication link. We then extend the control policy to handle arbitrary convex power generation costs and power generation capacity constraints. Moreover, we observe that a delayed integral control achieves near-optimal generation cost using significantly smaller convergence time.

Robustness of distributed computing networks

In Chapter 6, we study the robustness of distributed computing networks, where network flows are supported by both communication and computation resources. We define cut metrics, including communication cut, computation cut, and joint communication and computation cut, to characterize the vulnerability of computing networks under failures. We develop efficient algorithms to compute the maximum flow and the minimum computation cut. We prove the complexity for computing the minimum communication cut and the joint cut, and develop integer programs and approximation algorithms to evaluate the cuts. Unlike the classical flow network where the maximum flow equals the minimum cut, in computing networks, there is a gap between the flow and cut values. We then study network flow interdiction problems in computing networks, and develop integer linear programs to compute the optimal interdiction.

Optimal traffic control for distributed computing networks

In the Appendix, we develop routing and scheduling algorithms for traffic flows in a distributed computing network, where the flows are processed by a chain of service functions. The control policy achieves the maximum network throughput by the joint optimization of communication and computation resource allocation,

and can be adaptive to time-varying resource availability and failures. The policy is throughput-optimal for any mix of unicast and multicast traffic, and is the first throughput-optimal policy for non-unicast traffic in distributed computing networks with both communication and computation constraints. Moreover, the policy yields substantially lower average packet delay compared with existing control policies.

1.3 Organization

The rest of the thesis is organized as follows. In Chapter 2, we present a layered graph model for interdependent networks, and study network connectivity. In Chapter 3, we study robust routing in interdependent networks based on the same model. In Chapter 4, we develop an interdependent random geometric graph model to study large-scale interdependent networks, and derive sufficient conditions for percolation. In Chapter 5, we study power grid frequency control using limited communication. In Chapter 6, we analyze the robustness of distributed computing networks and study flow interdiction problems. We conclude the thesis in Chapter 7. In the Appendix, we develop a control policy for traffic flows in distributed computing networks, which achieves the maximum throughput is adaptive to failures of network resources.

Chapter 2

Connectivity in interdependent networks

In this chapter, we propose a layered graph model for two interdependent networks, and study the impacts of node failures in one network on the other network. We develop metrics that measure the robustness of interdependent networks, by generalizing canonical metrics for the robustness of a single network. Moreover, the model is simple enough to allow for the evaluation of the robustness of interdependent networks, and allows us to obtain insights and principles for designing robust interdependent networks.

The rest of the chapter is organized as follows. In Section 2.1, we develop a one-way dependence model, where a demand network depends on a supply network. This allows us to deliver key results and intuitions for studying the impacts of node failures in one network on its interdependent network, using simplified notations and presentations. We study this one-way dependence model in Sections 2.2 and 2.3. In Section 2.2, we evaluate the supply node connectivity of the demand network. In Section 2.3, we develop algorithms, which assign supply nodes to demand nodes, to maximize the supply node connectivity. In Section 2.4, we focus on the bidirectional interdependence model and generalize the above results. Section 2.5 provides simulation results. Finally, Section 2.6 summarizes this chapter.

2.1 Model

2.1.1 One-way dependence model

We start by considering a one-way dependence model, where nodes in a *demand network* depend on nodes in a *supply network*. This simplified model allows us to focus on the impacts of node failures in one network on the other network. Let two undirected graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ represent the topologies of the demand and supply networks, respectively. Each node in the demand network depends on one or more nodes in the supply network. The dependence is represented by the directed edges in Fig. 2-1. Every supply node provides substitutional supply to the demand nodes. A demand node is functional if it is adjacent to at least one supply node. Figure 2-1 illustrates the failure of a demand node due to the removals of its supply nodes.

As a more concrete example, we use G_1 to represent a communication network and G_2 to represent a power grid. Each node in G_1 represents a router, and each node in G_2 represents a power station. A router receives power from one or more power stations, and fails if all of the supporting power stations fail.

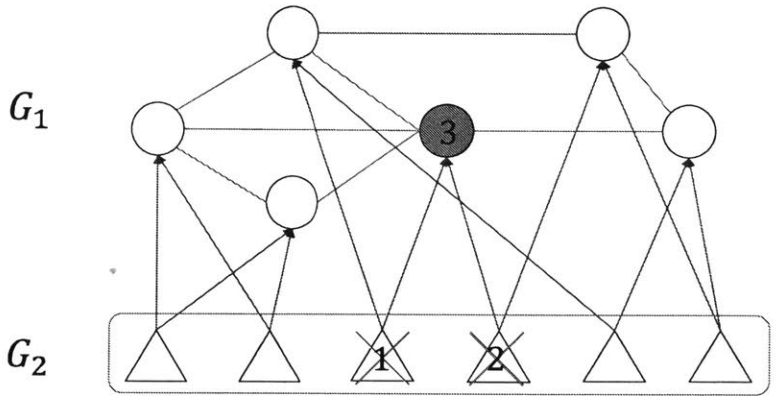


Figure 2-1: Failure cascades: demand node 3 fails if both supply nodes 1 and 2 fail.

We aim to characterize the impacts of node removals in the supply network on the connectivity of the demand network. Recall that (see, *e.g.* [51]), in a single graph, a *node cut* (*i.e.*, vertex cut) is a set of nodes whose removals either disconnect the graph into more than one connected component, or make the remaining graph trivial

(where a single node remains). The *node connectivity* of a graph is the number of nodes in the smallest node cut. In the one-way dependence model, the connectivity of the demand network depends not only on its topology $G_1(V_1, E_1)$, but also on the supply-demand relationship. We define the *supply node cut* and *supply node connectivity* of the demand network as follows.

Definition 2.1. A *supply node cut* of the demand graph is a set of supply nodes whose removals induce a node cut in the demand graph. (Mathematically, a supply node cut of G_1 is a set of nodes $V_s \subseteq G_2$, such that nodes $V_d \subseteq G_1$ do not have any supply nodes other than V_s, and that V_d contain a node cut of G_1.)

The *supply node connectivity* is the number of nodes in the smallest supply node cut.

The above definition is a generalization of the traditional node cut to include a superset of a cut. This is necessary because the removals of supply nodes may not correspond to proper cuts of the demand graph (see Fig. 2-2). Under this definition, graphs with larger supply node connectivity are more robust under supply node failures.

Remark. In Fig. 2-2, let every node have a single supply node, and let the red nodes share the same supply node $u \in G_2$. By removing u, the left graph stays connected after removing all the three red nodes, while the right graph is disconnected. However, the left graph is less robust under the removal of supply node u, because the failed nodes in the left figure include all the failed nodes in the right figure. Thus, "graph connectivity after supply node removals" does not serve as a good measure for the robustness of the demand graph when supply nodes fail. This motivates our definition of supply node cut and supply node connectivity. According to our definition, the supply node connectivity of the left graph is one.

We study the connectivity of a source-destination pair $(s, t) \in G_1$ as a starting point, which provides insights towards the graph connectivity with simpler analysis. In a graph, an *st node cut* is a set of nodes, excluding s and t, whose removals disconnect s from t. The number of nodes in the smallest $s - t$ node cut is the

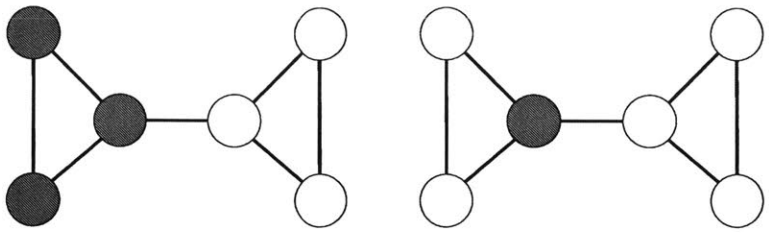


Figure 2-2: Illustration of supply node cut. Let the three red nodes in the left figure be supported by the same supply node. Removing the supply node leads to the failure of the three red nodes, which do not form a proper cut but form a superset of a proper cut (*i.e.*, the red node in the right figure). The supply node is viewed as a supply node cut.

$s-t$ *node connectivity*. Analogously, we define $s-t$ *supply node cut* and $s-t$ *supply node connectivity* as follows.

Definition 2.2. An $s-t$ *supply node cut* is a set of supply nodes whose removals induce an $s-t$ node cut. (Mathematically, an $s-t$ supply node cut is a set of nodes $V_s^{st} \subseteq G_2$, such that nodes $V_d^{st} \subseteq G_1$ do not have any supply nodes other than V_s^{st}, and that V_d^{st} contain an $s-t$ node cut.)

The $s-t$ *supply node connectivity* is the number of nodes in the smallest $s-t$ supply node cut.

An $s-t$ supply node cut may induce demand node failures V_d^{st} including s and/or t, since s, t may share the same supply nodes with nodes in the $s-t$ node cut. However, removing $V_d^{st} \setminus \{s,t\}$ must disconnect s from t.

We consider non-adjacent s and t throughout the chapter. Otherwise, if s and t are adjacent, they are always connected when other nodes are removed, and there is no node cut that disconnects them.

2.1.2 Transformation to a colored graph

Our model is closely related to the shared risk node group (SRNG) model [36,52]. In the SRNG model, several nodes share the same risk, and can be removed by a single failure event. In interdependent networks, if every node has *one* supply node, then the demand graph becomes exactly the same as the SRNG model, where the

demand nodes that have the same supply node share the same risk.

The SRNG model can be represented by a *colored graph*, where the nodes that have the same color share a common risk. We define[1] *color node cut* and $s-t$ *color node cut* as follows.

Definition 2.3. Given a colored graph $G(V, E, \mathcal{C})$ with colored nodes V, regular edges E, and node-color pairs $\mathcal{C}$ that represent the color for each node, a *color node cut* is a set of colors C_c such that the nodes covered by colors C_c contain a node cut of G.

A *minimum color node cut* of G is a color node cut $C_{c\min}$ that has the minimum number of colors. The number of colors in $C_{c\min}$ is the *value* of the minimum color node cut.

Definition 2.4. Given a colored graph $G(V, E, \mathcal{C})$ with colored nodes V, regular edges E, node-color pairs $\mathcal{C}$ that represent the color for each node, and a pair of nodes $(s, t) \in V$, a *color* $s-t$ *node cut* is a set of colors C_c^{st} such that the nodes covered by colors C_c^{st} contain an $s-t$ node cut.

A *minimum color* s–t *node cut* is a color s–t node cut $C_{c\min}^{st}$ that has the minimum number of colors. The number of colors in $C_{c\min}^{st}$ is the *value* of the minimum color $s-t$ node cut.

Colored graph provides an intuitive representation of the correlated node failures by color. If every demand node has a single supply node, then every demand node has a color that corresponds to its supply node. After the failure of a supply node, a demand node fails if it has the color that corresponds to the supply node.

In general, a demand node can have multiple supply nodes, and thus the mapping to a colored graph is not straightforward. We propose Algorithm 2.1 that transforms the demand network to a colored graph where every node has a single color, and use Fig. 2-3 to illustrate the algorithm.

[1]Previous study on colored graphs focused on color edge cuts in colored graphs with colored edges and regular nodes. Much less is known about the color node cut, a counterpart of color edge cut, in colored graphs with colored nodes and regular edges. In fact, to the best of our knowledge, there is no formal definition for color node cut.

Algorithm 2.1 Transformation from the demand graph G_1 to a colored graph $\tilde{G}_1$.

1. If a node $v_i \in G_1$ has $n_s(v_i)$ supply nodes, $n_s(v_i)$ copies of v_i exist in $\tilde{G}_1$. Each copy has a color which identifies a supply node. No edge exists between the copies of v_i.
2. If v_i and v_j are connected by an edge in G_1, then all the copies of v_i are connected to all the copies of v_j in $\tilde{G}_1$.

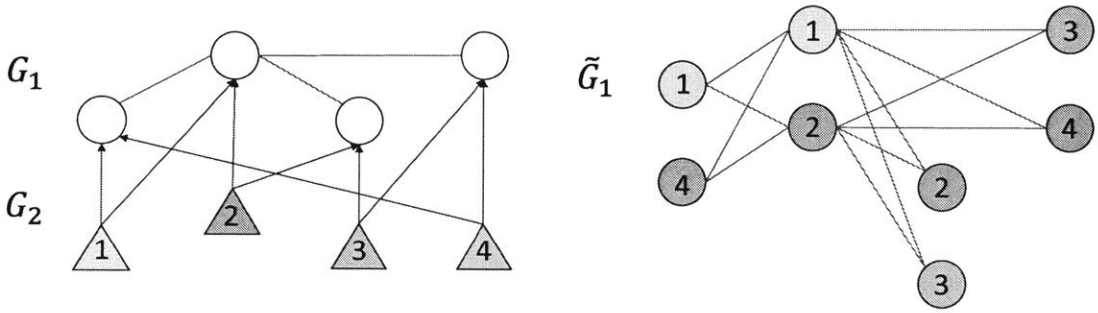


Figure 2-3: Illustration of the transformation algorithm.

We study the connectivity of the demand graph based on the colored graph, due to the following theorem.

Theorem 2.1. *There is a one-to-one mapping between a supply node cut in the demand network and a color node cut in the transformed graph of the demand network.*

Proof. Let G_1 be the demand graph and G_2 be the supply graph. Let $\tilde{G}_1$ be the transformed graph of G_1 by Algorithm 2.1. The result trivially holds if every demand node has a single supply node. Next we focus on the case where a demand node has more than one supply node.

We first prove that given any supply node cut V_s of G_1, there exists a color node cut C_c of $\tilde{G}_1$ where colors C_c correspond to supply nodes V_s. According to the definition of a supply node cut, the demand nodes in G_1 that have no supply nodes other than V_s contain a node cut V_d^* of G_1. By removing V_d^* from G_1, either G_1 is separated into at least two components, or a single node v in G_1 remains (by the definition of a node cut for a graph). In the first case, nodes in $\tilde{G}_1$ that correspond to $V_d^* \subseteq G_1$ have colors in C_c and they are removed. Among the remaining nodes, if no edge exists

between two nodes in G_1, then there is no edge between their corresponding nodes in $\tilde{G}_1$. Therefore, the remaining nodes in $\tilde{G}_1$ are disconnected after removing the nodes that correspond to V_d^* and have colors C_c. In the second case, copies of v are the only remaining nodes in $\tilde{G}_1$ and they are disconnected. Thus, C_c is a color node cut in $\tilde{G}_1$ in both cases.

We then prove that given any color node cut C_c of $\tilde{G}_1$, there exists a supply node cut V_s of G_1 where V_s corresponds to colors C_c. After removing all (or a subset) of nodes in $\tilde{G}_1$ that have colors C_c, either a single node remains in $\tilde{G}_1$, or $\tilde{G}_1$ is separated into multiple connected components. In the first case, at most a single node remains in G_1 after removing V_s, and thus V_s is a supply node cut. In the second case, if every component contains a single node, and the node corresponds to the same node in G_1, then at most one node survives in G_1 by removing supply nodes V_s. On the other hand, if these components correspond to different nodes in G_1, there must exist two disconnected nodes $v_1, v_2 \in G_1$, whose copies are in different components in $\tilde{G}_1$. (Recall that, if two nodes are connected in G_1, then their copies are connected in $\tilde{G}_1$. If all the remaining nodes in G_1 form a connected component, then their corresponding copies in $\tilde{G}_1$ also form a connected component.) In both cases, V_s is a supply node cut of G_1. □

Moreover, an $s-t$ supply node cut can be represented by a color $\tilde{s}\tilde{t}$ node cut in the colored graph, where $\tilde{s}$ is any copy of s and $\tilde{t}$ is any copy of t. By considering cuts that separate (s,t) in G_1 and cuts that separate $(\tilde{s},\tilde{t})$ in $\tilde{G}_1$, we obtain the following result by a similar proof to that of Theorem A.1.

Corollary 2.1. *There is a one-to-one mapping between a supply node $s-t$ cut in the demand network and a color $\tilde{s}\tilde{t}$ node cut in the transformed graph of the demand network, where $\tilde{s}$ is any copy of s and $\tilde{t}$ is any copy of t.*

Another corollary is a property of the transformed graph $\tilde{G}_1$ when every demand node in G_1 has a fixed number n_s of supply nodes. If G_1 has n_1 nodes and m_1 edges, the transformed graph $\tilde{G}_1$ has $n_1 n_s$ nodes and $m_1 n_s^2$ edges. Moreover,

Corollary 2.2. *If every demand node has a fixed number n_s of supply nodes, the following results hold.*

If the node connectivity of G_1 is k_1, then the node connectivity of $\tilde{G}_1$ is $k_1 n_s$.

If the $s-t$ node connectivity is k_1^{st} $(s,t \in G_1)$, then the $\tilde{s}\tilde{t}$ node connectivity is $k_1^{st} n_s$, where $\tilde{s} \in \tilde{G}_1$ is any copy of s and $\tilde{t} \in \tilde{G}_1$ is any copy of t.

Proof. By assigning n_s distinct supply nodes to each node in G_1, using a total of $n_1 n_s$ supply nodes, to remove a node in G_1, a distinct set of n_s supply nodes must be removed. Thus, the supply node connectivity of G_1 equals the node connectivity of G_1 times n_s. Moreover, in $\tilde{G}_1$, every node has a distinct color, and the number of colors in a color node cut equals the number of nodes in the same node cut. Thus, the node connectivity of $\tilde{G}_1$, without considering colors, equals the supply node connectivity of G_1, because of the one-to-one mapping proved in Theorem A.1. We have therefore proved that the node connectivity of $\tilde{G}_1$ is the node connectivity of G_1 times n_s. The same relationship holds for $s-t$ node connectivity in G_1 and $\tilde{s}\tilde{t}$ node connectivity in $\tilde{G}_1$. □

2.1.3 Notations

We define notations to be used throughout the rest of the chapter. For a finite set A, the cardinality of A is denoted by $|A|$. For a colored graph $G(V, E, C)$, the number of nodes, edges, and colors are denoted by n, m, n_c, respectively. The graph connectivity is denoted by k, and the $s-t$ connectivity is denoted by k^{st}. The subscript $i \in \{1, 2\}$ denotes the identity of a graph. For example, n_1 denotes the number of nodes in G_1. The subscript s denotes *supply*. For example, n_{s1} denotes the number of supply nodes for a node in G_1.

We use asymptotic notations in this chapter. Let $f(x) > 0$ and $g(x) > 0$ be two functions. If there exists a constant M and a positive number x_0, such that $f(x) \leq Mg(x)$ for all $x \geq x_0$, then $f(x) = O(g(x))$. Moreover, $f(x) = \Omega(g(x))$ if $g(x) = O(f(x))$; $f(x) = \Theta(g(x))$ if both $f(x) = O(g(x))$ and $f(x) = \Omega(g(x))$; $f(x) = o(g(x))$ if $\lim_{x\to\infty} f(x)/g(x) = 0$; $f(x) = \omega(g(x))$ if $g(x) = o(f(x))$.

2.2 Evaluation of the supply node connectivity

In this section, we study the supply node connectivity of the demand network. As discussed in the previous section, supply node cuts in the demand network are equivalent to color node cuts in a colored graph. To simplify the presentation, we consider a colored graph $G(V, E, C)$ throughout this section.

2.2.1 Complexity

We prove that computing both the global minimum color node cut of a graph and the minimum color $s-t$ node cut are NP-hard. The proof for the complexity of the minimum color $s-t$ node cut follows a similar approach to that of the minimum color $s-t$ edge cut in [36]. In contrast, the complexity of the global minimum color edge cut is unknown. The detailed proofs of Theorems 2.2 and 2.3 can be found in the Appendix.

Theorem 2.2. *Given a colored graph, computing the value of the global minimum color node cut is NP-hard.*

Theorem 2.3. *Given a colored graph and a pair of nodes (s, t), computing the value of the minimum color st node cut is NP-hard.*

Given the computational complexity, in the remainder of this section, we first develop integer programs to compute the exact value of the minimum color cuts, and then develop polynomial time approximation algorithms.

2.2.2 Exact computation for arbitrary colored graphs

We compute the minimum color $s-t$ node cut using a mixed integer linear program (MILP). In this formulation, each node has a *potential*. Connected nodes have the same potential. The source and the destination are disconnected if they have different potentials. We note that the classical MILP formulation for computing the minimum *edge cut* also uses node potentials to indicate disconnected components after removing edges [53].

In the MILP formulation, indicator variable c_r denotes whether color $r \in C$ is in the minimum color cut, where C is the set of colors in the colored graph. Indicator variable y_v denotes whether node $v \in V$ is a cut node that separates the $s-t$ pair, and may take value 1 only if the color of v is in the color cut. Note that y_v may take value 0 even if the color of v is in the color cut (constraint (2.4)). This allows the cut nodes to be a *subset* of nodes with colors $\{r|c_r = 1\}$ (recall Definition 2.4).

The potential of a node v is denoted by p_v. After removing all the cut nodes, the potentials of nodes in a connected component are the same, guaranteed by constraints (2.1) under the condition $y_i = y_j = 0$. The same constraints guarantee that nodes adjacent to the cut nodes may have different potentials from the cut nodes, if $y_i = 1$ or $y_j = 1$. The potential of the source is 0, and the potential of the destination is 1, guaranteed by constraint (2.2). Moreover, constraint (2.3) guarantees that neither s nor t is a cut node. Thus, the component that contains s and the component that contains t are separated by an $s-t$ node cut. The objective is to minimize the number of colors of the cut nodes.

$$
\begin{aligned}
\min \quad & \sum_{r \in C} c_r && \text{(MILP)} \\
\text{s.t.} \quad & -y_i - y_j \leq p_i - p_j \leq y_i + y_j, \quad \forall (i,j) \in E, && (2.1) \\
& p_s = 0, p_t = 1, && (2.2) \\
& y_s = y_t = 0, && (2.3) \\
& y_v \leq c_r, \forall r \in C, v \in \{v | r \text{ is the color of } v\}, && (2.4) \\
& p_v, y_v \geq 0, \quad \forall v \in V, \\
& c_r \in \{0,1\}, \quad \forall r \in C.
\end{aligned}
$$

Next we compute the global minimum color node cut of a colored graph using an integer program (IP). The variables c, y, p have the same representations as those in the above MILP. Recall that a global node cut of a graph *either* separates the remaining nodes into disconnected components, *or* makes the remaining graph trivial. In the first case, $z = 0$, and constraint (2.6) guarantees that there is at least one node

that has potential 1, in addition to all the cut nodes. Constraints (2.5) guarantee that all the cut nodes have potential 1. Constraint (2.7) guarantees that there is at least one node that has potential 0. The existence of both potential 0 nodes and potential 1 nodes, excluding the cut nodes, implies that the remaining graph is disconnected. In the second case, $z = 1$, and the number of cut nodes is at least $|V| - 1$, guaranteed by constraint (2.8). Given that M is sufficiently large (*e.g.*, $M = 2|V|$), if $z = 0$, constraint (2.8) is satisfied; if $z = 1$, constraints (2.6) and (2.7) are satisfied. Thus, a node cut that satisfies either condition is a feasible solution of the following IP.

$$
\begin{aligned}
\min \quad & \sum_{r \in C} c_r && \text{(IP)} \\
\text{s.t.} \quad & -y_i - y_j \leq p_i - p_j \leq y_i + y_j, \quad \forall (i,j) \in E, \\
& p_v \geq y_v, \quad \forall v \in V, && (2.5) \\
& \sum_{v \in V} p_v - \sum_{v \in V} y_v - 1 \geq -Mz, && (2.6) \\
& \sum_{v \in V} p_v - |V| + 1 \leq Mz, && (2.7) \\
& \sum_{v \in V} y_v - |V| + 1 \geq -M(1-z), && (2.8) \\
& y_v \leq c_r, \quad \forall r \in C, v \in \{v | r \text{ is the color of } v\}, \\
& c_r, p_v, y_v, z \in \{0,1\}, \quad \forall v \in V, \forall r \in C.
\end{aligned}
$$

2.2.3 A polynomially solvable case and an approximation algorithm

Although computing the minimum color node cut is NP-hard in general, there are special instances for which the value can be computed in polynomial time. Let V_i denote the nodes in G that have color i. The *induced graph* of V_i, denoted by $G[V_i]$, consists of V_i and edges of G that have both ends in V_i. We prove that if $G[V_i]$ is connected for all i, then the minimum color node cuts can be computed in polynomial time. It is worth noting that these special instances are reasonable representations for

real-world interdependent networks, where a supply node is likely to support multiple directly connected nearby demand nodes.

Algorithm 2.2 computes the minimum color $s-t$ node cut in G where $G[V_i]$ is connected $\forall i$, for a non-adjacent (s,t) pair.

Algorithm 2.2 Computation of the minimum color $s-t$ node cut in G where $G[V_i]$ is connected $\forall i$.

---

1. Construct a new graph G' from G as follows. Contract the nodes V_i, which have the same color i, into a single node u_i. Connect u_i and u_j if and only if there is at least one edge between V_i and V_j. Connect s' to $\{u_i | s \text{ is connected to } V_i\}$, and connect t' to $\{u_i | t \text{ is connected to } V_i\}$.
2. Compute the minimum $s't'$ node cut in G', in which every node has a distinct color. The minimum color $s-t$ node cut in G is given by the colors of the $s't'$ cut nodes in G'.

The following lemma proves the correctness of Algorithm 2.2.

Lemma 2.1. *The $s't'$ node connectivity in G' equals the value of the minimum color st node cut in G, if $G[V_i]$ is connected, $\forall i$, and (s,t) are non-adjacent.*

Proof. We aim to prove that there is a one-to-one mapping between a color $s-t$ node cut in G and an $s't'$ node cut in G', from which the result follows.

One direction is simple. Let C be the set of colors that appear in G. For any $s-t$ color node cut C_c^{st} in G, after removing all (or a certain subset) of nodes with colors in C_c^{st}, there does not exist a sequence of colored nodes that connect s and t. Two nodes u_i, u_j are connected in G' only if nodes with color i and nodes with color j are connected in G. Thus, there does not exist a sequence of nodes with colors in $C \setminus C_c^{st}$ that connect s' and t' in G'.

To prove the other direction, consider any $s't'$ node cut in G' and denote it by $V^{s't'}$. Let $V^{st} \subseteq G$ be a set of nodes with colors in $C_{\text{color}} = \{i | u_i \in V^{s't'}\}$. We aim to prove that V^{st} is a superset of an $s-t$ node cut in G.

If V^{st} does not contain s or t, after removing V^{st} from G, no edge exists between the component that contains s and the component that contains t. To see this, note

that if no edge exists between u_i and u_j in G', then no edge exists between any color i node and any color j node in G.

If V^{st} contains s, we need to prove that $V^{st} \setminus s$ is an $s-t$ cut in G. In Step 1 of Algorithm 2.2, s' is connected to all neighbors $N(s') := \{u_i | s \text{ is connected to } V_i\}$ in G'. After removing $V^{s't'}$, $N(s')$ are either removed or disconnected from t'. Therefore, the neighbors of s in G are either removed or disconnected from t after removing $V^{st} \setminus s$.

The same analysis proves that if V^{st} contains t, then $V^{st} \setminus t$ is an $s-t$ cut in G. Similarly, if V^{st} contains both s and t, then $V^{st} \setminus s,t$ is an $s-t$ cut in G. This concludes the proof that V^{st} is a superset of an $s-t$ node cut in G. □

Remark. A similar result exists in the computation of the minimum color $s-t$ *edge* cut under the condition that all the edges that have the same color are connected [36]. The difference in our problem is that the source or destination may have the same color as the nodes in a cut. Thus, to prove that a set of colors C_c^{st} is a color cut, we need to prove that removing nodes, excluding s and t, with colors C_c^{st} disconnects s and t. Thus, the proof has to take care of multiple corner cases.

To compute the global minimum color node cut of a colored graph, it is necessary to consider two different cases, resulting from the definition of a node cut that allows the remaining graph to be either disconnected or reduced to a single node. Algorithm 2.3 computes the exact value of the global minimum color node cut of G where $G[V_i]$ is connected $\forall i$.

We remark that the global minimum color node cut of G *can not* be computed by first contracting nodes that have the same color and then computing the global minimum node cut in the new graph, even if $G[V_i]$ is connected $\forall i$. We only claim that the minimum color $s-t$ node cut in G corresponds to the $s't'$ node cut in G' obtained by Algorithm 2.2, and that the global minimum color node cut of G can be computed by Algorithm 2.3. Note that the topology of G' depends on the choice of s and t (see Step 1 of Algorithm 2.2).

The above result can be used to develop an approximation algorithm to compute the minimum color node cuts in an arbitrary colored graph where the induced graph

Algorithm 2.3 Computation of the global minimum color node cut of G where $G[V_i]$ is connected $\forall i$.

1. Compute minimum color $s-t$ node cut C_c^{st} for all non-adjacent $s-t$ pairs in G by Algorithm 2.2. Let C_c^1 denote the minimum size C_c^{st} over all $s-t$ pairs. (The cut C_c^1 is the minimum color node cut of G that partitions G into more than one component.)
2. Compute the minimum set of colors C_c^2 that cover at least $n-1$ out of the n nodes in G. (*I.e.*, if there exists a color i that is carried by one node, then C_c^2 include all the colors except color i. If there is no color that is carried by a single node, then C_c^2 include all the colors.)
3. The minimum color node cut of G is given by the smaller of C_c^1 and C_c^2.

$G[V_i]$ is not necessarily connected. To approximate the value of the minimum color $s-t$ node cut, the algorithm is a slight modification of Algorithm 2.2. Instead of contracting $G[V_i]$ into a single node, in the new algorithm, *each connected component* of $G[V_i]$ is contracted into a single node. Let the new graph be G'', and connect s'', t'' to the nodes contracted by the components in G that are connected to s, t, respectively. The performance of the algorithm is given by Lemma 2.2.

Lemma 2.2. *The $s''t''$ node connectivity in G'' is at most q times the value of the minimum color $s-t$ node cut in G, where q is the maximum number of components of $G[V_i]$, $\forall i$.*

Proof. Given that the induced graph $G[V_i]$ has at most q components, after contracting each component into a node with color i, the number of nodes with color i in G'' is at most q. Let C_c^{st} denote a color node cut in G. By a similar reasoning as the proof of Lemma 2.1, removing nodes with colors C_c^{st} disconnects s'' from t'' in G''. Let $c_{\min}^{st}$ denote the value of the minimum color st node cut $C_{c\min}^{st}$ in G. The number of nodes in G'' with colors $C_{c\min}^{st}$ is at most $c_{\min}^{st}q$. Moreover, $c_{\min}^{st}q$ is no smaller than the $s''t''$ node connectivity $k^{s''t''}$. Equivalently, $c_{\min}^{st}$ is at least $k^{s''t''}/q$. □

The global minimum color node cut of G can be approximated to within factor q, by approximating the minimum color $s-t$ cuts for all non-adjacent $s-t$ pairs and

taking the minimum size cut, and continuing Steps 2 and 3 of Algorithm 2.3. We conclude this section by summarizing the performance of the approximation algorithms.

Theorem 2.4. *Given a colored graph $G(V, E, \mathcal{C})$, let V_i be the set of nodes that have color i. If there are at most q components in the induced graph $G[V_i]$, $\forall i$, then the values of the minimum color $s-t$ node cut and the global minimum color node cut can be approximated to within factor q in $O(|V|^{0.5}|E| + |V|^2)$ and $O(|V|^{2.5}|E|)$ time, respectively. Note that if $q = 1$ the exact solutions are obtained.*

Proof. The fact that the minimum color $s-t$ node cut can be approximated to within factor q follows from Lemma 2.2. The contraction of connected nodes that have the same color takes $O(|V|^2)$ time, by updating the adjacency matrix representation of G. Adding s'' and t'' to G'' takes $O(|V|)$ time, by increasing the numbers of rows and columns of the adjacency matrix by two and adding the new connections. Computing the minimum node $s''t''$ cut in G'' takes $O(|V|^{0.5}|E|)$ time [51]. The total time of approximating the minimum color $s-t$ node cut is $O(|V|^{0.5}|E| + |V|^2)$.

The global minimum color node cut of G is the minimum over 1) C_c^1: the minimum color node $s-t$ cut $\forall st$, and 2) C_c^2: the minimum number of colors that cover at least $n-1$ nodes. Since the value of the minimum color $s-t$ node cut can be approximated to within factor q, the minimum over all non-adjacent $s-t$ pairs, $|C_c^1|$, can also be approximated to within factor q. Moreover, the exact value of $|C_c^2|$ can be obtained in $O(|V|)$ time. Thus, the global minimum color node cut of G can be approximated to within factor q. The number of non-adjacent $s-t$ pairs is at most $|V|^2/2$. The contraction of nodes with the same color can be computed once and reused. Computing the connections between s'', t'' and the contracted nodes takes $O(|V|)$ time for each (s'', t'') pair. Computing the minimum node $s''t''$ cut in G'' takes $O(|V|^{0.5}|E|)$ time for each (s'', t'') pair. Thus, the computation of $|C_c^1|$ requires $O(|V|^2 + |V|^{0.5}|E||V|^2 + |V||V|^2) = O(|V|^{2.5}|E|)$ time.

We remark that although there are faster algorithms to compute the global minimum node cut (*e.g.*, [54]), not all the accelerations can be applied to our problem. For example, computing $(k+1)|V|$ pairs of minimum $s-t$ node cut is enough to

obtain the global minimum node cut in a graph G, where k is the node connectivity of G, because at least one node among $k+1$ nodes does not belong to a minimum cut and can be a source or destination node. However, this does not hold in our problem, where the number of nodes covered by a minimum color node cut can be large, and the $s-t$ node connectivity for $\Theta(|V|^2)$ $s-t$ pairs should be evaluated. □

2.3 Maximizing the supply node connectivity

In this section, we develop supply-demand assignment algorithms to maximize the supply node connectivity of the demand network. Given a fixed demand network topology, the robustness of the demand network depends on the assignment of supply nodes for each demand node. For example, if every node in a cut depends on the same set of supply nodes, then removing these supply nodes could disconnect the demand network. In contrast, if different nodes in every cut depend on different supply nodes, then a larger number of supply nodes should be removed to disconnect the demand network.

For simplicity, in this section, we assume:

1. Every demand node has a fixed number of supply nodes, denoted by n_s.

2. Every supply node can support an arbitrary number of demand nodes.

The total number of supply-demand pairs is $n_1 n_s$, where n_1 is the number of nodes in the demand network G_1. In Section 2.4, we study the case where the number of nodes supported by every supply node is fixed as well, and study the interdependence assignment that maximizes the supply node connectivity of both G_1 and G_2.

The supply-demand assignment problem can be stated as follows in the context of a colored graph. Given a graph $G(V,E)$ and colors C, assign a color $c_i \in C$ to each node, such that the value of the minimum color node cut of G (or the minimum color $s-t$ node cut for $s,t \in V$) is maximized. Graph G is the transformed graph of the demand graph G_1, obtained by Algorithm 2.1, where each node is replicated into n_s nodes.

Under the first assumption, according to Corollary 2.2, the node connectivity of G is $k = k_1 n_s$, where k_1 is the node connectivity of the demand graph G_1. Under any color assignment, the minimum color node cut of G is at most k. Moreover, the minimum color node cut of G is upper bounded by n_c, the total number of available colors (*i.e.*, the total number of supply nodes in G_2). We aim to assign colors to nodes in order for the value of the minimum color node cut to be close to $\min(k, n_c)$. If the value of the minimum color node cut is $\min(k, n_c)/\alpha$ under an assignment algorithm A, then A is an α-approximation algorithm.

2.3.1 Maximizing the $s-t$ supply node connectivity by path-based assignment

We first propose Algorithm 2.4 that maximizes the value of the minimum color $s-t$ node cut, which is simple but provides insight towards maximizing the value of the global minimum color node cut of a graph.

Algorithm 2.4 Path-based Color Assignment.

1. Compute the $s-t$ node connectivity k^{st}. Identify k^{st} node-disjoint $s-t$ paths.
2. Assign the same color to all the nodes in a path. If $n_c \geq k^{st}$, assign a distinct color to each path. If $n_c < k^{st}$, assign a distinct color to each of n_c paths, and assign an arbitrary color to each remaining path.

For the k^{st} node-disjoint $s-t$ paths, any pair of paths do not share the same color if there are sufficient colors ($n_c \geq k^{st}$). Thus, s and t stay connected after removing fewer than k^{st} colors. On the other hand, if $n_c < k^{st}$, there exist n_c paths with distinct colors, and s and t stay connected after removing fewer than n_c colors. To summarize, the performance of Algorithm 2.4 is given by the following theorem.

Theorem 2.5. *The value of the minimum color $s-t$ node cut is* $\min(k^{st}, n_c)$ *if the colors are assigned according to the* Path-based Color Assignment *algorithm, where n_c is the number of colors and k^{st} is the $s-t$ node connectivity.*

It is worth noting that assigning the same color to multiple nodes in a path does not reduce the value of the minimum color $s-t$ node cut, compared with assigning a distinct color to each node. The reason is that, a path is disconnected as long as at least one node in the path is removed. To generalize, if a set of nodes together form a "functional group", it is better for nodes in the same group to share the same risk. In contrast, nodes in different groups should avoid sharing the same risk. We leverage this idea to maximize the global minimum color cut of a graph.

2.3.2 Maximizing the global supply node connectivity by CDS-based assignment

In the remainder of this section, we consider the color assignment that maximizes the global minimum color node cut of a graph. It is helpful to identify the group of nodes that support graph connectivity, analogous to nodes in a path that support $s-t$ connectivity. Indeed, nodes in a *connected dominating set* (CDS) form such a group. A connected dominating set is a set of nodes S such that the induced graph $G[S]$ is connected and that every node in $G(V,E)$ either belongs to S or is adjacent to a node in S. If none of the nodes S are removed, then the graph stays connected regardless of the number of removed nodes in $V \setminus S$. Namely, any subset of nodes $V \setminus S$ is not a node cut of the graph.

The natural analog of node-disjoint $s-t$ paths is (node) disjoint CDS, which support graph connectivity. The failures of nodes in one CDS do not affect another disjoint CDS, while a survived CDS suffices to keep the graph connected. CDS partitions, which partition nodes of $G(V,E)$ into multiple disjoint CDS, have been studied in [41–43]. If the node connectivity of $G(V,E)$ is k and $G(V,E)$ has n nodes, then $\Omega(k/\log^2 n)$ node-disjoint CDS can be obtained in nearly linear time $O(m \text{ polylog } m)$, where m is the number of edges [41, 43].

We propose Algorithm 2.5 that assigns colors based on CDS partitions.

The performance of Algorithm 2.5 can be analyzed in a similar approach to that of Algorithm 2.4. If $n_c \geq k^{\text{CDS}}$, each CDS has a distinct color, and the graph stays

Algorithm 2.5 CDS-based Color Assignment.

1. Compute the node connectivity k of G. Identify $k^{\text{CDS}} = \Omega(k/\log^2 n)$ node-disjoint CDS using the algorithm in [43].
2. Assign the same color to all the nodes in a CDS. If $n_c \geq k^{\text{CDS}}$, assign a distinct color to each CDS. If $n_c < k^{\text{CDS}}$, assign a distinct color to each of n_c CDS, and assign an arbitrary color to each remaining CDS.

connected after removing fewer than k^{CDS} colors. If $n_c < k^{\text{CDS}}$, n_c CDS have distinct colors, and the graph stays connected after removing fewer than n_c colors. Therefore, the value of the minimum color node cut is at least $\min(k^{\text{CDS}}, n_c)$. The performance of Algorithm 2.5 is summarized by the following algorithm.

Theorem 2.6. *The value of the minimum color node cut of G is at least* $\min(\Omega(k/\log^2 n), n_c)$ *if the colors are assigned according to the* CDS-based Color Assignment *algorithm, where n_c is the number of colors, n is the number of nodes, and k is the node connectivity of G. The* CDS-based Color Assignment *algorithm is an $O(\log^2 n)$-approximation algorithm.*

2.3.3 Maximizing the global supply node connectivity by random assignment

Finally, we study a Random Assignment algorithm. The algorithm is to assign each node a color randomly with equal probability. The intuition behind the Random Assignment algorithm is that nodes in a small cut are unlikely to be assigned with the same color if the number of colors is large. Thus, removing the nodes associated with a small number of colors is unlikely to disconnect the graph.

In fact, the Random Assignment algorithm has provably good performance. The analysis relies on the recently studied *vertex sampling* problem in [43]. We first restate a sampling theorem in [43] as follows.

Lemma 2.3 (Theorem 6 in [43]). *Consider a graph G in which each node is removed independently with a given probability $1-p$. For $0 < \delta < 1$, if the probability that a*

node is not removed satisfies $p \geq \beta\sqrt{\log(n/\delta)/k}$ *for a sufficiently large constant* β*, then the remaining graph is connected with probability at least* $1-\delta$*, where* n *is the number of nodes and* k *is the node connectivity of* G*.*

This sampling theorem provides a sufficient condition for a graph to be connected with high probability after its nodes are randomly removed. In particular, we use the following corollary.

Corollary 2.3. *Given a graph* G *with* n *nodes and node connectivity* $k = \omega(\log n)$*, if each node is removed with up to a* constant *probability* $1-p < 1$*, then the remaining nodes in* G *are connected with probability* $1-\delta$ *where* $\delta = O(ne^{-\alpha k})$ *for some constant* α*.*

Proof. Given that the probability p that each node remains in G is at least a constant greater than zero, from Lemma 2.3 we know that the probability δ that G is disconnected satisfies the following equation.

$$k(p/\beta)^2 = \log(n/\delta),$$
$$\delta = ne^{-\alpha k},$$

where $\alpha = (p/\beta)^2$ is a constant.

Moreover, since $k = \omega(\log n)$, $\delta = ne^{-\alpha k} \leq n^{-1} = o(1)$. The probability that the remaining nodes are connected is high. □

On the other hand, if $k = O(\log n)$, $\beta\sqrt{\log(n/\delta)/k} \geq \beta\sqrt{\log(n)/k} = \Omega(1)$. The condition in Lemma 2.3 cannot be satisfied, unless the hidden constant in $k = O(\log n)$ is large. Thus, the probability that the graph is disconnected after randomly removing a given fraction of nodes cannot be bounded using this approach. For simplicity, in the following we focus on graphs where $k = \omega(\log n)$.

In a colored graph G where nodes are randomly colored using a total of n_c colors, removing nodes with colors that belong to a given set of k' colors is equivalent to removing each node with probability k'/n_c. The probability of removing a node is at most a constant, by restricting k' to be at most $(1-\epsilon)n_c$ for a constant $\epsilon > 0$.

Thus, by Corollary 2.3, the probability that G is disconnected after removing nodes with a given set of k' colors is small. By a union bound over $\binom{n_c}{k'}$ combinations of k' colors, the probability p_{union} that G is disconnected after removing nodes with *any* set of k' colors can be bounded. If p_{union} is small, and the remaining nodes form a CDS with high probability (such that removing any subset of nodes with any k' colors does not disconnect G), then the value of the minimum color node cut of G is at least $k'+1$ with high probability. We next fill in the details of the proof, and our approach closely follows the approach of computing node connectivity after random node sampling in [43].

Theorem 2.7. *By assigning a color uniformly at random to each of the n nodes of G, the value of the minimum color node cut of G is $\Theta(\min(k, n_c))$ with high probability, where n_c is the number of colors and $k = \omega(\log n)$ is the node connectivity of G. If, in addition, $k = \omega(n_c)$, then the value of the minimum color node cut of G is at least $(1-\epsilon)n_c$ with high probability for any constant $\epsilon > 0$.*

Proof. We prove the theorem under three cases: i) $k = \Theta(n_c)$; ii) $k = \omega(n_c)$; and iii) $k = o(n_c)$. In all of the three cases, $k = \omega(\log n)$.

i) First we consider the case where $k = \Theta(n_c)$. For $k' \leq (1-\epsilon)n_c$, where $\epsilon > 0$ is a constant, the probability that G is disconnected after removing the nodes covered by a randomly selected set of k' colors is $O(ne^{-\alpha k})$, for a constant α (Corollary 2.3). The total number of k' color combinations among the n_c colors is $\binom{n_c}{k'} \leq (\frac{en_c}{k'})^{k'}$. Thus, by the union bound, the probability that G is disconnected after removing nodes with any k' colors is at most $p_{\text{union-1}} = O(ne^{-\alpha k}(\frac{en_c}{k'})^{k'})$. Let $k' = \alpha \min(k, n_c)/(2\eta) \leq (1-\epsilon)n_c$, where η satisfies $\eta = \log \frac{en_c}{k'} = \log \frac{2\eta e n_c}{\alpha \min(k, n_c)}$ and is a constant.

$$\begin{aligned}
\log p_{\text{union-1}} &\leq \log(ne^{-\alpha k}(\frac{en_c}{k'})^{k'}) \\
&= \log n - \alpha k + k' \log \frac{en_c}{k'} \\
&= \log n - \alpha k + \alpha \min(k, n_c)/2 \\
&\leq \log n - \alpha k/2 \\
&\leq -\gamma \log n,
\end{aligned}$$

for a constant $\gamma > 0$. The last inequality follows from $k = \omega(\log n)$. Therefore, the probability that G is disconnected is at most $n^{-\gamma} = o(1)$.

The above approach proves that with high probability, removing nodes with any k' colors does not disconnect G. Before concluding that the value of the minimum color node cut of G is at least k', we need to prove that removing *any subset* of nodes with any k' colors does not disconnect G (recall Definition 2.3 of a color node cut). A sufficient condition is that the remaining nodes form a dominating set of G.

Since the node connectivity of G is k, the minimum degree of a node in G is at least k. The probability that all the neighbors of a node are removed is $(k'/n_c)^k$. Let $k' \leq (1-\epsilon)n_c$ for a constant $\epsilon > 0$. The probability that there is at least one node whose neighbors are all removed can be upper bounded using the union bound

$$p_{\text{union-2}} = n(k'/n_c)^k \leq n(1-\epsilon)^k = o(1). \tag{2.9}$$

The last inequality follows from $k = \omega(\log n)$. With probability $1 - o(1)$, there does not exist a node whose neighbors are all removed. Thus, the remaining nodes form a dominating set.

To conclude, with probability at least $1 - p_{\text{union-1}} - p_{\text{union-2}} = 1 - o(1)$, the value of the minimum color node cut of G is at least $k' = \Theta(k)$ if $k = \Theta(n_c)$ and $k = \omega(\log n)$.

ii) Next we consider the case where $k = \omega(n_c)$. Let $k' = (1-\epsilon)n_c$.

$$\begin{aligned}
\log p_{\text{union-1}} &\leq \log(ne^{-\alpha k}(\frac{en_c}{k'})^{k'}) \\
&= \log n - \alpha k + k' \log \frac{en_c}{k'} \\
&\leq \log n - \alpha k + 2k' \leq -\gamma \log n,
\end{aligned}$$

for a constant γ. The last inequality holds because $k' = o(k)$ (equivalently, $(1-\epsilon)n_c = o(k)$ and $k = \omega(n_c)$) and $\log n = o(k)$ (equivalently, $k = \omega(\log n)$). The value of the minimum color node cut of G is at least $(1-\epsilon)n_c$ with probability $1 - p_{\text{union-1}} - p_{\text{union-2}} = 1 - o(1)$.

iii) Finally we consider the case where $k = o(n_c)$. Directly using $p_{\text{union-1}}$

would incur an $O(\log n_c)$ gap from the optimal k' (*i.e.*, $k' = \Omega(k/\log n_c)$), because the number of k' out of n_c choices is large and the union bound $p_{\text{union-1}}$ is too weak. However, it is possible to reduce the number of choices, at the cost of removing a larger number of nodes. We use the same approach as in [43]. Partition the colors into $2k' = o(n_c)$ groups. Instead of removing nodes with colors in a selected set of k' colors, we consider removing nodes with colors in a selected set of k' color groups, which consists of around $n_c/2$ colors. The probability that each node is removed is $1/2$. The probability that G become disconnected is still $\delta = O(ne^{-\alpha k})$. The total number of events (*i.e.*, combinations of k' color groups out of $2k'$ color groups) is reduced to $\binom{2k'}{k'} \leq (2e)^{k'}$. For $k' = \alpha k/(2\log(2e))$,

$$\begin{aligned}\log p_{\text{union-3}} &\leq \log(ne^{-\alpha k}(\frac{2ek'}{k'})^{k'}) \\ &= \log n - \alpha k + k' \log(2e) \\ &\leq \log n - \alpha k/2 \leq -\gamma \log n,\end{aligned}$$

for a constant γ.

Thus, the value of the minimum color node cut of G is at least $k' = \alpha k/(2\log(2e)) = \Theta(k)$ with high probability $1 - p_{\text{union-3}} - p_{\text{union-2}} = 1 - o(1)$.

□

Theorem 2.7 proves that the Random Assignment algorithm is an $O(1)$-approximation algorithm if $k = \omega(\log n)$. If $k = O(\log n)$, under any assignment the minimum color node cut value is at least one, and the approximation ratio is at most $O(\log n)$.

2.4 Bidirectional interdependence

In the previous sections, we considered a one-way dependence model. In this section, we extend the results to a *bidirectional interdependence* model. Let $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ denote two interdependent networks. *Interdependence edges* connect nodes between two networks, which represent their supply-demand relationship. The key difference from the one-way dependence model is that the interdependence edges

are *bidirectional* (*i.e.*, if node $v \in G_1$ depends on node $u \in G_2$, then u depends on v as well).

If a node v in G_1 fails due to the failures of its supply nodes in G_2, then the failure of v does not lead to further node failures (due to a lack of supply) in G_2, because all the nodes in G_2 that depend on v have failed. Otherwise, v would not have failed in the first place. Therefore, the evaluation of supply node connectivity in the bidirectional interdependence model follows the same methods as the one-way dependence model. What remains to be developed is the *interdependence assignment* that maximizes the supply node connectivity of *both* networks.

We assume that there are n_{si} interdependence edges adjacent to each of the n_i nodes in G_i ($\forall i = \{1, 2\}$). The total number of bidirectional interdependence edges is $n_1 n_{s1} = n_2 n_{s2}$. Under this assumption, a node in G_i is functional if at least one of its adjacent n_{si} interdependence edges is connected to a remaining node (*i.e.*, a node that has not been removed) in G_j ($\forall i, j = \{1, 2\}, i \neq j$).

We now give an overview of the bidirectional interdependence assignment algorithms. To extend the CDS-based color assignment to interdependence assignment, we aim to avoid disjoint CDS sharing the same supply nodes as much as possible, in both networks. Nodes in G_1 are partitioned into groups of size n_{s2}, and nodes in G_2 are partitioned into groups of size n_{s1}. Interdependence is assigned between each group in G_1 and each group in G_2. Consider a group $P_1 \in G_1$, and a corresponding group $P_2 \in G_2$. Every node $v_1 \in P_1$ depends on all the nodes in P_2, and every node $v_2 \in P_2$ depends on all the nodes in P_1. The key is to partition nodes in G_1 and G_2 into groups. The partition is obvious when the number of nodes in each CDS in G_i is a multiple of n_{sj} ($\forall i, j = \{1, 2\}, i \neq j$), in which case disjoint CDS do not share any supply node. See Fig. 2-4 for an illustration. Otherwise, in general, disjoint CDS may have to share some supply nodes. As we will prove later, the supply node connectivity will be reduced by at most a half, compared with the ideal case where disjoint CDS do not share any supply node. The same analysis applies to the path-based assignment that maximizes the $s - t$ supply node connectivity, and is omitted.

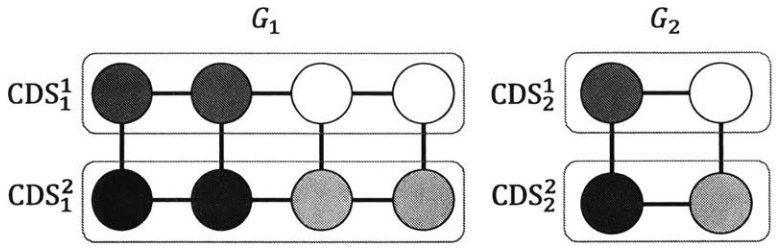


Figure 2-4: An example of the partition of CDS nodes into groups. Every node in G_1 and G_2 has $n_{s1} = 1$ and $n_{s2} = 2$ supply nodes, respectively. Each CDS in G_i is partitioned into two groups of size n_{sj} $(i, j \in \{1, 2\}, i \neq j)$. In each graph, nodes that have the same color are in the same group. Between two graphs, nodes in groups with the same color are interdependent. The partition achieves the optimal supply node connectivity: 2 and 4 for G_1 and G_2, respectively.

2.4.1 CDS-based interdependence assignment

We develop an algorithm to partition the nodes in G_1 into groups of size n_{s2}, and to partition the nodes in G_2 into groups of size n_{s1}. A group of size n_{sj} is *empty* if it contains no node, is *full* if it contains n_{sj} nodes, and is *occupied* if it contains more than zero but fewer than n_{sj} nodes. If $|V_i|/n_{sj}$ is an integer, we aim to partition V_i into $|V_i|/n_{sj}$ full groups. Otherwise, if $|V_i|/n_{sj}$ is not an integer, we aim to partition V_i into $\lfloor |V_i|/n_{sj} \rfloor$ full groups and one occupied group that contains $|V_i^*| = |V_i| - \lfloor |V_i|/n_{sj} \rfloor n_{sj}$ nodes $(\forall i, j \in \{1, 2\}, i \neq j)$, where V_i^* denotes the nodes in the occupied group. Since $|V_1|/n_{s_2} = |V_2|/n_{s_1}$, the total number of groups are the same in both G_1 and G_2.

Interdependence is assigned between nodes in two groups, one from each graph. For each node in $V_i \setminus V_i^*$, there are n_{si} supply nodes. For each node in V_i^*, there are $|V_j^*| < n_{si}$ supply nodes. (Multiple interdependence edges exist between some nodes in V_i^* and some nodes in V_j^*). Given that nodes within a group depend on the same set of supply nodes while different groups of nodes depend on different supply nodes, we aim to partition nodes into groups such that *a large number of groups* need to be removed in order to disconnect all the CDS. Consequently, a large number of supply nodes need to be removed in order to disconnect all the CDS. The partition of V_i into $\lfloor |V_i|/n_{sj} \rfloor$ full groups follows Algorithm 2.6. The remaining nodes (if any) form an occupied group V_i^* if $|V_i|/n_{sj}$ is not an integer.

We denote by h the number of disjoint CDS in G_i. Using the algorithm in [43],

$h = \Omega(k_i/\log^2 n_i)$ disjoint CDS can be computed. If there are extra nodes in V_i that do not belong to the h CDS, then these nodes are added to the largest CDS. Note that adding extra nodes to a CDS still yields a CDS, since these nodes are adjacent to the nodes in the original CDS.

Algorithm 2.6 Assign nodes V_i into $\lfloor |V_i|/n_{sj} \rfloor$ full groups of size n_{sj}.

---

1. Sort the h disjoint CDS in the ascending order of their sizes. Denote the nodes in the l-th CDS in G_i by N^l, $l = 1, 2, \ldots, h$.
2. For l from 1 to h, start with an empty group if available, and assign nodes from N^l into the group. Repeat until all nodes are assigned. If there are not enough empty groups, assign the rest nodes into occupied groups until these groups become full.
3. The algorithm terminates when the $\lfloor |V_i|/n_{sj} \rfloor$ groups become full.

The following example illustrates Step 2 of the algorithm. Before assigning N^l, there are *enough* empty groups if the number of empty groups is at least $\lceil |N^l|/n_{sj} \rceil$. Nodes N^l are assigned to $\lfloor |N^l|/n_{sj} \rfloor$ groups, which then become full. If $|N^l|/n_{sj}$ is not an integer, the remaining $|N^l| - \lfloor |N^l|/n_{sj} \rfloor n_{sj} \leq n_{sj} - 1$ nodes are assigned to another empty group and the group becomes occupied. On the other hand, if there are $n_r < \lceil |N^l|/n_{sj} \rceil$ empty groups before assigning N^l, then $n_r n_{sj}$ nodes in N^l are assigned to the n_r groups. The remaining nodes in N^l and nodes in $N^{l+1}, \ldots, N^h$ are assigned to the already occupied groups.

The algorithm is further illustrated by Fig. 2-5. Suppose that G_1 has 12 nodes, and has three disjoint CDS, consisting of $|N^1| = 2, |N^2| = 4, |N^3| = 6$ nodes, respectively, and that $n_{s2} = 3$. Our goal is to assign the 12 nodes in G_1 to 4 groups of size 3. Before assigning nodes in N^1, all the four groups are empty. Thus, the two nodes in N^1 can be assigned to an empty group. After the assignment, the group becomes occupied, illustrated by the left figure in Fig. 2-5. Before assigning nodes in N^2, there are three empty groups. The assignment of N^2 uses two groups, one of which becomes full and the other becomes occupied (groups 2 and 3 in Fig. 2-5). Finally, when assigning N^3, there is only one empty group, and thus there are not enough

empty groups to hold all the nodes in N^3. The last empty group can be assigned with 3 nodes. The remaining 3 nodes in N^3 are assigned to the occupied groups (*i.e.*, groups 1 and 3 in Fig. 2-5).

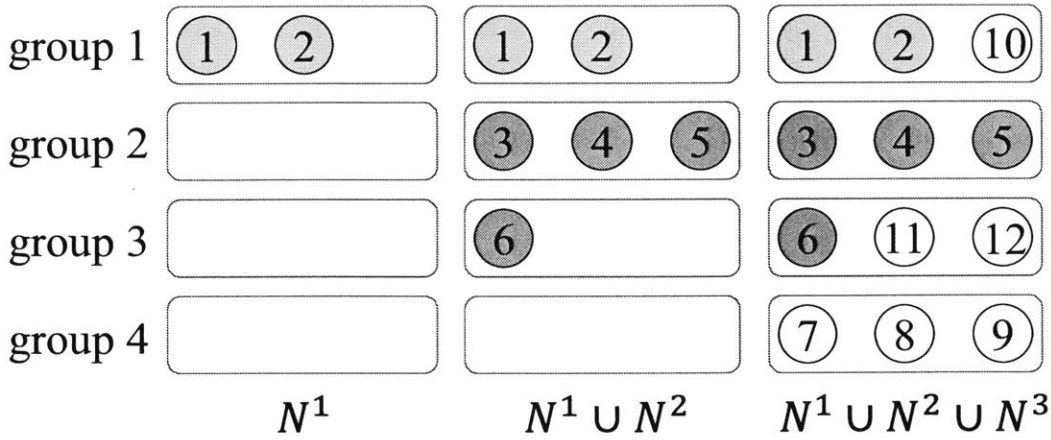


Figure 2-5: Partition the CDS nodes $\{N^1, N^2, N^3\}$ into four groups of size three. The left, middle, right figures represent the snapshots after assigning nodes in N^1, N^2, N^3 in Step 2 of Algorithm 2.6, respectively.

We prove that disjoint CDS are sufficiently group-disjoint, by characterizing the number of groups that need to be removed to disconnect all the CDS.

Lemma 2.4. *Let V_i be assigned to groups according to Algorithm 2.6. The minimum number of full groups that need to be removed, in order for each CDS to contain at least one removed node, is at least* $\min(\lceil (h-1)/2 \rceil, \lfloor |V_i|/n_{sj} \rfloor)$.

Proof. Let $|N^l|$ denote the number of nodes in the l-th CDS of G_i, $\forall l \in \{1, \ldots, h\}$. If $|N^l|$ is a multiple of n_{sj}, $\forall l \in \{1, \ldots, h\}$, then nodes in N^{l_1} are assigned to different groups from nodes in N^{l_2}, $\forall l_1, l_2 \in \{1, \ldots, h\}, l_1 \neq l_2$. To remove at least one node from each of the CDS, h full groups need to be removed. In the rest of the proof, we focus on the case where $|N^l|$ is not a multiple of n_{sj} for some $l \in \{1, \ldots, h\}$.

In the first few assignments in Algorithm 2.6 when there are enough empty groups, nodes in N^{l_1} are assigned to different groups from nodes in N^{l_2}, $\forall l_1, l_2 \in \{1, \ldots, k_{\text{th}}\}, l_1 \neq l_2$. In order to disconnect all the CDS, at least one node should be removed from each CDS. The removed nodes in CDS $N^l, l = 1, \ldots, k_{\text{th}}$ belong to at least k_{th} distinct groups. Therefore, at least k_{th} groups need to be removed in order to

disconnect all the CDS. (Note that these groups become full by the end of Algorithm 2.6.)

Determining k_{th}: Consider one CDS N^l ($l \in \{1, \ldots, k_{\text{th}}\}$). If $|N^l|/n_{sj}$ is not an integer, one group occupied by N^l is not full, and the group can still be assigned with $r^l \leq n_{sj} - 1$ extra nodes. If $|N^l|/n_{sj}$ is an integer, then $r^l = 0$. The total number of extra nodes that can be assigned into these occupied groups is $\sum_{l=1}^{k_{\text{th}}} r^l \leq k_{\text{th}}(n_{sj} - 1)$.

Consider the assignment when there are not enough empty groups to hold all the nodes in $N^l, \forall l = k_{\text{th}} + 1, \ldots, h$.

1) If $|N^{k_{\text{th}}+1}| \leq n_{sj}$, then $|N^l| \leq n_{sj}, l = 1, \ldots, k_{\text{th}}$. (Recall that the CDS are sorted in the ascending order of their sizes.) Nodes in each CDS N^l belong to a single occupied group, $l = 1, \ldots, k_{\text{th}}$. Moreover, since there is no empty group available when assigning nodes in $N^{k_{\text{th}}+1}$, all the empty groups have been used, and $k_{\text{th}} = \lfloor |V_i|/n_{sj} \rfloor$.

2) If $|N^{k_{\text{th}}+1}| \geq n_{sj} + 1$, the number of remaining CDS is at most

$$\begin{aligned} h - k_{\text{th}} &\leq \lfloor \frac{\sum_{l=1}^{k_{\text{th}}} r^l + n_{sj} - 1}{n_{sj} + 1} \rfloor + 1 \\ &\leq \lfloor \frac{(k_{\text{th}} + 1)(n_{sj} - 1)}{n_{sj} + 1} \rfloor + 1 \leq k_{\text{th}} + 1. \end{aligned}$$

To see this, note that there is no empty group available when assigning nodes in $\cup_{l=k_{\text{th}}+2}^{h} N^l$. Otherwise, all the nodes in $N^{k_{\text{th}}+1}$ would have been assigned to empty groups, which contradicts the assumption. Let $N^o \subseteq \cup_{l=k_{\text{th}}+2}^{h} N^l$ denote the nodes that will be assigned to the occupied groups (occupied by nodes in $N^l, l \in \{1, \ldots, k_{\text{th}}\}$). Let $N^* \subseteq \cup_{l=k_{\text{th}}+2}^{h} N^l$ denote the remaining nodes that cannot be assigned to the $\lfloor |V_i|/n_{sj} \rfloor$ groups when $|V_i|/n_{sj}$ is not an integer. By definition, $N^o \cup N^* = \cup_{l=k_{\text{th}}+2}^{h} N^l$. We know that $|N^o|$ is at most $\sum_{l=1}^{k_{\text{th}}} r^l$, which is the number of extra nodes that the occupied groups can fit. Moreover, $|N^*|$ is at most $|V_i| - \lfloor |V_i|/n_{sj} \rfloor n_{sj} \leq n_{sj} - 1$. Therefore, $\sum_{l=1}^{k_{\text{th}}} r^l + n_{sj} - 1$ is an upper bound on the number of nodes in $\cup_{l=k_{\text{th}}+2}^{h} N^l$. Since the size of N^l ($k_{\text{th}}+2 \leq l \leq h$) is at least $n_{sj}+1$, the first term in the summation is an upper bound on the number of CDS $N^{k_{\text{th}}+2}, \ldots, N^h$. The additional one (second term in the summation) accounts for the CDS $N^{k_{\text{th}}+1}$.

In summary, given that the total number of CDS $h = k_{\text{th}} + (h - k_{\text{th}}) \leq k_{\text{th}} + (k_{\text{th}} + 1)$,

we obtain $k_{\text{th}} \geq (h-1)/2$. Since k_{th} is an integer, k_{th} is at least $\lceil (h-1)/2 \rceil$. □

Given that n_{si} supply nodes need to be removed in order to remove a full group of nodes of V_i, we have the following result.

Theorem 2.8. *Given G_i with n_i nodes and node connectivity k_i, and that every node has n_{si} supply nodes, $\forall i \in \{1,2\}$, assign interdependence between nodes in G_1 and the nodes in G_2 by groups, obtained in Algorithm 2.6. Then, the supply node connectivity of G_i is $\Omega(\min(k_i n_{si}/\log^2 n_i, n_j))$, $\forall i,j \in \{1,2\}, i \neq j$.*

Proof. Using the algorithm in [43], $h = \Omega(k_i/\log^2 n_i)$ disjoint CDS can be found in G_i. By Lemma 2.4, the number of full groups that should be removed in order to remove at least one node from each CDS is $\min(\lceil (h-1)/2 \rceil, \lfloor n_i/n_{sj} \rfloor)$, $\forall i,j \in \{1,2\}, i \neq j$.

Each group of V_i can be removed by removing n_{si} supply nodes in G_j. Noting that $h = \Omega(k_i/\log^2 n_i)$ and that $n_i n_{si}/n_{sj} = n_j$, the supply node connectivity of G_i is $\Omega(\min(k_i n_{si}/\log^2 n_i, n_j))$, $\forall i,j \in \{1,2\}, i \neq j$. □

We have proved that the CDS-based interdependence assignment algorithm is an $O(\log^2 n_i)$-approximation algorithm in maximizing the supply node connectivity of G_i, $\forall i \in \{1,2\}$.

2.4.2 Random interdependence assignment

We study the random assignment in order to maximize the supply node connectivity of both graphs. The random assignment algorithm is to randomly match n_{s1} copies of nodes in G_1 with n_{s2} copies of nodes in G_2, and assign interdependence between matched nodes. Under the assignment, each of the n_i nodes in G_i is supported by n_{si} nodes in G_j $(i,j \in \{1,2\}, i \neq j)$.

The key difference of the analysis from the random assignment algorithm for the one-way dependence model is as follows. By randomly removing k' nodes in G_2, $k' n_{s2}$ nodes in the transformed graph of G_1 (by Algorithm 2.1) are removed. In contrast, in the one-way dependence model (Section 2.3.3), every node is removed with probability $k' n_{s2}/n_1 n_{s1}$, and the total number of node removals follows a binomial distribution

with mean $k'n_{s2}$. We derive the following lemma that bounds the probability of a graph being disconnected after a constant fraction of nodes are removed, instead of each node being removed with a constant probability as in Corollary 2.3.

Lemma 2.5. *Given graph G with n nodes and node connectivity $k = \omega(\log n)$, after randomly removing up to a* constant *(less than one) fraction of n nodes, the remaining nodes in G are connected with probability $1-\delta$ where $\delta = O(ne^{-\alpha' k})$ for some constant α'.*

Proof. We prove a stronger result that the remaining nodes form a connected dominating set (CDS) with high probability. In particular, we prove for the case where $(1-\epsilon)(1-p)n$ nodes are randomly removed, for a constant $\epsilon < 1$ and a constant $p \in (0,1)$.

Let $A(n_{\text{rm}})$ denote the event that the remaining nodes in G form a CDS after randomly removing n_{rm} nodes, where n_{rm} is a deterministic value. Since adding extra nodes to a CDS still yields a CDS, $\Pr(A(n_{\text{rm}}))$ is decreasing in n_{rm}.

Consider the case where each node is randomly removed with probability $1-p \in (0,1)$. The number of removed nodes, N_{rm}, follows a binomial distribution with mean $(1-p)n$. Using the Chernoff bound, for a constant $\epsilon < 1$,

$$\Pr(N_{\text{rm}} < (1-\epsilon)(1-p)n) \leq e^{-(1-p)n\epsilon^2/2}.$$

The probability that the remaining nodes in G form a CDS after removing N_{rm} nodes is:

$$\Pr(A(N_{\text{rm}})) = \sum_{n_{\text{rm}}=0}^{n} \Pr(A(n_{\text{rm}}))\Pr(N_{\text{rm}} = n_{\text{rm}}) \tag{2.10}$$

$$\leq \Pr(A((1-\epsilon)(1-p)n))\Pr(N_{\text{rm}} \geq (1-\epsilon)(1-p)n) + 1\Pr(N_{\text{rm}} < (1-\epsilon)(1-p)n) \tag{2.11}$$

$$\leq \Pr(A((1-\epsilon)(1-p)n))(1-e^{-(1-p)n\epsilon^2/2}) + e^{-(1-p)n\epsilon^2/2}, \tag{2.12}$$

where Eq. (3.15) follows from the law of total probability, Eq. (2.11) follows from

that $\Pr(A(n_{\rm rm}))$ is non-increasing in $n_{\rm rm}$, and Eq. (2.12) follows from the Chernoff bound. Thus,

$$\Pr(A((1-\epsilon)(1-p)n)) \geq \frac{\Pr(A(N_{\rm rm})) - e^{-(1-p)n\epsilon^2/2}}{1 - e^{-(1-p)n\epsilon^2/2}}.$$

From the proof of Corollary 2.3, we know that by removing $N_{\rm rm}$ nodes, G is disconnected with probability at most $ne^{-\alpha k}$, where α is a constant. Moreover, let $k'/n_c = 1-p$ in Eq. (2.9), the probability that the remaining nodes in G do not form a dominating set is at most $n(1-p)^k$. Thus, by the union bound, the probability that the remaining nodes in G do not form a connected dominating set is at most $1 - \Pr(A(N_{\rm rm})) \leq ne^{-\alpha k} + n(1-p)^k$.

We now bound the probability that the remaining nodes in G form a CDS, after randomly removing $p'n$ nodes, where $p' = (1-\epsilon)(1-p)$ is a constant.

$$\begin{aligned}
\Pr(A((1-\epsilon)(1-p)n)) &\geq \frac{\Pr(A(N_{\rm rm})) - e^{-(1-p)n\epsilon^2/2}}{1 - e^{-(1-p)n\epsilon^2/2}} \\
&\geq \Pr(A(N_{\rm rm})) - e^{-(1-p)n\epsilon^2/2} \\
&\geq 1 - ne^{-\alpha k} - n(1-p)^k - e^{-(1-p)n\epsilon^2/2}
\end{aligned}$$

Let $\alpha' = \min(\alpha, -\log(1-p), (1-p)n\epsilon^2/2k)$. Then $ne^{-\alpha k}, n(1-p)^k, e^{-(1-p)n\epsilon^2/2} \leq ne^{-\alpha' k}$. Therefore, $\Pr(A(p'n)) \geq 1 - O(ne^{-\alpha' k})$. Moreover, since α, p are constants and $n = \Omega(k)$, α' is a constant.

□

Then, following the analysis in Theorem 2.7, and noting Corollary 2.2, we obtain the following result.

Theorem 2.9. *Given G_i with n_i nodes and node connectivity k_i, if each node in G_i has n_{si} supply nodes, by randomly matching n_{si} copies of nodes in G_i to n_{sj} copies of nodes in G_j, and assigning interdependence between each pair of matched nodes, then the supply node connectivity of G_i is $\Theta(\min(k_i n_{si}, n_j))$ with high probability, if $k_i n_{si} = \omega(\log(n_i n_{si}))$, $\forall i, j \in \{1,2\}, i \neq j$. If, in addition, $k_i n_{si} = \omega(n_j)$, then*

the supply node connectivity of G_i is at least $(1-\epsilon)n_j$ with high probability for any constant $\epsilon > 0$.

Proof. By Corollary 2.2, the transformed graph (by Algorithm 2.1) $\tilde{G}_i$ has $n_i n_{si}$ nodes and node connectivity $k_i n_{si}$, $\forall i \in \{1,2\}$. The number of colors in G_i is the number of nodes n_j in G_j, $\forall i,j \in \{1,2\}, i \neq j$. Given Lemma 2.5, the supply node connectivity of G_i can be computed in the same approach as the proof for Theorem 2.7. □

Thus, the random assignment is an $O(1)$-approximation algorithm in maximizing the supply node connectivity of both G_1 and G_2, if $k_i n_{si} = \omega(\log(n_i n_{si}))$, $\forall i \in \{1,2\}$. If $k_i n_{si} = O(\log(n_i n_{si}))$, the approximation ratio is at most $O(\log(n_i n_{si}))$, since the supply node connectivity is at least one under any assignment, $\forall i \in \{1,2\}$.

2.5 Numerical results

In this section, we apply the algorithms in the previous sections and provide numerical results. We use MATLAB to generate network topologies and dependence assignment, and use JuMP [55] to compute the supply node connectivity by calling CPLEX to solve the integer programs in a workstation that has an Intel Xeon Processor (E5-2687W v3) and 64GB RAM.

The key observations are as follows. First, the supply node connectivity for a network of reasonable size can be computed using the integer program in a short time. For example, the results can be obtained within one minute, for a network that has around 180 nodes and 650 edges. Second, the assignment algorithms have good performance even when the value of supply node connectivity is moderate. This complements the theoretical results that the assignment algorithms are optimal up to a constant or polylogarithmic factor. The numerical results therefore suggest that the algorithms are practical in the design of interdependent networks.

2.5.1 $s-t$ supply node connectivity

We use the XO communication network [56] of 60 nodes as an example of the demand network, and randomly generate 36 supply nodes (marked as triangles in Fig. 3-2) within the continental US.

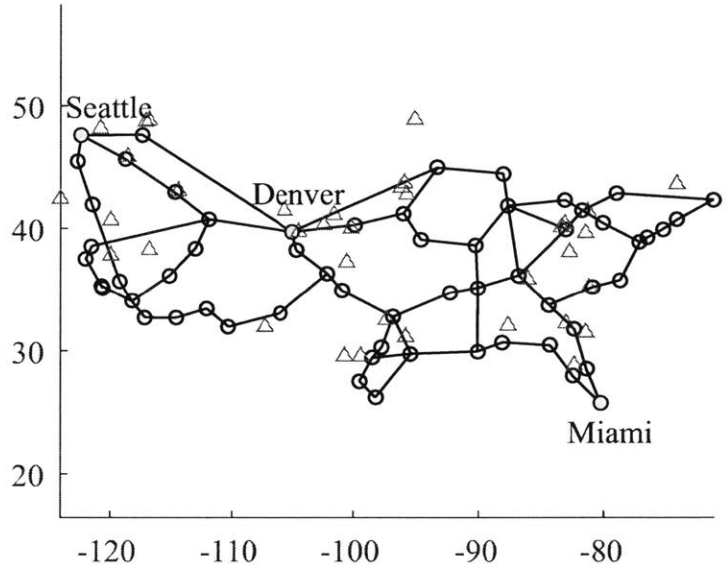


Figure 2-6: XO network as a demand network, with randomly generated supply nodes. The x-axis represents longitude degrees (west), and the y-axis represents latitude degrees (north).

Let each node in the XO network be supported by three nearest supply nodes. After transforming the network into a colored graph by Algorithm 2.1 and solving the MILP, we obtain that the supply node connectivity of the $s-t$ pair Seattle-Denver is 5. In contrast, the maximum $s-t$ supply node connectivity is 9, by assigning distinct supply nodes to each of the three node-disjoint paths (*i.e.*the path-based assignment outlined in Algorithm 2.4). As another example, the supply node connectivity of the $s-t$ pair Seattle-Miami is only 3, because one node in an $s-t$ path has the same set of three supply nodes as another node in a disjoint $s-t$ path. By assigning distinct supply nodes to two disjoint paths (Algorithm 2.4), the supply node connectivity of Seattle-Miami can be increased to 6.

2.5.2 Global supply node connectivity

If each node in the XO network is supported by its three nearest supply nodes, the global supply node connectivity is 3. In contrast, if each node is supported by three randomly chosen supply nodes, the global supply node connectivity can be increased to 5. It is close to the maximum possible global supply node connectivity 6, given that the node connectivity of the XO network is two and each node has three supply nodes. However, the CDS-based assignment (Algorithm 2.5) only guarantees that the supply node connectivity is at least 3, since there do not exist two disjoint CDS in the XO network.

2.5.3 Bidirectional interdependence assignment

We implement the bidirectional interdependence assignment algorithms on randomly generated Erdos-Renyi graphs. Let G_i be an Erdos-Renyi graph with n_i nodes. Let the probability that an edge exists between any two nodes be p_i. Each node in G_i has n_{si} supply nodes from G_j. Let k_i denote the node connectivity of G_i. Recall that the maximum supply node connectivity is $k^s_{i\max} = \min(k_i n_{si}, n_j)$ $(i, j \in \{1, 2\}, i \neq j)$. Table 2.1 depicts the supply node connectivity k^s_i of G_i under the CDS-based and random interdependence assignment algorithms. To obtain the numerical results for CDS-based interdependence assignment algorithm, instead of using the CDS partition algorithm in [43], we use a greedy approach to compute the disjoint CDS, which has good performance for Erdos-Renyi graphs. The results are averaged over 10 instances for each of the two combinations of interdependent networks: 1) $n_1 = 50$, $n_2 = 75$, $p_1 = p_2 = 0.1$; 2) $n_1 = 50$, $n_2 = 75$, $p_1 = p_2 = 0.2$. From the results, we observe that the (near-linear time) CDS-based and the (linear time) random interdependence assignment algorithms yields near-optimal supply node connectivity in both graphs.

Table 2.1: Supply node connectivity k_i^s of random graphs under CDS-based and random assignments.

n_1	p_1	k_1	n_{s1}	$k^s_{1\max}$	k_1^s CDS	k_1^s random
50	0.1	1.6	3	4.8	4.8	4.7
50	0.2	3.6	3	10.8	10.2	10.0
n_2	p_2	k_2	n_{s2}	$k^s_{2\max}$	k_2^s CDS	k_2^s random
75	0.1	2.4	2	4.8	4.6	4.6
75	0.2	7.0	2	14.0	12.4	12.2

2.6 Summary

We studied the robustness of interdependent networks based on a finite-size, arbitrary-topology graph model. We defined supply node connectivity as a robustness metric, by generalizing the node connectivity in a single network. We developed integer programs to compute the supply node connectivity both for an $s-t$ pair and for a network, and developed approximation algorithms for faster computation. Moreover, we develop interdependence assignment algorithms to design robust interdependent networks.

Our study extends the shared risk group model, by considering that multiple risks together lead to the failure of a node. The color assignment algorithms in Section 2.3 can be used as solutions to the less intensively studied design problems for the shared risk group model, to maximize the number of risks that a network can tolerate.

2.7 Chapter appendix

Proof of Theorem 2.2. The minimum color node cut problem can be reduced from the *vertex cover* problem. Given a graph $G'(V', E')$, the minimum vertex cover problem aims to select the minimum number of nodes $V^* \subseteq V'$ such that every edge in E' is incident to at least one node in V^*.

We construct a colored graph G in which the value of the minimum color node cut equals the size the the minimum vertex cover in G'. Let m' denote the number of edges in G'. Without loss of generality we assume that m' is even. (Otherwise, one edge can be added parallel to any existing edge, which does not change the size of

the minimum vertex cover.) Graph G consists of four cliques of size m' each. Nodes in every clique are divided into two disjoint sets of size $m'/2$. Four cliques are joined into a ring, by matching two disjoint set of $m'/2$ nodes of a clique to $m'/2$ nodes in each of the two adjacent cliques (see Fig. 2-7).

Then, assign colors to nodes in G. Consider two matchings M_1 and M_2 that connect two pairs of cliques. There are m' edges in the union of the two matchings. Each edge (v_1, v_2) in this union corresponds to an edge (v_1', v_2') in G'. Let each node in G' have a distinct color, and assign v_1 (v_2) the same color as v_1' (v_2'). Finally, assign each remaining node in G a distinct color, and these remaining nodes are not adjacent to matchings edges M_1 or M_2. See Fig. 2-7 for an example of G, where $m' = 8$ and the number on each node represents its color, and the corresponding G' represented by Fig. 2-8.

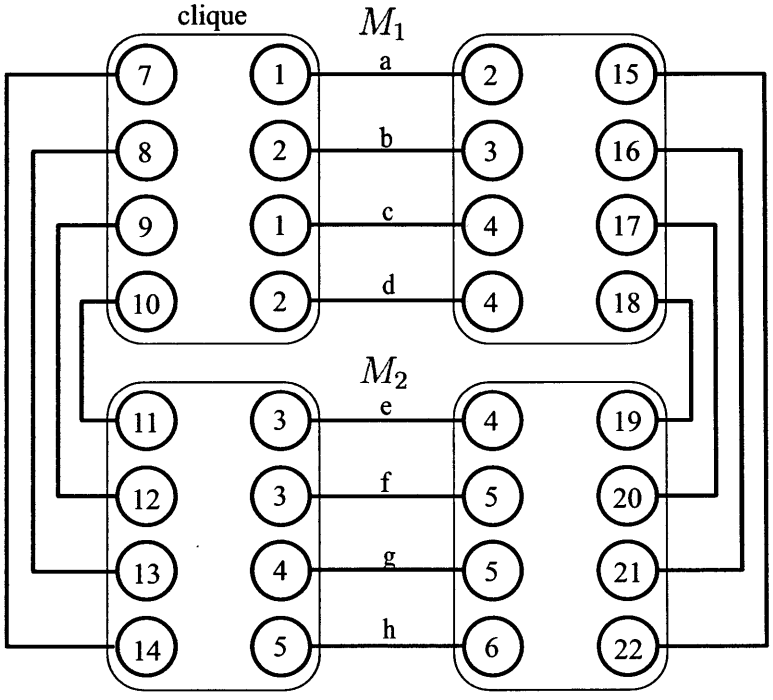


Figure 2-7: In a colored graph G where the number on each node represents its color, the minimum color node cut is $\{2, 4, 5\}$.

By removing at least one node incident to each of the m' matching edges, G becomes disconnected. In particular, the minimum color node cut of G consists of

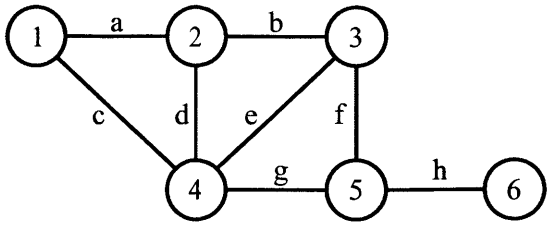


Figure 2-8: The minimum vertex cover in G' is $\{2, 4, 5\}$.

a set of colors C_c such that all the matching edges M_1 and M_2 are incident to at least one node that has a color in C_c. The nodes in G' that have colors C_c form the minimum vertex cover in G', since every edge in G' is adjacent to at least one node that has a color in C_c. (Note that the minimum color node cut of G has size smaller than m', because the number of nodes in a cut of G is m' and some nodes have the same color. Therefore, colors of nodes incident to the other two unlabeled matchings in Fig. 2-7 cannot be in the minimum color node cut.)

Finally, to see that the reduction can be done in polynomial time, note that G has $4m'$ nodes, $2m'^2$ edges, and $n' + 2m'$ colors, where n' and m' are the number of nodes and edges in G', respectively. This concludes the proof. □

Proof of Theorem 2.3. The minimum color $s - t$ node cut problem can be reduced from the *hitting set* problem. Given a universe U of elements, sets S_i consisting of elements in U $(i = 1, 2, \ldots, p)$, the minimum hitting set problem aims to select a minimum number of elements from U such that each set S_i contains at least one selected element.

We construct a colored graph in which the minimum color st node cut is identical to the minimum hitting set. Construct p node-disjoint paths between an st pair, each of which corresponds to a set S_i. If S_i has j elements, then its corresponding path has j nodes with colors that represent the elements in S_i. Nodes that correspond to the same element have the same color. The reduction can clearly be done in polynomial time. A minimum color st node cut contains a set of colors C_c^{st} such that every path has at least one node with a color in C_c^{st}. This is exactly the minimum set of elements such that every set contains at least one such element.

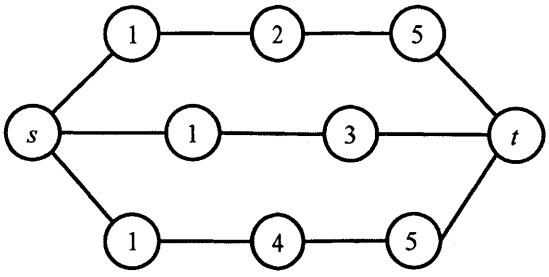


Figure 2-9: The minimum color $s-t$ node cut is $\{1\}$.

We illustrate the reduction by the following example. Consider a hitting set problem where $U = \{1,2,3,4,5\}$, $S_1 = \{1,2,5\}, S_2 = \{1,3\}, S_3 = \{1,4,5\}$. A minimum hitting set is $\{1\}$. The equivalent minimum color st node cut problem is represented by Fig. 2-9. □

Chapter 3

Robust routing in interdependent networks

In this chapter, we study robust routing problems in interdependent networks. For an overview of the problems and challenges, it is helpful to consider a simplified scenario where a demand network depends on a supply network, illustrated by Fig. 3-1. Every node in the demand network is supported by one or more nodes in the supply network. Thus, nodes in the demand network and nodes in the supply network can be viewed as demand nodes and supply nodes, respectively. Given that a demand node fails if it loses all of its supply nodes, supply node failures may lead to correlated demand node failures, which makes it difficult to route traffic through reliable paths in the demand network. We develop techniques to tackle the failure correlation. This simplified one-way dependence exists in current systems. Moreover, the analysis based on this simplified scenario can be applied to interdependent networks.

In this chapter, we develop an analytically tractable framework to study the following robust routing problems in interdependent networks.

`Single-path routing`: Compute the probability that a specified path fails. Obtain the most reliable path between a source-destination pair.

`Diverse routing`: Compute the probability that two specified paths both fail. Obtain the pair of most reliable paths between a source-destination pair.

The rest of the chapter is organized as follows. In Section 3.1, we state our model

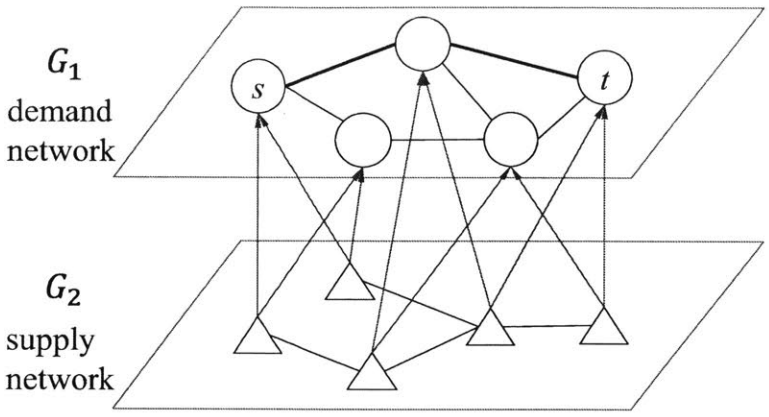


Figure 3-1: Every node in the demand network G_1 is supported by two nodes in the supply network G_2.

for interdependent networks and failures. In Section 3.2, we prove the complexity, and develop approximation algorithms to compute the path failure probability. In Section 3.3, we develop algorithms to find the most reliable path between a pair of nodes. In Section 3.4, we study the diverse routing problem in interdependent networks, and find a pair of reliable paths whose failure probability is minimized. Section 3.5 provides numerical results. Finally, Section 3.6 concludes the chapter.

3.1 Model

We consider a demand network G_1 and a supply network G_2, where every demand node in G_1 depends on one or more supply nodes in G_2. We assume that every supply node provides substitutional supply to the demand nodes, and a demand node is functioning if it is directly connected to at least one supply node. To study the impact of node failures in G_2 on G_1, it is equivalent to study the following model.

Consider a graph $G(V, E, \mathcal{S}_V)$, where nodes V and edges E are identical to nodes and edges in G_1, and $\mathcal{S}_V$ are the supply node sets, each of which is a set of nodes in G_2 that provide supply to a node in V. In this model, each node $v_i \in V$ is a demand node, supported by a set of supply nodes $S_i \in \mathcal{S}_V$, and v_i fails if all the nodes in S_i fail. (Note that nodes V may have different number of supply nodes.) Finally, let $s, t \in V$ be a source-destination pair.

Under the condition that supply nodes fail independently with given probabilities, and following the convention that s, t do not fail, we study the robust routing problems in $G(V, E, \mathcal{S}_V)$.

Remark. The analysis for this model can be directly applied to interdependent networks, as long as the interdependence is bidirectional (*i.e.*, if $v \in G_1$ depends on $u \in G_2$, then u depends on v as well) and failures initially occur in one network. It suffices to observe that, given a set of failed nodes $S \subseteq G_2$, a node $v \in G_1$ fails if and only if its supply nodes are all in S. Notice that the failure of v does not further lead to node failures in G_2, because all the nodes that v supports, which are exactly the supply nodes for v due to the bidirectional interdependence, have failed.

3.2 Computing the reliability of a path

If every node has a single supply node, the path failure probability is given by $1 - (1 - p)^r$, where each supply node fails independently with probability p and the path is supported by r supply nodes. In contrast, if every node has more than one supply node, computing the path failure probability becomes $\#P$-hard. The proof can be found in the Appendix.

Theorem 3.1. *Computing the failure probability of a path is $\#P$-hard, if every node has two or more supply nodes and each supply node fails independently with probability p.*

Although it is $\#P$-hard to compute, the path failure probability can be well approximated. We apply the solution to the *DNF probability* problem and propose an (ϵ, δ)-approximation algorithm based on importance sampling, which approximates the path failure probability to within a multiplicative factor $1 \pm \epsilon$ with probability at least $1 - \delta$.

The DNF probability problem computes the probability that a Disjunctive Normal Form (DNF) formula is true, when literals are set to be true independently with given probabilities. A DNF formula is a disjunction of clauses, each of which is a conjunction

of literals, and takes the following form: $(x_1^1 \wedge \cdots \wedge x_{n_1}^1) \vee (x_1^2 \wedge \cdots \wedge x_{n_2}^2) \vee \cdots \vee (x_1^m \wedge \cdots \wedge x_{n_m}^m)$. Let $v_1 - \cdots - v_m$ be a path in $G(V, E, \mathcal{S}_V)$. The key observation is that computing the path failure probability can be formulated by a DNF probability problem, in which a clause C_i represents a node v_i in the path and the literals x_j^i in clause C_i represent the supply nodes of v_i. For completeness, we state Algorithm 3.1 that approximates the path failure probability, by adapting the algorithm that approximates the DNF probability in [57].

Algorithm 3.1 Estimating the path failure probability based on importance sampling.

Initialization:

1. Given a path $\{v_1, v_2, \ldots, v_m\}$, let $\{u_j^i | j = 1, \ldots, n_s(v_i)\}$ denote the set of supply nodes of v_i, where $n_s(v_i)$ is the number of supply nodes of v_i.

Main loop:

2. Among $\{v_1, v_2, \ldots, v_m\}$, randomly choose v_i with probability

$$\prod_{1 \le j \le n_s(v_i)} p(u_j^i) / \sum_{1 \le k \le m} \prod_{1 \le j \le n_s(v_k)} p(u_j^k).$$

   If every demand node has an identical number of supply nodes, and the supply node failure probability $p(u_j^i)$ is identical, then node v_i is chosen with probability $1/m$.

3. If v_i is chosen, set all of its supply nodes $\{u_j^i | j = 1, \ldots, n_s(v_i)\}$ to be failed. The other supply nodes are randomly set to be failed with their respective failure probabilities. Let U denote the set of failed supply nodes.

4. Test whether v_i is the first failed node among $\{v_1, v_2, \ldots, v_m\}$, given that U fail (and no other supply nodes fail). If true, set $I = 1$; otherwise, set $I = 0$. Repeat the loop for $a = 3m \ln(2/\delta)/\epsilon^2$ iterations.

Result:

5. Count the number of $I = 1$ and denote the number by b. An (ϵ, δ)-approximation of the path failure probability is given by $b/a \sum_{1 \le k \le m} \prod_{1 \le j \le n_s(v_k)} p(u_j^k)$.

The intuition behind this importance sampling algorithm is as follows. Some events, although rare, are important in determining the path failure probability, es-

pecially when the path failure probability is small. The algorithm samples in a space consisting of important events, each of which is a set of supply node failures U that lead to the path failure. In this space, the failure of U may appear multiple times, given that multiple choices of v_i in Step 2 may lead to the same U in Step 3. The algorithm then remove the duplicated U via sampling in Step 4.

To prove the correctness of the algorithm, we take the following two steps. First, following a similar analysis to [57], we prove that the path failure probability is given by $E[I] \sum_{1 \le k \le m} \prod_{1 \le j \le n_s(v_k)} p(u_j^k)$, where $E[I]$ is the expectation of I in Step 4 of the algorithm. Second, by repeating the loop a sufficiently large number of times, $E[I]$ can be approximated to within factor $1 \pm \epsilon$ with probability at least $1 - \delta$. The details of the proof can be found in the Appendix.

The advantage of this algorithm over a naïve Monte-Carlo algorithm (*e.g.*, by repeatedly simulating the supply node failure events and counting the fraction of trials in which the path fails) is that the number of iterations in the naïve Monte-Carlo algorithm is large when the path failure probability is small[1]. In contrast, by sampling in a more important space, the number of iterations is reduced. Note that the only quantity that needs to be estimated in Algorithm 3.1 by simulation is $E[I]$, and that $\Pr(I = 1) \ge 1/m$. We conclude this section by the following theorem, whose proof is in the Appendix.

Theorem 3.2. *The path failure probability can be estimated to within a multiplicative factor $1 \pm \epsilon$ with probability $1 - \delta$, in time $O(m^2 n_s \ln(1/\delta)/\epsilon^2)$, where m is the path length and n_s is the maximum number of supply nodes for a demand node.*

Although the failure probability of a specific path can be well approximated by the importance sampling algorithm, the algorithm hardly gives an intuition for path properties that characterize a reliable path. In the remainder of this section, we develop indicators and bounds on the path failure probability, which can be used for finding the most reliable path.

[1]If F occurs in b out of a trials, $\Pr(F) \in (1 \pm \epsilon)b/a$ with probability $1 - \delta$, under the condition that $b = \Omega(\ln(1/\delta)/\epsilon^2)$. The total number of trials $a = \Omega(\ln(1/\delta)/\epsilon^2)/\Pr(F)$ is large when $\Pr(F)$ is small.

3.2.1 Small and identical failure probability

Consider a path $v_1 - \cdots - v_m$ in $G(V, E, \mathcal{S}_V)$. Let F_i denote the event that all the supply nodes of v_i fail. Let F denote the event that the path fails. Clearly, the path fails if at least one node v_i loses all of its supply nodes ($F = \cup_{1 \le i \le m} F_i$).

By the inclusion-exclusion principle, we have

$$\Pr(F) = \sum_{1 \le i \le m} \Pr(F_i) - \sum_{1 \le i_1 < i_2 \le m} \Pr(F_{i_1} \cap F_{i_2}) + \cdots + (-1)^{m-1} \Pr(F_1 \cap F_2 \cdots \cap F_m). \quad (3.1)$$

Directly computing the path failure probability is difficult, given that there are $\binom{m}{j}$ summations in the j-th term of the inclusion-exclusion formula. We first reduce the number of events in the inclusion-exclusion formula, and then further simplify the computation under the condition that the supply node failure probability is small and identical.

To reduce the number of events, some *redundant* events can be ignored. For example, if F_i occurs only if F_j occurs, then the event F_i is redundant in determining F with the knowledge of F_j. To see this, note that 1) if F_j occurs, then the path fails regardless of F_i; 2) if F_j does not occur, then F_i does not occur as well. If the supply nodes of v_j form a subset of the supply nodes of v_i, then F_i is redundant. With an abuse of language, we call a node v_i redundant if F_i (*i.e.*, the state of v_i) is redundant. With this simplification, we derive the following result.

Let $n_s(v_i)$ denote the number of distinct supply nodes of v_i. Let $n_s^{\min} = \min_{1 \le i \le m} n_s(v_i)$. After removing the redundant nodes sequentially, let $\bar{m}$ be the number of remaining nodes that each have $n_s^{\min}$ supply nodes. The path failure probability can be estimated by the following theorem.

Theorem 3.3. *If every supply node fails independently with probability $p \le \epsilon/m$, then the path failure probability satisfies $(1-\epsilon)\bar{m}p^{n_s^{\min}} \le \Pr(F) \le (1+\epsilon)\bar{m}p^{n_s^{\min}}$.*

Proof. We first reduce the number of failure events that appear in the inclusion-exclusion formula by removing the redundant nodes. Note that determining whether

a node is redundant and removing the redundant node are done sequentially. Thus, among the set of nodes that have the same supply nodes, one node remains. Let D denote the nodes in the path excluding the redundant nodes.

First, we consider the first term in Eq. (3.1) that provides an upper bound on the path failure probability, known as the union bound. Let $D_1 \subset D$ denote the set of nodes that each have $n_s^{\min}$ supply nodes, and let $\bar{m} = |D_1|$. The remaining nodes $D_2 = D \setminus D_1$ each have $n_s^{\min} + 1$ or more supply nodes. Thus, the first term of Eq. (3.1) is at most

$$\begin{aligned}\Pr(F) \leq & \bar{m}p^{n_s^{\min}} + (m - \bar{m})pp^{n_s^{\min}} \\ \leq & \bar{m}p^{n_s^{\min}} + \epsilon p^{n_s^{\min}},\end{aligned}$$

for $p \leq \epsilon/m$.

Next, we consider the first two terms that provide a lower bound on the path failure probability (*c.f.*Bonferroni inequalities). For any pair of nodes $v_j, v_k \in D$, the union of their supply node sets contains at least $\max(n_s(v_j), n_s(v_k)) + 1$ nodes, because neither supply node set includes the other as a subset. At least $n_s^{\min} + 1$ supply nodes have to be removed in order for a pair of nodes in D_1 to fail. At least $n_s^{\min} + 2$ supply nodes need to be removed in order for a pair of nodes to fail if at least one node belongs to D_2. The absolute value of the second term is at most $\binom{\bar{m}}{2}p^{n_s^{\min}+1} + [\binom{m}{2} - \binom{\bar{m}}{2}]p^{n_s^{\min}+2}$. A lower bound on $\Pr(F)$ is

$$\begin{aligned}\Pr(F) \geq & \bar{m}p^{n_s^{\min}} - \left(\frac{\bar{m}^2}{2}pp^{n_s^{\min}} + \frac{m^2}{2}p^2p^{n_s^{\min}}\right) \\ \geq & \bar{m}p^{n_s^{\min}} - \epsilon\bar{m}p^{n_s^{\min}},\end{aligned}$$

for $p \leq \epsilon/m$. □

For the special case where every node in the path has the same number n_s of distinct supply nodes, let $\bar{m}$ be the number of nodes, in the path, among which no pair of nodes share the same set of n_s supply nodes. The following stronger result

can be proved in a similar approach.

Corollary 3.1. *If every supply node fails independently with probability* $p \leq 2\epsilon/\bar{m}$, *then the path failure probability satisfies* $(1-\epsilon)\bar{m}p^{n_s} \leq \Pr(F) \leq \bar{m}p^{n_s}$.

Proof. If node v_i and v_j in the path share the same set of supply nodes, then v_i and v_j must fail simultaneously, and $F = \cup_{1\leq i\leq m} F_i = \cup_{1\leq i\leq m, i\neq j} F_i$. Thus, in the calculation of path failure probability $\Pr(F)$, nodes that have the same set of supply nodes can be represented by a single node.

Let $\{\bar{v}_1, \bar{v}_2, \ldots, \bar{v}_{\bar{m}}\}$ denote the nodes in the path such that the supply nodes of $\bar{v}_i$ differ from the supply nodes of $\bar{v}_j$ by at least one supply node, $i \neq j$, $\bar{m} \leq m$. The first term of Eq. (3.1) is $\bar{m}p^{n_s}$, since every node has n_s distinct supply nodes and the probability that a node fails is p^{n_s}. Moreover, the union of the supply nodes of $\bar{v}_i$ and $\bar{v}_j$ has size at least $n_s + 1$, and the probability that both $\bar{v}_i$ and $\bar{v}_j$ fail (because of their supply nodes' failures) is at most p^{n_s+1}. The absolute value of the second term of Eq. (3.1) is at most $\binom{\bar{m}}{2}p^{n_s+1} \leq \bar{m}^2 p^{n_s+1}/2 \leq \epsilon\bar{m}p^{n_s}$, for $p \leq 2\epsilon/\bar{m}$. Therefore, $\Pr(F) \in [(1-\epsilon)\bar{m}p^{n_s}, \bar{m}p^{n_s}]$ given that $p \leq 2\epsilon/\bar{m}$. □

Thus, we have obtained the following two *reliability indicators* for a path. These combinatorial properties are useful in finding a reliable path, which will be studied in the next section.

- $n_s^{\min}$: the minimum number of distinct supply nodes for a node in the path.
- $\bar{m}$: the number of combinations of $n_s^{\min}$ supply node failures that lead to the failure of at least one node in the path.

3.2.2 Arbitrary failure probability

In contrast with the case where supply node failure probability is small and identical, it is difficult to characterize the reliability of a path by its combinatorial properties, with limited knowledge of node failure probabilities. Therefore, we obtain bounds on path failure probability that will be useful in finding a reliable path.

First, we develop an upper bound on the path failure probability. Let $p(v_i)$ be the failure probability of node v_i, under the condition that each of its supply nodes u_j^i fails independently with probability $p(u_j^i)$. The path failure probability, under the condition that the failures of V are positively correlated, is no larger than the path failure probability by assuming that the failures of V are independent.

Lemma 3.1. *The failure probability of a path P where a supply node u_j^i fails independently with probability $p(u_j^i)$ is upper bounded by $1 - \prod_{v_i \in P}(1 - p(v_i))$.*

Proof. If nodes v_i and $\cup_k v_k$ do not share any supply node, then the event that v_i survives and the event that $\cup_k v_k$ survive are independent. Otherwise, if they share one or more common supply nodes, the two events are positively correlated. Therefore,

$$\Pr(v_i \text{ and } \cup_k v_k \text{ survive}) \geq \Pr(v_i \text{ survives}) \Pr(\cup_k v_k \text{ survive}),$$

and

$$\Pr(v_i \text{ survives} | \cup_k v_k \text{ survive}) \geq 1 - p(v_i).$$

The reliability of a path $P = v_1 - v_2 - \cdots - v_m$ is given by

$$\begin{aligned}
\Pr(P \text{ survives}) &= \Pr(\cup_{k \in \{1,\ldots,m\}} v_k \text{ survive}) \\
&= \Pr(v_1 \text{ survives}) \Pr(v_2 \text{ survives} | v_1 \text{ survives}) \\
&\ldots \Pr(v_m \text{ survives} | \cup_{k \in \{1,\ldots,m-1\}} v_k \text{ survive}) \\
&\geq \prod_{v_i \in P} (1 - p(v_i)).
\end{aligned}$$

□

Then, we develop a lower bound on the path failure probability. The intuition is as follows. After replacing a supply node that supports multiple demand nodes by multiple independent supply nodes with sufficiently small failure probability, the path failure probability does not increase. In the original graph $G(V, E, \mathcal{S}_V)$, consider a node $v_i \in V$. Let U^i denote the set of supply nodes of v_i, let $u_j^i \in U^i$ denote one supply node, let $p(u_j^i)$ denote the failure probability of u_j^i, and let $n_d(u_j^i)$ denote the number

of nodes that u_j^i supports. Let $\tilde{p}(v_i) = \prod_{u_j^i \in U^i} \tilde{p}(u_j^i)$ denote the failure probability of v_i if u_j^i fails independently with probability $\tilde{p}(u_j^i) = 1 - (1 - p(u_j^i))^{1/n_d(u_j^i)}$. A lower bound on the path failure probability is as follows, whose proof follows a similar technique in [58] and is in the technical report.

Lemma 3.2. *The failure probability of a path P where a supply node u_j^i fails independently with probability $p(u_j^i)$ is lower bounded by $1 - \prod_{v_i \in P}(1 - \tilde{p}(v_i))$.*

Proof. Let $U^P = \cup_{1 \le i \le m} U^i$ denote the set of supply nodes for the nodes in path P. Let $u_j^i \in U^P$ denote one supply node. Let $n_d(u_j^i)$ denote the total number of demand nodes that u_j^i supports. Let P_j^i denote the set of nodes in P that are supported by u_j^i and $|P_j^i| = n_d(u_j^i, P) \le n_d(u_j^i)$. We follow a similar method in [58] to prove the claim.

Given the realizations of $U^P \setminus u_j^i = U_S \cup U_F$, where U_S denote the survived nodes and U_F denote the failed nodes, there are three possibilities. First, each node in P has at least one survived supply node in U_S, and P survives regardless of the state of u_j^i. Second, there exists at least one node in P whose supply nodes are all in U_F. Thus, P fails regardless of the state of u_j^i. Third, P survives if and only if u_j^i survives.

The last case occurs if for some nodes in P_j^i, all the other supply nodes have failed except u_j^i. The probability that P survives is given by $1 - p(u_j^i)$. By replacing u_j^i with $n_d(u_j^i, P)$ *distinct* nodes, each of which supports a node in P_j^i and fails independently with probability $1 - (1 - p(u_j^i))^{1/n_d(u_j^i)}$, the probability that all the $n_d(u_j^i, P)$ nodes survive is $(1 - p(u_j^i))^{n_d(u_j^i, P)/n_d(u_j^i)} \ge 1 - p(u_j^i)$. In this case, each node in P_j^i has at least one survived supply node, and the path P survives.

Thus, by the law of total probability, the probability that P survives never decreases after the above replacement.

After repeatedly replacing each supply node that supports multiple demand nodes by distinct nodes, each of which supports a single demand node and fails independently with the specified probability, the demand node failures become independent. Let $\tilde{p}(v_i)$ denote the failure probability of a demand node v_i after the replacement of supply nodes. The failure probability of a path can be computed efficiently as

$1-\prod_{v_i\in P}(1-\tilde{p}(v_i))$, and is a lower bound on the failure probability of the same path in the original problem. □

Let n_d denote the maximum number of demand nodes that a supply node supports, and let n_s denote the maximum number of supply nodes for a demand node. The following lemma bounds the ratio between the upper and lower bounds. Its proof can be found in the Appendix.

Lemma 3.3. *For any path, the ratio of the upper bound on its failure probability obtained in Lemma 3.1 to the lower bound obtained in Lemma 3.2 is at most* $(n_d)^{n_s}$.

Proof. We aim to prove

$$\frac{1-\prod_{v_i\in P}(1-p(v_i))}{1-\prod_{v_i\in P}(1-\tilde{p}(v_i))} \leq n_d^{n_s},$$

given

$$1-p(v_i) \geq (1-\tilde{p}(v_i))^{n_d^{n_s}}, \tag{3.2}$$

which will be proved in Lemma 3.7.

Given $p(v_i), \tilde{p}(v_i) \in (0,1)$, with Eq. (3.2),

$$1-\prod_{v_i\in P}(1-p(v_i)) \leq 1-\prod_{v_i\in P}(1-\tilde{p}(v_i))^{n_d^{n_s}}. \tag{3.3}$$

and

$$0 < \prod_{v_i\in P}(1-\tilde{p}(v_i)) < 1.$$

Moreover, let

$$f(x) = 1 - x^{n_d^{n_s}} - n_d^{n_s}(1-x).$$

Since $f(1) = 0$ and

$$f'(x) = n_d^{n_s}(1 - x^{n_d^{n_s}-1}) \geq 0,$$

for $0 < x \leq 1$ and $n_d^{n_s} \geq 1$, $f(x)$ is an increasing function, and

$$f(x) \leq 0,$$

for $0 < x \leq 1$ and $n_d^{n_s} \geq 1$.

Let $x = \prod_{v_i \in P}(1 - \tilde{p}(v_i)) \in (0, 1)$, since $f(x) \leq 0$, we obtain that

$$1 - \prod_{v_i \in P}(1 - \tilde{p}(v_i))^{n_d^{n_s}} \leq n_d^{n_s}(1 - \prod_{v_i \in P}(1 - \tilde{p}(v_i))).$$

With Eq. (3.3),

$$1 - \prod_{v_i \in P}(1 - p(v_i)) \leq n_d^{n_s}(1 - \prod_{v_i \in P}(1 - \tilde{p}(v_i))).$$

The claim is proved. □

3.3 Finding the most reliable path

In this section, we aim to compute the most reliable path between a source-destination pair $s, t \in V$ in $G(V, E, \mathcal{S}_V)$. We first prove that it is NP-hard to approximately compute the most reliable path. We then develop an algorithm to compute the most reliable path when the supply nodes fail independently with an identically small probability, and finally develop an approximation algorithm under arbitrary failure probabilities.

Hardness of approximation: Although the failure probability of any given path can be approximated to within factor $1 \pm \epsilon$ for any $\epsilon > 0$, it is NP-hard to obtain an $s - t$ path whose failure probability is less than $1 + \epsilon$ times the optimal for a small ϵ. The proof can be found in the Appendix.

Theorem 3.4. *Computing an $s - t$ path whose failure probability is less than $1 + \epsilon$ times the failure probability of the most reliable $s - t$ path is NP-hard for $\epsilon < 1/m$, where m is the maximum path length.*

Small and identical failure probability

If every supply node fails independently with an identically small probability, there are two reliability indicators: $n_s^{\min}$ and $\bar{m}$. Recall that $n_s^{\min}$ is the minimum number

of supply nodes for a node in the path, and that $\bar{m}$ is the number of combinations of $n_s^{\min}$ supply node failures that disconnect the path. With the two indicators, the path failure probability can be approximated to within a multiplicative factor $1 \pm \epsilon$ by $\bar{m}p^{n_s^{\min}}$, under the condition that $p \leq \epsilon/m$. Moreover, the indicator $n_s^{\min}$ is more important (and has a higher priority to be optimized) than $\bar{m}$. We next develop algorithms to optimize the two indicators.

Given a graph $G(V, E, \mathcal{S}_V)$ and a pair of nodes (s, t), the problem of computing an $s-t$ path with the maximum $n_s^{\min}$ can be formulated as the *maximum capacity path* problem, where the capacity of a node equals the number of its distinct supply nodes and the capacity of a path is the minimum node capacity along the path. The maximum capacity path can be obtained by a modified Dijkstra's algorithm, and can be obtained in linear time [59].

However, it is NP-hard to minimize $\bar{m}$, even in the special case where every demand node has a single supply node. The result follows from the NP-hardness of computing a path with the minimum colors in a colored graph [37].

We develop an integer program to compute the path P with the minimum $\bar{m}$, under the condition that $n_s^{\min}(P) = \min_{v_i \in P} n_s(v_i)$ is maximized. The following preprocessing reduces the size of the integer program. First, compute $k = \max_{P \in \mathcal{P}} n_s^{\min}(P)$, where $\mathcal{P}$ is the set of all the $s-t$ paths, using the linear-time maximum capacity path algorithm. Then, remove all the nodes that have fewer than k distinct supply nodes and their attached edges, and denote the remaining graph by $G'(V', E', \mathcal{S}_{V'})$. The removed nodes and edges will not be used by the optimal path. Let $V'' \subseteq V'$ denote the nodes among which each has exactly k distinct supply nodes. We aim to find a path V_P where the number of distinct supply node sets for $V_P \cap V''$ is minimized.

Let S_i denote the set of supply nodes of $i \in V''$. Let $\mathcal{S}_{V''}$ denote the union of these sets. Let x_{ij} denote the flow variable which takes a positive value if and only if edge (i, j) belongs to the selected path. An $s-t$ path is identified by constraint (3.5). A node i is on the selected path if at least one of x_{ij} and x_{ji} is positive. Let $h(S_i)$ denote whether removing supply nodes S_i disconnects the selected path. If a node i is on the selected path and has k supply nodes, then $h(S_i)$ must be one, guaranteed

by constraint (3.6). All the other nodes either do not belong to the selected path or have more than k supply nodes, and their supply node failures are not considered. The objective minimizes $\tilde{m}$, which is the number of combinations of k supply node failures that disconnect the path.

$$\min \sum_{S_i \in \mathcal{S}_{V''}} h(S_i) \tag{3.4}$$

$$\text{s.t.} \sum_{\{j|(i,j)\in E'\}} x_{ij} - \sum_{\{j|(j,i)\in E'\}} x_{ji} = \begin{cases} 1, \text{ if } i = s, \\ -1, \text{ if } i = t, \\ 0, \text{ otherwise.} \end{cases} \tag{3.5}$$

$$\sum_{\{j|(i,j)\in E'\}} x_{ij} + \sum_{\{j|(j,i)\in E'\}} x_{ji} \leq 2h(S_i), \ \forall i \in V'' \setminus s,t, \tag{3.6}$$

$$x_{ij} \geq 0, \quad \forall (i,j) \in E',$$

$$h(S_i) = \{0,1\}, \quad \forall S_i \in \mathcal{S}_{V''}.$$

Arbitrary failure probability

If nodes $\tilde{V}$ in a graph $\tilde{G}(\tilde{V},\tilde{E})$ fail independently, the probability that a path survives is the product of the survival probabilities of nodes along the path. The most reliable path can be obtained by the classical shortest path algorithm, by replacing the length of traversing a node $\tilde{v}_i$ by $-\ln(1-p(\tilde{v}_i))$, where $p(\tilde{v}_i)$ is the failure probability of $\tilde{v}_i$. It is easy to see that the length of a path P is $\sum_{\tilde{v}_i\in P} -\ln(1-p(\tilde{v}_i)) = -\ln\prod_{\tilde{v}_i\in P}(1-p(\tilde{v}_i))$. The shortest path has the smallest failure probability $1-\prod_{\tilde{v}_i\in P}(1-p(\tilde{v}_i))$.

Compared with the above simple model, the difficulty in obtaining the most reliable $s-t$ path in interdependent networks is the failure correlations of nodes $V \subseteq G(V,E,\mathcal{S}_V)$. The failure probability of a path can no longer be characterized by $1-\prod_{v_i\in P}(1-p(v_i))$. Moreover, let $s-\cdots-v_i-\cdots-t$ be the most reliable $s-t$ path. The sub-path $s-\cdots-v_i$ may not be the most reliable path between s and v_i. Thus, the label-correction approach in dynamic programming (*e.g.*, Dijkstra's

algorithm) cannot be used, even though the failure probability of a given path can be approximated.

Given the bounds obtained in the previous section, we propose Algorithm 3.2 to compute a path whose failure probability is within $(n_d)^{n_s}$ times the optimal failure probability. Recall that the bounds on path survival probability are the product of (original or new) node survival probabilities, which exactly match the path survival probability in the case of independent node failures.

Algorithm 3.2 An approximation algorithm to compute a reliable $s-t$ path in $G(V, E, \mathcal{S}_V)$.

---

1. For each $v_i \in V$, compute $\tilde{p}(v_i)$ as follows. Let u_j^i be a supply node of v_i with failure probability $p(u_j^i)$. If u_j^i supports $n_d(u_j^i)$ nodes, let $\tilde{p}(u_j^i) = 1 - (1 - p(u_j^i))^{1/n_d(u_j^i)}$. Let $\tilde{p}(v_i)$ be the failure probability of v_i if u_j^i fails independently with probability $\tilde{p}(u_j^i)$.
2. Compute the most reliable $s-t$ path assuming that v_i fails independently with probability $\tilde{p}(v_i)$. The most reliable path can be obtained by a standard shortest path algorithm (*e.g.*, Dijkstra's algorithm), by letting $-\ln(1-\tilde{p}(v_i))$ be the length of traversing node v_i.

Theorem 3.5. *The failure probability of the path obtained by Algorithm 3.2 is at most $(n_d)^{n_s}$ times the failure probability of the most reliable $s-t$ path under arbitrary supply node failure probabilities.*

Proof. Let the path obtained by Algorithm 3.2 be P' and let the path with the minimum failure probability be P^*. Let $p(P')$ and $p(P^*)$ denote their failure probabilities. Moreover, let $\tilde{p}(P')$ and $\tilde{p}(P^*)$ denote their failure probabilities by assuming that each node v_i fails independently with probability $\tilde{p}(v_i)$. We have $p(P') \leq n_d^{n_s}\tilde{p}(P') \leq n_d^{n_s}\tilde{p}(P^*) \leq n_d^{n_s}p(P^*)$, where the first inequality follows from Lemma 3.3 and the last inequality follows from Lemma 3.2. □

Remark. If $n_s = 1$ and every supply node fails independently with an identically small probability, our result reduces to the following result in the classical shared risk group model: The number of risks associated with the shortest path is at most n_d times the number of risks associated with the minimum-risk path [45].

3.4 Reliability of a pair of paths

To study diverse routing in interdependent networks, we consider the simplest case of two $s-t$ paths in this section. Given that computing the failure probability of a single path is $\#P$ hard if every node has more than one supply node, it is also $\#P$ hard to compute the failure probability of two paths[2]. To see this, note that if two paths have the same number of nodes and each node in the first path has identical supply nodes as its corresponding node in the second path, then the probability that both paths fail equals the probability that a single path fails. Fortunately, we are still able to obtain $1 \pm \epsilon$-approximation of the failure probability in polynomial time.

Small and identical failure probability

A central concept in diverse routing is the *disjoint paths* or *risk disjoint paths* [35–37]. In the classical shared risk group model, if every risk occurs independently with an identically small probability $p = o(1/m^2)$, the probability that two paths fail is $\Theta(f(m)p^2)$ if they are risk disjoint and $\Theta(f(m)p)$ if they share one or more risks, where m is the maximum path length and $f(m)$ is a function of m. Thus, risk-disjointness characterizes the order of the reliability of two paths. In interdependent networks where every demand node has multiple supply nodes, if nodes in P^1 do not share any supply nodes with nodes in P^2, then P^1 and P^2 are risk disjoint. However, risk-disjointness does not suffice to characterize the reliability of two paths, for the following two reasons. First, the failure probability of a demand node depends on the number of supply nodes for it, which is not related to risk-disjointness. Second, if P^1 and P^2 share some supply nodes, the failure probability depends further on the maximum number of supply node failures that the two paths can withstand. To study the reliability of two paths in interdependent networks, we define *d-failure resilient paths* as follows.

[2]Meanwhile, it is still simple to compute the failure probability of two paths if every node has a single supply node, by first computing the probability that the first path fail, and then computing the probability that the second path fail while the first path does not fail (*i.e.*, none of the supply nodes of the first path fail), both in polynomial time, and summing the two probabilities.

Definition 3.1. Two paths are d-failure resilient if removing any d supply nodes would not disconnect both paths.

Remark. In the classical graph model $\tilde{G}(\tilde{V}, \tilde{E})$, two disjoint paths are 1-failure resilient while two overlapping paths are 0-failure resilient. In the classical shared risk group model, two risk disjoint paths are 1-failure resilient while two paths that share risks are 0-failure resilient. Two paths can never be more than one failure resilient. Thus, the disjointness or risk-disjointness suffices to characterize (the order of) the reliability of two paths in these models.

Evaluation of failure probability

Consider two paths $P^1 = s - v_1^1 - v_2^1 - \cdots - v_{m_1}^1 - t$, $P^2 = s - v_1^2 - v_2^2 - \cdots - v_{m_2}^2 - t$ between a pair of nodes (s, t). We study the event that at least one node in P^1 and at least one node in P^2 both fail. Let F_i^k denote the event that all the supply nodes of v_i^k fail, and let F^k denote the event that the k-th path fails, $k \in \{1, 2\}$. Then $F^1 \cap F^2 = \cup_{1 \le i \le m_1, 1 \le j \le m_2} (F_i^1 \cap F_j^2)$. For simplicity of presentation, let $F^{\text{both}} = F^1 \cap F^2$ and $F_{ij} = F_i^1 \cap F_j^2$. Let S_{ij} denote the union of supply nodes of v_i^1 and v_j^2.

To decide whether two paths are d-failure resilient, we consider the number of supply node failures that lead to the event F_{ij}, and denote the number by d_{ij}. Then $d = \min_{1 \le i \le m_1, 1 \le j \le m_2} d_{ij} - 1$. Moreover, let $\bar{m}$ be the number of pairs of nodes, one from each path, such that each pair of nodes in total have $d + 1$ distinct supply nodes and any two pairs do not have the same set of $d + 1$ supply nodes. (*I.e.*, $\bar{m}$ combinations of $d + 1$ supply node failures each disconnect both paths.) The next theorem formalizes the connection between the reliability of two paths and d.

Theorem 3.6. *If every supply node fails independently with probability $p \le \epsilon/(m_1 m_2)$, then the probability that two d-failure resilient paths with lengths m_1, m_2 both fail satisfies $(1 - \epsilon)\bar{m}p^{d+1} \le \Pr(F^{both}) \le (1 + \epsilon)\bar{m}p^{d+1}$.*

Proof. First consider the events $F_{ij} = F_i^1 \cap F_j^2$, $1 \le i \le m_1, 1 \le j \le m_2$. Let S_{ij} denote the union of supply nodes of v_i^1 and v_j^2. Then the event F_{ij} occurs if

and only if all the nodes S_{ij} fail. By a similar argument as the proof of Theorem 3.3, if $S_{i_1j_1}$ is a subset of $S_{i_2j_2}$, then $F_{i_2j_2}$ occurs only if $F_{i_1j_1}$ occurs, and $F_{i_2j_2}$ is redundant. In the following we only consider $\mathcal{S} = \{S_{ij} | 1 \leq i \leq m_1, 1 \leq j \leq m_2,$ none of S_{ij} is a subset of another.$\}$. The cardinality of $\mathcal{S}$ at most $m_1 m_2$.

By the inclusion-exclusion principle, $\Pr(F^{\text{both}})$ can be computed as follows.

$$\begin{aligned}\Pr(F^{\text{both}}) = &\sum_{S_{ij} \in \mathcal{S}} \Pr(S_{ij} \text{ fail}) \\ &- \sum_{S_{i_1j_1}, S_{i_2j_2} \in \mathcal{S}} \Pr(S_{i_1j_1} \cup S_{i_2j_2} \text{ fail}) \\ &+ \cdots + (-1)^{|\mathcal{S}|-1} \Pr(\cup_{S_{ij} \in \mathcal{S}} S_{ij} \text{ fail}). \end{aligned} \tag{3.7}$$

Since two paths are d-failure disjoint, the number of nodes in S_{ij} is $d+1$ for some $i \in \{1, \ldots, m_1\}, j \in \{1, \ldots, m_2\}$ while the number of nodes in all the other S_{ij} is larger than $d+1$. Let $\mathcal{S}_1 \subseteq \mathcal{S}$ be the union of the supply node sets, each of which contains $d+1$ nodes, and let $\bar{m} = |\mathcal{S}_1|$. The first term in Eq. (3.7) is at most $\bar{m} q^{d+1} + (|\mathcal{S}| - \bar{m}) q^{d+2}$. Therefore,

$$\begin{aligned}\Pr(F^{\text{both}}) \leq &\bar{m} q^{d+1} + m_1 m_2 q q^{d+1} \\ \leq &\bar{m} q^{d+1} + \epsilon q^{d+1},\end{aligned}$$

if $q \leq \epsilon/(m_1 m_2)$.

We next consider the supply node failures of two pairs of nodes. Recall that $\mathcal{S}_1$ consists of supply node sets that each contain $d+1$ nodes. Let $\mathcal{S}_2 = \mathcal{S} \setminus \mathcal{S}_1$ be the remaining supply node sets that each contain $d+2$ or more nodes. The union of two sets $S_{i_1j_1} \cup S_{i_2j_2}$ ($S_{i_1j_1}, S_{i_2j_2} \in \mathcal{S}_1$) contains at least $d+2$ nodes. The union $S_{i_1j_1} \cup S_{i_2j_2}$ ($S_{i_1j_1}, S_{i_2j_2} \in \mathcal{S}_2$, or $S_{i_1j_1} \in \mathcal{S}_1, S_{i_2j_2} \in \mathcal{S}_2$) contains at least $d+3$ nodes. The absolute value of the second term is at most $\binom{\bar{m}}{2} q^{d+2} + [\binom{m_1 m_2}{2} - \binom{\bar{m}}{2}] q^{d+3}$. To conclude,

$$\begin{aligned}\Pr(F^{\text{both}}) \geq &\bar{m} q^{d+1} - \left(\frac{\bar{m}^2}{2} q q^{d+1} + \frac{(m_1 m_2)^2}{2} q^2 q^{d+1} \right) \\ \geq &\bar{m} q^{d+1} - \epsilon \bar{m} q^{d+1},\end{aligned}$$

if $q \leq \epsilon/(m_1 m_2)$.

□

Finding the most reliable pair of paths

From Theorem 3.6, we know that the probability that two d-failure resilient paths both fail is smaller for larger values of d. Moreover, for a fixed d, the failure probability is proportional to $\bar{m}$, the number of combinations of $d+1$ supply node failures that disconnect both paths. We have obtained two reliability indicators for two paths: d and $\bar{m}$.

Unfortunately, computing the pair of $s-t$ paths that have the maximum d and the minimum $\bar{m}$ are both NP-hard, even in the special case where every demand node has a single supply node. This special case reduces to the classical shared risk group model. In this special case, $d = 1$ if there exist two risk-disjoint paths, and $d = 0$ otherwise. The NP-hardness of determining the existence of two risk-disjoint paths between an $s-t$ pair has been proved in [35]. Moreover, in this special case, for two paths that share common supply nodes, $\bar{m}$ is the number of overlapping risks between the two paths (*i.e.*, removing any of the $\bar{m}$ supply nodes disconnects both paths). The NP-hardness of the least coupled paths problem, which computes a pair of paths that share the minimum number of risks in the classical shared risk group model, has also been proved in [35].

We develop an integer program to compute a pair of $s-t$ paths with the maximum d in $G(V, E, \mathcal{S}_V)$. Let variable x_{ij}^k denote whether edge (i,j) is part of the k-th path, and let variable b_i^k denote whether node i is part of the k-th path, $k \in \{1,2\}$. Same as before, let S_i denote the supply nodes of node i. Constraints (3.9) guarantee that two paths are node-disjoint. Notice that these constraints can be dropped if there is no restriction on the physical disjointness of two paths. Constraints (3.10) guarantee that at least $d+1$ supply nodes need to be removed in order for one node in each path to fail (*i.e.*, $b_i^1 = b_j^2 = 1$, $i, j \in V$), where M is a sufficiently large number, *e.g.*,

twice the maximum number of supply nodes for a demand node.

$$\max \quad d \tag{3.8}$$

$$\text{s.t.} \sum_{\{j|(i,j)\in E\}} x_{ij}^k - \sum_{\{j|(j,i)\in E\}} x_{ji}^k = \begin{cases} 1, \text{ if } i = s, \\ -1, \text{ if } i = t, \quad k \in \{1,2\}, \\ 0, \text{ otherwise.} \end{cases}$$

$$\sum_{\{j|(i,j)\in E\}} x_{ij}^k + \sum_{\{j|(j,i)\in E\}} x_{ji}^k \leq 2b_i^k, \quad \forall i \in V, k \in \{1,2\}$$

$$b_i^1 + b_i^2 \leq 1, \ \forall i \in V \setminus s,t, \tag{3.9}$$

$$d + 1 \leq |S_i \cup S_j| + M(2 - b_i^1 - b_j^2), \forall i,j \in V \setminus s,t, \tag{3.10}$$

$$x_{ij}^k \in \{0,1\}, \quad \forall (i,j) \in E, k \in \{1,2\},$$

$$b_i^k \in \{0,1\}, \quad \forall i \in V, k \in \{1,2\}.$$

A slightly modified integer program suffices to minimize $\bar{m}$ under the condition that d is maximized. Let $h(S_i \cup S_j)$ denote whether removing the union of supply nodes for i and j disconnects both paths. Constraints (3.12) guarantee that if i and j belong to two different paths, *i.e.*, $b_i^1 = b_j^2 = 1$, then $h(S_i \cup S_j) = 1$. Otherwise, $h(S_i \cup S_j) = 0$ in the optimal solution. Let a positive value $w(|S_i \cup S_j|)$ denote its *weight*, which is a decreasing function of the cardinality $|S_i \cup S_j|$. We aim to minimize the total weights of supply node failures that disconnect two paths. In order to guarantee that d is maximized, $w(l)/w(l+1)$ should be sufficiently large for any integer l, *e.g.*, $|V|^2/2$. Since there are at most $|V|(|V|-1)/2$ pairs of nodes, larger d is always preferable and has a higher priority to be optimized over $\bar{m}$.

$$\min \sum_{S_i,S_j \in \mathcal{S}_V} w(|S_i \cup S_j|) h(S_i \cup S_j) \tag{3.11}$$

$$\text{s.t.} \sum_{\{j|(i,j)\in E\}} x_{ij}^k - \sum_{\{j|(j,i)\in E\}} x_{ji}^k = \begin{cases} 1, \text{ if } i = s, \\ -1, \text{ if } i = t, \quad k \in \{1,2\}, \\ 0, \text{ otherwise.} \end{cases}$$

$$
\begin{aligned}
&\sum_{\{j|(i,j)\in E\}} x_{ij}^k + \sum_{\{j|(j,i)\in E\}} x_{ji}^k \leq 2b_i^k, \quad \forall i \in V, k \in \{1,2\} \\
&b_i^1 + b_i^2 \leq 1, \ \forall i \in V \setminus s,t, \\
&h(S_i \cup S_j) \geq b_i^1 + b_j^2 - 1, \quad \forall i,j \in V \setminus s,t, \\
&x_{ij}^k \in \{0,1\}, \quad \forall (i,j) \in E, k \in \{1,2\}, \\
&h(S_i \cup S_j) \in \{0,1\}, \quad \forall i,j \in V.
\end{aligned} \tag{3.12}
$$

Arbitrary failure probability

Evaluation of failure probability

We use a similar importance sampling approach to Algorithm 3.1 and formulate the problem of computing the failure probability of two paths as a DNF probability problem. A clause C_{ij} represents a pair of nodes v_i^1 and v_j^2. Literals in C_{ij} represent the union of supply nodes of v_i^1 and v_j^2. A literal is true if and only if the supply node that it represents fails, and the probability that the literal is true is the same as the supply node failure probability. The disjunction of clauses is true if and only if at least one clause is true, in which case both paths fail because at least one node from each path fails. The rest of the computation follows the same manner as Algorithm 3.1, by replacing a node in Algorithm 3.1 by a pair of nodes. An (ϵ, δ)-approximation of the failure probability $\Pr(F^{\text{both}})$ can be obtained in $O(m_1^2 m_2^2 n_s \ln(1/\delta)/\epsilon^2)$ time.

Finding the most reliable pair of paths

It is more difficult to find two paths that have the smallest failure probability. Recall Theorem 3.6. The failure probability of two paths is $\Theta(f(m_1, m_2)p^{d+1})$ if they are d-failure resilient when the supply node failure probability p is small, where $f(m_1, m_2)$ is a function of two path lengths. As a corollary of the fact that it is NP-hard to compute two paths that have the maximum level of resilience d, it is also NP-hard to compute two paths whose failure probability is within a factor α from the optimal, where α is any function of the network size. Thus, we develop the following heuristic. After computing the failure probability $\tilde{p}(v_i)$ of a node v_i

in Step 1 of Algorithm 3.2, let $-\ln(1-\tilde{p}(v_i))$ be the length of traversing node v_i, and compute two node disjoint paths with the minimum total lengths. The two paths can be efficiently obtained using a slightly modified shortest augmenting path algorithm [60]. The computation is outlined in Algorithm 3.3. The reason for the graph transformation in Step 1 is to simplify the computation of a residual graph, to which the shortest augmenting path algorithm can be applied.

Algorithm 3.3 A heuristic to compute a pair of reliable $s-t$ path in $G(V, E, \mathcal{S}_V)$.

1. Transform $G(V, E, \mathcal{S}_V)$ with node failure probabilities to a directed graph G' with edge failure probabilities using the standard approach. (Split every node v into v_{in} and v_{out}. Add a directed edge from v_{in} to v_{out}, which has length $-\ln(1-\tilde{p}(v_i))$. Add a directed edge from $v_{1\text{out}}$ to $v_{2\text{in}}$ and a directed edge from $v_{2\text{out}}$ to $v_{1\text{in}}$, both with zero length, if an edge exists between v_1 and v_2 in $G(V, E, \mathcal{S}_V)$.)
2. Compute the shortest path P_1' from s_{out} to t_{in} in G'.
3. Compute the residual graph. Remove all the edges in P_1'. Add a backward edge from v_2' to v_1' with a negated length if an edge from v_1' to v_2' is part of P_1'.
4. Compute the shortest path P_2' from s_{out} to t_{in} in the residual graph.
5. Combine P_1' and P_2' by cycle cancellation. The two paths become node-disjoint and can be mapped to two paths in $G(V, E, \mathcal{S}_V)$.

3.5 Numerical results

We study the robust routing problems in the XO backbone communication network with 60 nodes and 75 edges [56], by assuming that the XO nodes are supported by 36 randomly generated supply nodes within the continental US. The XO network topology is depicted in Fig. 3-2, and the supply nodes are marked as triangles. The x-axis represents the longitude and the y-axis represents the latitude. We do not claim that the XO network needs supply from these randomly generated points, and we use this example only to provide a visualization of the robust routing problems using available data.

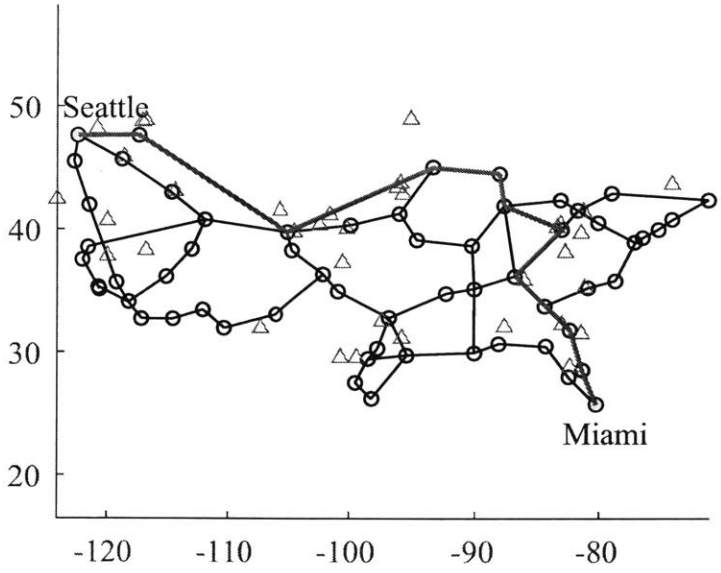


Figure 3-2: Topology of the XO network and randomly generated triangle supply nodes. The most reliable Seattle-Miami path is colored red under the condition that supply node failure probability is small and identical and every XO node depends on two nearest supply nodes.

First, we assume that every XO node depends on two nearest supply nodes and every supply node fails independently with probability 10^{-2}. Since the supply node failure probability is small and identical, we are able to obtain the most reliable path and pair of paths by optimizing the reliability indicators using integer programs.

To identify the most reliable path, since $n_s^{\min}(P) = 2$ for any path P, we only need to compute a path with the minimum $\bar{m}$ using the integer program in Section 3.3. The most reliable path is colored red in Fig. 3-2, for which $\bar{m} = 8$. To evaluate the path failure probability, by Corollary 3.1, setting $\epsilon = 4 \times 10^{-2}$, $\Pr(F) \in [7.68 \times 10^{-4}, 8 \times 10^{-4}]$. To compare, using Algorithm 3.1, we obtain 7.9686×10^{-4} as a (1 ± 0.01)-approximation of the path failure probability with probability 0.99. These results suggest that the two reliability indicators $(n_s^{\min}, \bar{m})$ well characterize the path failure probability when the supply node failure probability is small and identical.

We compute the most reliable pair of paths connecting Seattle-Miami using the integer programs in Section 3.4. The two paths are plotted in Fig. 3-3, and they

are 1-failure resilient ($d = 1$). The failure probability of both paths is approximately 1.0388×10^{-4}. In contrast, the most reliable pair of paths connecting Seattle-Denver are 3-failure resilient and their failure probability is approximately 2.9800×10^{-8}. Thus, the level of resilience well indicates the reliability of two paths.

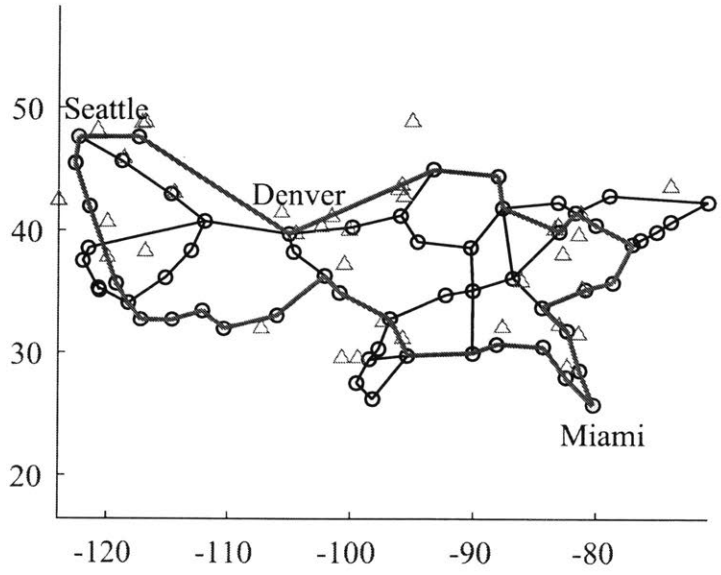


Figure 3-3: The most reliable pair of paths between Seattle-Miami are colored red, under the condition that supply node failure probability is small and identical and every XO node depends on two nearest supply nodes.

Next, we assume that an XO node depends on N_s randomly chosen supply nodes, where N_s is uniformly chosen among 1, 2, and 3. Let the failure probability of each supply node be uniformly and independently chosen from $[0.005, 0.015]$. We use Algorithm 3.2 to obtain a reliable path connecting Seattle-Miami. Averaged over 10 trials, the path failure probability is approximately 2.1032×10^{-2}, while the lower bound on the failure probability of the most reliable path is 5.2365×10^{-3}. The obtained path has failure probability around four times the lower bound. Moreover, by using the heuristic to find a pair of paths, the paths have average failure probability 3.9732×10^{-3}, which improves the reliability of a single path.

We compare the performance of the heuristic (Algorithm 3.3) with the optimal pair of paths. Since it is difficult to obtain the optimal pair of paths under arbitrary failure

probabilities, we use the integer program (3.11), under the condition that supply nodes fail independently with probability 10^{-2}. If every XO node depends on two nearest supply nodes, the failure probabilities of two optimal paths and two paths obtained by the heuristic are approximately 1.0388×10^{-4} and 1.0773×10^{-4}, respectively. If every XO node depends on three nearest supply nodes, the failure probability of two optimal paths and two paths obtained by the heuristic are approximately 1.0200×10^{-6} and 1.0508×10^{-6}, respectively. These experiments validate the performance of our heuristic algorithm.

Finally, we report the running times of the algorithms, executed in a workstation that has an Intel Xeon Processor (E5-2687W v3) and 64GB RAM. The integer programs that find the most reliable path and pair of paths (under small and identical supply node failure probability) can both be solved within 1 second. The approximation algorithm to find a reliable path and the heuristic to find a pair of paths (under arbitrary failure probabilities) can both be solved within 0.1 second. The evaluation of the failure probability of one path or a pair of paths by Algorithm 3.1 takes several minutes, by setting $\epsilon = \delta = 0.01$. Thus, the algorithms (integer programs and Algorithms 3.2 and 3.3) can be used to find reliable routes in realistic size networks.

3.6 Summary

We studied the robust routing problem in interdependent networks. We developed approximation algorithms to compute the path failure probability, and identified reliability indicators for a path, based on which we develop algorithms to find the most reliable route in interdependent networks. We also studied diverse routing in interdependent networks, and developed approximation algorithms to compute the probability that two paths both fail and to find two reliable paths. Our work extends the shared risk group models, and provides a new framework to study robust routing problems in interdependent networks.

3.7 Chapter appendix

3.7.1 Computational complexity

Proof of Theorem 3.1. The problem of computing the path failure probability can be reduced from a *monotone DNF counting* problem. A monotone DNF counting problem aims to compute the number of satisfying assignments of literals, for a DNF formula that has no negated literals. The monotone DNF counting problem is $\#P$-hard, even if every clause contains two literals [61]. (The original paper [61] considers conjunctive normal form counting, monotone 2-CNF(SAT). It is easy to see the equivalence between the monotone 2-DNF and monotone 2-CNF by applying De Morgan's law and negating all the literals.)

Given a monotone DNF counting problem, construct a path as follows. Each node in the path represents a clause, and its supply nodes represent the literals in the clause. (See Fig. 3-4 for an example.)

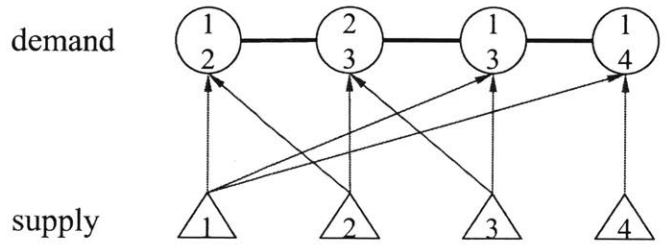


Figure 3-4: A path constructed from a monotone DNF formula $(x_1 \wedge x_2) \vee (x_2 \wedge x_3) \vee (x_1 \wedge x_3) \vee (x_1 \wedge x_4)$.

If every supply node fails independently with probability $1/2$, then the path failure probability is $N/2^m$, where m is the total number of supply nodes (literals), and N is the number of combinations of supply node failures that lead to the failure of at least one node, which equals the number of satisfying assignments for the DNF formula. Thus, the failure probability of a path under $p = 1/2$ gives an answer to the monotone DNF counting problem. To conclude, computing the path failure probability is $\#P$-hard if every node has two or more supply nodes. □

Remark. We further consider the complexity of computing the path reliability, with

additional restrictions on the maximum number of demand nodes that any supply node supports. If every supply node supports at most two demand nodes, in addition to the restriction that every demand node has at most two supply nodes, then the failure probability can be computed in polynomial time, when every supply node fails independently with an identical probability. The computation follows from the algorithm in [62], which relates the number of satisfying assignments of a DNF formula, where each literal appears at most twice and each clause contains two literals, to the number of independent sets in a graph with node degree at most two. Nevertheless, if every supply node supports three or more demand nodes, the computation becomes $\#P$-hard even if every demand node has at most two supply nodes, because counting the number of independent sets in a graph with node degree three is $\#P$-hard [63].

Proof of Theorem 3.4. We prove that computing such a path is NP-hard even in the following restricted case. Consider a graph where every node has a single supply node, and every supply node fails independently with an identically small probability $p = o(1/m^2)$. The failure probability of a path supported by $\bar{m} \leq m$ supply nodes is $1 - (1-p)^{\bar{m}} = \bar{m}p + o(p)$.

Suppose that the most reliable path is supported by $\bar{m}_{\min}$ supply nodes and has failure probability $p_{\min} = \bar{m}_{\min}p + o(p)$. For $\epsilon < 1/m \leq 1/\bar{m}_{\min}$, a path with failure probability strictly smaller than $(1+\epsilon)p_{\min} < (\bar{m}_{\min}+1)p + o(p)$ is supported by exactly $\bar{m}_{\min}$ supply nodes. Computing a path that is supported by the minimum number of supply nodes in this example is NP-hard, which is known as the minimum color path problem in [37]. Therefore, computing a path that has failure probability within $1+\epsilon$ times the optimal is NP-hard for $\epsilon < 1/m$. □

3.7.2 Approximating the path failure probability by importance sampling

In this section, we prove the correctness of Algorithm 3.1 and Theorem 3.2.

Lemma 3.4. *The path failure probability is given by* $E[I]\sum_{1\leq k\leq m}\prod_{1\leq j\leq n_s(v_k)} p(u_j^k)$.

Proof. Let V_s denote the set of all supply nodes. Let U denote a set of failed supply nodes that lead to the failure of at least one of $\{v_1, v_2, \ldots, v_m\}$ (*i.e.*, the failure of the path). Let $\mathcal{U} = \{U_1, U_2, \ldots, U_R\}$ denote all the sets of supply node failures that lead to the failure of the path. Let $p(u)$ denote the failure probability of node u. Let Pr(exactly U_r fail) denote the probability that supply nodes U_r fail and all the other supply nodes $V_s \setminus U_r$ do not fail. Since the events that U_r fail while the others do not fail are mutually exclusive for different r, the path failure probability is given by

$$\begin{aligned}\Pr(F) &= \sum_{1\le r\le R} \Pr(\text{exactly } U_r \text{ fail}) \\ &= \sum_{1\le r\le R} \prod_{u\in U_r} p(u) \prod_{u\in V_s\setminus U_r} (1-p(u)). \end{aligned} \tag{3.13}$$

Let $\{u_j^k | j = 1, \ldots, n_s(v_k)\}$ denote the set of supply nodes of v_k. Let m_r denote the number of demand node failures, among $\{v_1, v_2, \ldots, v_m\}$, if supply nodes U_r fail. Then, by summing the failure probabilities of demand nodes $p(v_k) = \prod_{1\le j\le n_s(v_k)} p(u_j^k)$, $k = 1, \ldots, m$, the probability that supply nodes U_r fail is counted m_r times.

$$\begin{aligned}\sum_{1\le k\le m} \prod_{1\le j\le n_s(v_k)} p(u_j^k) &= \sum_{1\le r\le R} m_r \Pr(\text{exactly } U_r \text{ fail}) \\ &= \sum_{1\le r\le R} m_r \prod_{u\in U_r} p(u) \prod_{u\in V_s\setminus U_r} (1-p(u)). \end{aligned} \tag{3.14}$$

We now construct the relationship between the left hand sides of Eq. (3.13) and Eq. (3.14) using $E[I]$. In Algorithm 3.1, the value of I in Step 4 depends on both v_i (obtained in Step 2) and U (obtained in Step 3). In the remainder of the proof, we first compute the probability that a specific U is obtained (in an iteration of the main loop), and then compute $\Pr(I = 1|U)$ (*i.e.*, the probability that v_i is the first failed node given that U fail). As a consequence, $\Pr[I = 1]$ can be determined using the law of total probability.

Consider an iteration of the main loop. Let $\Pr(U_r)$ denote the probability that U_r is obtained in Step 3 of Algorithm 3.1. Let $r(t), t = 1, \ldots, m_r$ denote the indices of

failed nodes $v_{r(t)}$ among $\{v_1, v_2, \ldots, v_m\}$ if U_r fail ($m_r \leq m$). We have

$$\Pr(U_r) = \sum_{1 \leq t \leq m_r} \left(\frac{\prod_{1 \leq j \leq n_s(v_{r(t)})} p(u_j^{r(t)})}{\sum_{1 \leq k \leq m} \prod_{1 \leq j \leq n_s(v_k)} p(u_j^k)} \right.$$

$$\left. \times \prod_{u \in U_r \setminus \{u_j^{r(t)}, 1 \leq j \leq n_s(v_{r(t)})\}} p(u) \prod_{u \in V_s \setminus U_r} (1 - p(u)) \right) \tag{3.15}$$

$$= \sum_{1 \leq t \leq m_r} \frac{\prod_{u \in U_r} p(u) \prod_{u \in V_s \setminus U_r} (1 - p(u))}{\sum_{1 \leq k \leq m} \prod_{1 \leq j \leq n_s(v_k)} p(u_j^k)} \tag{3.16}$$

$$= \frac{m_r \prod_{u \in U_r} p(u) \prod_{u \in V_s \setminus U_r} (1 - p(u))}{\sum_{1 \leq k \leq m} \prod_{1 \leq j \leq n_s(v_k)} p(u_j^k)}. \tag{3.17}$$

To see this, note that there are m_r choices of $\{v_{r(t)} | t = 1, \ldots, m_r\}$ which may lead to U_r. In Eq. (3.15), the first term (in the product) is the probability of choosing $v_{r(t)}$ and setting its supply nodes $U^{r(t)} = \{u_j^{r(t)} | j = 1, \ldots, n_s(v_{r(t)})\}$ to be failed; the second term is the probability that $U_r \setminus U^{r(t)}$ fail; the last term is the probability that the remaining supply nodes $V_s \setminus U_r$ do not fail.

Eq. (3.16) implies that $v_{r(t)}, t \in \{1, \ldots, m_r\}$ contribute equally to the occurrence of U_r. Namely,

$$\begin{aligned}
&\Pr(v_{r(t)} \text{ has been chosen in Step 2} | U_r) \\
=&\left(\frac{\prod_{1 \leq j \leq n_s(v_{r(t)})} p(u_j^{r(t)})}{\sum_{1 \leq k \leq m} \prod_{1 \leq j \leq n_s(v_k)} p(u_j^k)} \right. \\
&\left. \times \prod_{u \in U_r \setminus \{u_j^{r(t)}, 1 \leq j \leq n_s(v_{r(t)})\}} p(u) \prod_{u \in V_s \setminus U_r} (1 - p(u)) \right) \Big/ \Pr(U_r) \\
=&1/m_r,
\end{aligned}$$

for $t \in \{1, \ldots, m_r\}$.

Given U^r, the probability that $v_{r(1)}$ has been chosen in Step 2 of Algorithm 3.1 is

$1/m_r$. Thus, $\Pr[I = 1|U_r] = 1/m_r$. By the law of total probability,

$$\begin{aligned}\Pr[I = 1] &= \sum_{1 \le r \le R} \Pr[I = 1|U_r] \Pr(U_r) \\ &= \frac{\sum_{1 \le r \le R} \prod_{u \in U_r} p(u) \prod_{u \in V_s \setminus U_r} (1 - p(u))}{\sum_{1 \le k \le m} \prod_{1 \le j \le n_s(v_k)} p(u_j^k)}.\end{aligned}$$

Since I is an indicator variable, $E[I] = \Pr(I = 1)$.

$$\begin{aligned}&E[I] \sum_{1 \le k \le m} \prod_{1 \le j \le n_s(v_k)} p(u_j^k) \\ &= \sum_{1 \le r \le R} \prod_{u \in U_r} p(u) \prod_{u \in V_s \setminus U_r} (1 - p(u)) \\ &= \Pr(F).\end{aligned}$$

□

Next, we prove that $E[I]$ can be estimated accurately within $3m \ln(2/\delta)/\epsilon^2$ iterations.

Lemma 3.5.

$$\Pr\left(\left|\frac{E[I] - b/a}{E[I]}\right| \ge \epsilon E[I]\right) \le \delta,$$

where $a = 3m \ln(2/\delta)/\epsilon^2$ is the number of iterations of the main loop of Algorithm 3.1, b is the number of observations of $I = 1$, and $0 < \epsilon < 1$. Namely, by repeating $a = 3m \ln(2/\delta)/\epsilon^2$ times, one obtains an (ϵ, δ)-approximation of $E[I]$.

Proof. The proof is based on the Chernoff inequality and is a standard result in estimation theory. From the proof of Lemma 3.4, we know that $\Pr(I = 1|U^r) = 1/m^r \ge 1/m$ for all U^r. Thus, $\Pr(I = 1) \ge 1/m$. To estimate $E[I]$ within $1 \pm \epsilon$ accuracy $(0 < \epsilon < 1)$, let the number of trials be $a = 3m \ln(2/\delta)/\epsilon^2$.

$$\begin{aligned}&\Pr\left(\left|\sum_{1 \le i \le a} I_i - \sum_{1 \le i \le a} E[I_i]\right| \ge \epsilon \sum_{1 \le i \le a} E[I_i]\right) \\ &\le \exp(-\frac{\epsilon^2 \sum_{1 \le i \le a} E[I_i]}{2}) + \exp(-\frac{\epsilon^2 \sum_{1 \le i \le a} E[I_i]}{3})\end{aligned}$$

$$\leq 2\exp(-\frac{\epsilon^2 a/m}{3}) = 2\exp(-\frac{3\ln(2/\delta)}{3})$$
$$\leq \delta.$$

□

Proof of Theorem 3.2. Consider an iteration of the main loop of Algorithm 3.1. In Step 2, obtaining v_i takes $O(mn_s)$ time. In Step 3, obtaining U takes $O(mn_s)$ time, because the total number of supply nodes is at most $O(mn_s)$. In Step 4, testing whether v_i is the first failed node under the failure of U takes $O(mn_s)$ time, given that checking whether a node fail takes $O(n_s)$ time and there are at most m nodes in the path.

Since $3m\ln(2/\delta)/\epsilon^2$ iterations are sufficient, the total running time of Algorithm 3.1 is $O(m^2 n_s \ln(1/\delta)/\epsilon^2)$. □

3.7.3 Bounds on path failure probability

Lemma 3.6.

$$1-(1-p_1p_2)^{\alpha\beta} \leq [1-(1-p_1)^{\alpha}][1-(1-p_2)^{\beta}],$$

for $p_1, p_2 \in (0,1), \alpha, \beta \in (0,1]$.

Proof. Let

$$g(x) = (1-x)^{\gamma} - (1-\gamma x),$$

for $x \in [0,1), \gamma \in (0,1]$.

By taking the derivatives,

$$g'(x) = -\gamma(1-x)^{\gamma-1} + \gamma;$$

$$g''(x) = \gamma(\gamma-1)(1-x)^{\gamma-2}.$$

Since $g(0) = 0$, according to the mean value theorem,

$$g(x) = g'(\xi)x,$$

where $0 < \xi \leq x < 1$. Substituting $g(x)$ and $g'(\xi)$,

$$\begin{aligned} (1-x)^\gamma - (1-\gamma x) &= g'(\xi)x \\ 1-(1-x)^\gamma &= \gamma x - g'(\xi)x \\ 1-(1-x)^\gamma &= \gamma x(1-\xi)^{\gamma-1}. \end{aligned} \tag{3.18}$$

Given $g''(x) < 0$ for $x, \gamma \in (0,1)$, $g(x)$ is strictly concave for $x, \gamma \in (0,1)$. For $0 < x_1 < x_2 < 1$,

$$\begin{aligned} g(x_2) &< g(0) + \frac{g(x_1) - g(0)}{x_1} x_2, \\ g(x_2)/x_2 &< g(x_1)/x_1. \end{aligned}$$

Given that $g(x_1) = g'(\xi_1)x_1$, $g(x_2) = g'(\xi_2)x_2$, $0 < \xi_1 < x_1$, $0 < \xi_2 < x_2$, and that $g'(x)$ is decreasing in x, we have $\xi_1 < \xi_2$. If $\gamma = 1$, then $g(x) = g'(x) = 0$ for $x \in [0,1)$. Clearly, there also exist $\xi_1 < \xi_2$ such that $g(x_1) = g'(\xi_1)x_1$, $g(x_2) = g'(\xi_2)x_2$.

Applying Eq. (3.18),

$$\begin{aligned} & [1-(1-p_1)^\alpha][1-(1-p_2)^\beta] \\ = \; & \alpha p_1(1-\eta_1)^{\alpha-1}\beta p_2(1-\eta_2)^{\beta-1} \\ = \; & \alpha\beta p_1 p_2(1-\eta_1)^{\alpha-1}(1-\eta_2)^{\beta-1}, \end{aligned}$$

for $0 < \eta_1 < p_1$, $0 < \eta_2 < p_2$, and

$$1-(1-p_1p_2)^{\alpha\beta} = \alpha\beta p_1 p_2(1-\eta_3)^{\alpha\beta-1},$$

for $0 < \eta_3 < p_1p_2$. Moreover, $\eta_3 \leq \min(\eta_1, \eta_2)$, because $p_1p_2 \leq \min(p_1, p_2)$.

Given $0 < \alpha, \beta \leq 1$,

$$\begin{aligned} & (1-\eta_1)^{\alpha-1}(1-\eta_2)^{\beta-1} \\ \geq\ & (1-\eta_3)^{\alpha-1}(1-\eta_3)^{\beta-1} \\ =\ & (1-\eta_3)^{\alpha+\beta-2} \\ \geq\ & (1-\eta_3)^{\alpha\beta-1}, \end{aligned}$$

where the first inequality follows from the fact that $h_1(x) = (1-x)^\alpha$ is decreasing in x if $x \in (0,1)$ and $\alpha \in (0,1]$, and the last inequality follows from

$$\begin{aligned} \alpha(1-\beta) &\leq 1-\beta, \\ \alpha+\beta-2 &\leq \alpha\beta-1, \end{aligned}$$

and $h_2(x) = (1-\eta_3)^x$ is decreasing in x for $\eta_3 \in (0,1)$.

Therefore,

$$1-(1-p_1p_2)^{\alpha\beta} \leq [1-(1-p_1)^\alpha][1-(1-p_2)^\beta].$$

□

Lemma 3.7.

$$1-p(v_i) \geq (1-\tilde{p}(v_i))^{n_d^{n_s}},$$

where $\tilde{p}(v_i)$ is defined before Lemma 3.2.

Proof. For $n_s = 1$, every node has a single supply node. Let u_j^i be the supply node of v_i. Since $1-p(u_j^i) \geq (1-\tilde{p}(u_j^i))^{n_d}$ for any supply node u_j^i, the result trivially holds.

We next focus on the case where $n_s \geq 2$. Recall that $p(v_i) = \prod_{u_j^i} p(u_j^i)$ and $\tilde{p}(v_i) = \prod_{u_j^i} \tilde{p}(u_j^i)$ (*i.e.*, a demand node fails if and only if all of its supply nodes fail), where u_j^i are the supply nodes of v_i.

Consider two supply nodes of v_i and let $p(u_1^i)$ and $p(u_2^i)$ be their failure probabilities. Moreover, $\tilde{p}(u_1^i)$ and $\tilde{p}(u_2^i)$ satisfy $\tilde{p}(u_1^i) \geq 1-(1-p(u_1^i))^{1/n_d}$ and $\tilde{p}(u_2^i) \geq 1-(1-p(u_2^i))^{1/n_d}$.

Then,

$$\begin{aligned} 1-\tilde{p}(u_1^i)\tilde{p}(u_2^i) &\leq 1-[(1-(1-p(u_1^i))^{1/n_d}) \\ &\qquad (1-(1-p(u_2^i))^{1/n_d})] \\ &\leq (1-p(u_1^i)p(u_2^i))^{1/n_d^2}, \end{aligned}$$

where the last inequality follows from Eq. (3.19), by letting $p_1 = p(u_1^i), p_2 = p(u_2^i), \alpha, \beta = 1/n_d$, which we proved in Lemma 3.6.

$$1-(1-p_1p_2)^{\alpha\beta} \leq [1-(1-p_1)^{\alpha}][1-(1-p_2)^{\beta}], \tag{3.19}$$

for $p_1, p_2 \in (0,1), \alpha, \beta \in (0,1]$.

Consider the third supply node of v_i which has failure probability $p(u_3^i)$. We have $\tilde{p}(u_3^i) \geq 1-(1-p(u_3^i))^{1/n_d}$. Moreover, notice that $\tilde{p}(u_1^i)\tilde{p}(u_2^i) \geq 1-(1-p(u_1^i)p(u_2^i))^{1/n_d^2}$. By letting $p_1 = p(u_1^i)p(u_2^i), p_2 = p(u_3^i), \alpha = 1/n_d^2, \beta = 1/n_d$ in Eq. 3.19, we have

$$1-\tilde{p}(u_1^i)\tilde{p}(u_2^i)\tilde{p}(u_3^i) \leq (1-p(u_1^i)p(u_2^i)p(u_3^i))^{1/n_d^3}.$$

By repeating the process until all the supply nodes of v_i are considered, and let $n_s(v_i) \leq n_s$ denote the number of supply nodes of v_i, we have

$$\begin{aligned} 1-\tilde{p}(v_i) &\leq (1-p(v_i))^{1/n_d^{n_s(v_i)}} \\ &\leq (1-p(v_i))^{1/n_d^{n_s}}. \end{aligned}$$

□

Chapter 4

Robustness of interdependent random geometric networks

Previous chapters developed a layered graph model to represent arbitrary topologies of interdependent networks. In this chapter, we develop an interdependent random geometric graph (RGG) model to study the properties of large-scale interdependent networks. In this model, nearby nodes are connected by edges, which represent the topology of many physical networks. We study the conditions under which a positive fraction of nodes are functional in interdependent RGGs as the number of nodes approaches infinity. In this case, the interdependent RGGs *percolate*. We extend percolation theory to interdependent RGGs, and study the robustness of interdependent RGGs under random and geographical node failures.

The rest of the chapter is organized as follows. We state the model and preliminaries in Section 4.1. We derive analytical upper bounds on percolation thresholds in Section 4.2, and obtain confidence intervals for percolation thresholds in Section 4.3. In Section 4.4, we study the robustness of interdependent RGGs under random failures and geographical attacks. In Section 4.5, we extend the techniques to study graphs with more general interdependence. Section 4.6 concludes the chapter.

4.1 Model

4.1.1 Preliminaries on RGG and percolation

An RGG in a two-dimensional square consists of nodes generated by a Poisson point process and links connecting nodes within a given connection distance [64]. Let $G(\lambda, d, a^2)$ denote an RGG with node density λ and connection distance d in an $a \times a$ square. The studies on RGG focus on the regime where the expected number of nodes $n = \lambda a^2$ is large. We first present some preliminaries which are useful for developing our model. The *giant component* of an RGG is a connected component that contains $\Theta(n)$ nodes. A node belongs to the giant component with a positive probability $\Theta(n)/n$ if the giant component exists. For a given connection distance, the *percolation threshold* is a node density above which a node belongs to the giant component with a positive probability (*i.e.*, a giant component exists) and below which the probability is zero (*i.e.*, no giant component exists). By scaling, if the percolation threshold is λ^* under connection distance d, then the percolation threshold is $\lambda^* c^2$ under connection distance d/c. Therefore, without loss of generality, in this chapter, we study the percolation thresholds represented by node densities, for given connection distances.

The RGG is closely related to the *Poisson boolean model* [65], where nodes are generated by a Poisson point process on an *infinite plane*. Let $G(\lambda, d)$ denote a Poisson boolean model with node density λ and connection distance d. The difference between $G(\lambda, d)$ and $G(\lambda, d, a^2)$ is that the number of nodes in $G(\lambda, d)$ is infinite while the expected number of nodes in $G(\lambda, d, a^2)$ is large but finite. The Poisson boolean model can be viewed as a limit of the RGG as the number of nodes approaches infinity. The percolation threshold of $G(\lambda, d)$ under a given d is defined as the node density above which a node belongs to the *infinite component* with a positive probability and below which the probability is zero. It has been shown that a node belongs to the infinite component with a positive probability if and only if an infinite component exists, and thus the percolation of $G(\lambda, d)$ can be equivalently defined as the existence of the infinite component [65]. Moreover, the percolation threshold of $G(\lambda, d)$ is identical

with the percolation threshold of $G(\lambda, d, a^2)$ [64, 66].

4.1.2 Interdependent RGGs

Two interdependent networks are modeled by two RGGs $G_1(\lambda_1, d_1, a^2)$ and $G_2(\lambda_2, d_2, a^2)$ on the *same* $a \times a$ square. A node in one graph is interdependent with *all* the nodes in the other graph within the *interdependent distance* d_{dep}. See Fig. 4-1 for an illustration. Nodes in one graph are *supply nodes* for nodes in the other graph within d_{dep}. The physical interpretation of supply can be either electric power or information that is essential for proper operation. A node can receive supply from nearby nodes within the interdependent distance. Larger interdependent distance leads to more robust interdependent networks. The geographical nature of interdependence is observed in physical networks [1, 12].

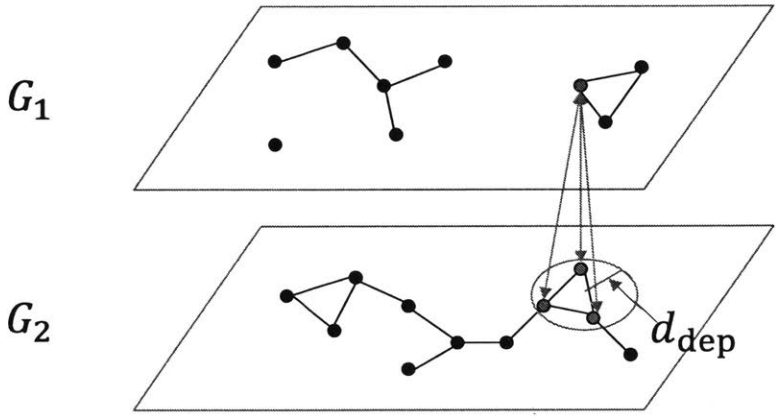


Figure 4-1: Two interdependent RGGs with interdependent distance d_{dep}.

Most analysis in this chapter is given in the context of two interdependent Poisson boolean models $G_{\text{IntDep}} = (G_1(\lambda_1, d_1), G_2(\lambda_2, d_2), d_{\text{dep}})$, which is the limit of two interdependent RGGs as the numbers of nodes in both graphs approach infinity.

We define a mutual component and an infinite mutual component in G_{IntDep}, in the same way as one defines a connected component and an infinite component in $G(\lambda, d)$.

Definition 4.1. Let V_i^0 denote nodes in a connected component in $G_i(\lambda_i, d_i)$, $\forall i \in \{1, 2\}$. If each node in $V_i \subseteq V_i^0$ has at least one supply node in $V_j \subseteq V_j^0$ within d_{dep},

$\forall i, j \in \{1, 2\}, i \neq j$, then nodes V_1 and V_2 form a *mutual component* of G_{IntDep}.

If, in addition, V_i contains an infinite number of nodes, $\forall i \in \{1, 2\}$, then V_1 and V_2 form an *infinite mutual component.*

A mutual component can be viewed as an autonomous system in the sense that nodes in a mutual component have supply nodes in the same mutual component, and in each graph, nodes that belong to a mutual component are connected regardless of the existence of nodes outside the mutual component. Note that a node can receive supply from any of its supply nodes in the same mutual component, and thus is content if it has at least one supply node. Nodes in an infinite mutual component are *functional*, since they constitute two large connected interdependent networks and can perform a given network function (*e.g.*, data communication or power transmission to a large number of clients). This definition of functional is consistent with previous research on interdependent networks based on random graph models [7].

For a fixed d_{dep}, if an infinite mutual component exists in $G_{\text{IntDep}} = (G_1(\lambda_1, d_1), G_2(\lambda_2, d_2), d_{\text{dep}})$, then an infinite mutual component exists in $G'_{\text{IntDep}} = (G_1(\lambda'_1, d_1), G_2(\lambda_2, d_2), d_{\text{dep}})$, where $\lambda'_1 > \lambda_1$. This can be explained by coupling G'_1 with G_1 as follows. By removing each node in G'_1 independently with probability $1 - \lambda_1/\lambda'_1$, the density of the remaining nodes in G'_1 is λ_1, and an infinite mutual component exists in the interdependent graphs that consist of G_2 and the graph formed by the remaining nodes in G'_1. Since adding nodes to a graph does not disconnect any mutual component, an infinite mutual component exists in $G'_{\text{IntDep}} = (G_1(\lambda'_1, d_1), G_2(\lambda_2, d_2), d_{\text{dep}})$. By the same analysis, an infinite mutual component also exists in $G''_{\text{IntDep}} = (G_1(\lambda_1, d_1), G_2(\lambda'_2, d_2), d_{\text{dep}})$, if $\lambda'_2 > \lambda_2$.

We define a percolation threshold of G_{IntDep} as follows.

Definition 4.2. A pair of node densities $(\lambda_1^*, \lambda_2^*)$ is a *percolation threshold* of G_{IntDep}, given connection distances d_1, d_2 and the interdependent distance d_{dep}, if an infinite mutual component exists in G_{IntDep} for $\lambda_1 > \lambda_1^*$ and $\lambda_2 > \lambda_2^*$, and no infinite mutual component exists otherwise.

For fixed d_1, d_2 and d_{dep}, there may exist multiple percolation thresholds. We

show that, in most cases, the larger the node density is in one graph, the smaller the required node density is in the other graph in order for the infinite mutual component to exist. This is in contrast with the situation for a single graph $G(\lambda, d)$ where there is a unique percolation threshold λ^* for a fixed d.

There is a non-trivial phase transition in G_{IntDep}. If λ_i is smaller than the percolation threshold of a single graph $G_i(\lambda_i, d_i)$, there is no infinite component in $G_i(\lambda_i, d_i)$, and therefore there is no infinite mutual component in G_{IntDep}. Thus, $\lambda_i^* > 0$, $\forall i \in \{1, 2\}$. As we will see in the next section, there exist percolation thresholds $\lambda_i^* < \infty$, $\forall i \in \{1, 2\}$, which concludes the non-trivial phase transition.

Given that the conditions for the percolation of a random geometric graph $G_i(\lambda_i, d_i, a^2)$ and a Poisson boolean model $G_i(\lambda_i, d_i)$ are the same, the above definitions can be naturally extended to interdependent RGGs. Consider nodes $V_1 \subseteq G_1(\lambda_1, d_1, a^2)$ and $V_2 \subseteq G_2(\lambda_2, d_2, a^2)$ that form a mutual component. If V_i contains $\Theta(n_i)$ nodes, where $n_i = \lambda_i a^2$, $\forall i \in \{1, 2\}$, then V_1 and V_2 form a *giant mutual component* in interdependent RGGs. The percolation of interdependent RGGs is defined as the existence of a giant mutual component. In the rest of the chapter, we sometimes use G_i to denote both $G_i(\lambda_i, d_i, a^2)$ and $G_i(\lambda_i, d_i)$. The model that it refers to will be clear from the context.

4.1.3 Related work

In the interdependent networks literature, the model which is closest to ours is the interdependent lattice model, first proposed in [67] and further studied in [12,13]. In the lattice model, nodes in a network are represented by the open *sites* (nodes) of a square lattice, where every site is open independently with probability p. Network links are represented by the *bonds* (edges) between adjacent open sites. Every node in one lattice is interdependent with *one* randomly chosen node within distance r_d in the other lattice. The distance r_d indicates the geographical proximity of the interdependence. The percolation threshold of the interdependent lattice model is characterized as a function of r_d, assuming the same p in both lattices [67]. Percolation of the model where some nodes do not need to have supply nodes was studied in [12].

The analysis relies on quantities estimated by simulation and extrapolation, such as the fraction of nodes in the infinite component of a lattice for any fixed p, which cannot be computed rigorously. In contrast, we study the percolation of the interdependent RGG model using a mathematically rigorous approach.

The percolation of a single RGG (or a Poisson boolean model) has been studied in the previous literature [65,68,69]. The techniques employed therein involves inferring the percolation of the continuous model from the percolation of a discrete lattice model. The key is obtaining a lattice whose percolation condition is known and is related to the percolation of the original model, by discretization. The study of the percolation conditions of discrete lattice models can be found in [70,71]. We extend the previous techniques to discretize G_{IntDep}, and obtain bounds on the percolation thresholds.

4.2 Analytical upper bounds on percolation thresholds

In this section, we study sufficient conditions for the percolation of G_{IntDep}. We provide closed-form formulas for (λ_1, λ_2), which depend on d_1, d_2, d_{dep}, such that there exists an infinite mutual component in $G_{\text{IntDep}} = (G_1(\lambda_1, d_1), G_2(\lambda_2, d_2), d_{\text{dep}})$. The formulas provide guidelines for node densities in deploying physical interdependent networks, in order for a large number of nodes to be connected.

In G_{IntDep}, nodes in the infinite mutual component are viewed as functional while all the other nodes are not. Thus, a node is functional only if it is in the infinite component of its own graph, and it depends on at least one node in the infinite component of the other graph. For any node b_1 in G_1, although the number of nodes in G_2 within the interdependent distance from b_1 follows a Poisson distribution, the number of functional nodes is hard to calculate, since the probability that a node in G_2 is in the infinite component is unknown. Moreover, the nodes in the infinite component of G_2 are clustered, and thus the thinning of the nodes in G_1 due to a lack

of supply nodes in G_2 is inhomogeneous. To overcome these difficulties, we consider the percolation of two graphs jointly, instead of studying the percolation of one graph with reduced node density due to a lack of supply nodes.

We now give an overview of our approach. We develop mapping techniques (discretizations) to characterize the percolation of G_{IntDep} by the percolation of a discrete model. Mappings from a model whose percolation threshold is unknown to a model with known percolation threshold are commonly employed in the study of continuum percolation. For example, one can study the percolation threshold of the Poisson boolean model $G(\lambda, d)$ by mapping it to a triangle lattice and relating the state of a site in the triangle lattice to the point process of $G(\lambda, d)$. By the mapping, the percolation of the triangle lattice implies the percolation of $G(\lambda, d)$. Consequently, an upper bound on the percolation threshold of $G(\lambda, d)$ is given by λ for which the triangle lattice percolates, a known quantity [65,68]. In general, more than one mapping can be applied, and the key is to find a mapping that gives a good (smaller) upper bound. Following this idea, we propose different mappings that fit different conditions to obtain upper bounds on the percolation thresholds of G_{IntDep}.

In the rest of this section, we first study an example, in which the connection distances of the two graphs are the same, to understand the tradeoff between the two node densities in order for G_{IntDep} to percolate. We then develop two upper bounds on the percolation thresholds. The first bound is tighter when the ratio of the two connection distances is small, and is obtained by mapping G_{IntDep} to a square lattice with independent bond open probabilities. The second bound is tighter when the ratio of the two connection distances is large, and is obtained by mapping G_{IntDep} to a square lattice with correlated bond open probabilities.

4.2.1 A motivating example

To see the impact of varying the node density in one graph on the minimum node density in the other graph in order for G_{IntDep} to percolate, consider an example where $d_1 = d_2 = 2d_{\text{dep}}$. We apply a mapping similar to what is used to obtain an upper bound on the percolation threshold of $G(\lambda, d)$ in [68], to obtain upper bounds on the

percolation thresholds of G_{IntDep}.

Consider a triangle lattice where each site is surrounded by a cell. The lattice bond length is determined such that any two points in adjacent cells have distance smaller than $2r$, where $2r = d_1$. The boundary of the cell consists of arcs of radius r centered at the middle of the bonds in the triangle lattice. See Fig. 4-2 for an illustration. The area of the cell is $A = 0.8227r^2$. A site in the triangle lattice is either *open* or *closed*. If the probability that a site is open is strictly larger than $1/2$, open sites form an infinite component, and the triangle lattice percolates [68].

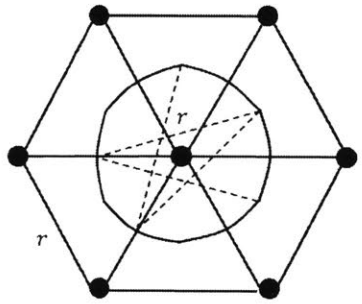


Figure 4-2: A cell that contains a site in a triangle lattice.

To study the percolation of G_{IntDep}, we declare a site in the triangle lattice to be open if there is at least one node in its cell from G_1 and at least one node in its cell from G_2. If the triangle lattice percolates, then G_{IntDep} also percolates. To see this, consider two adjacent open sites in the triangle lattice. Nodes from G_i in the two adjacent cells that contain the two open sites are connected, because they are within distance $d_i = 2r$ ($\forall i \in \{1, 2\}$). If the open sites in the triangle lattice form an infinite component, then nodes from G_i in the corresponding cells form an infinite component V_i ($\forall i \in \{1, 2\}$). Moreover, given that any pair of nodes in a cell are within distance $r \leq d_{\text{dep}}$, each node in V_i has at least one supply node in V_j within the same cell ($\forall i, j \in \{1, 2\}, i \neq j$).

Since $1 - e^{-\lambda_i A}$ is the probability that there is at least one node in the cell from G_i and the point processes in G_1 and G_2 are independent, an upper bound on the

percolation thresholds of G_{IntDep} is given by (λ_1, λ_2) satisfying

$$(1 - e^{-\lambda_1 A})(1 - e^{-\lambda_2 A}) = 1/2.$$

If λ_i is large, the percolation threshold λ_j^* approaches the threshold of a single graph G_j. Intuitively, if λ_j is above the percolation threshold of G_j, disks of radius $d_j/2$ centered at nodes in G_j form a connected infinite-size region. Since λ_i is large, nodes in G_i in this region are connected and form an infinite component. Moreover, since $d_{\text{dep}} = d_j/2$, all the nodes in this region have supply nodes, and they form an infinite mutual component.

The above upper bounds on percolation thresholds are still valid if $d_{\text{dep}} > d_i/2$, because each node can depend on a larger set of nodes by increasing d_{dep} and it is easier for G_{IntDep} to percolate under the same node densities and connection distances. However, if $d_{\text{dep}} < d_i/2$, the bond length of the triangle lattice should be adjusted to $r = d_{\text{dep}}$ in order for any pair of nodes in a cell to be within d_{dep}. The percolation threshold curve (λ_1, λ_2) would shift upward. Intuitively, if d_{dep} decreases, the node density in one network should increase to provide enough supply for the other network.

4.2.2 Small ratio d_2/d_1

Given $G_{\text{IntDep}} = (G_1(\lambda_1, d_1), G_2(\lambda_2, d_2), d_{\text{dep}})$, without loss of generality we assume that $d_1 \leq d_2$. Moreover, we assume that $d_{\text{dep}} \geq \max(d_1/2, d_2/2) = d_2/2$ (see the remark at the end of the section for comments on this assumption). Let $c = \lfloor d_2/d_1 \rfloor = \max\{c : d_2/d_1 \geq c, c \in \mathbb{N}\}$. For small c, we study the percolation of G_{IntDep} by mapping it to an independent bond percolation of a square lattice, and prove the following result.

Theorem 4.1. *If* (λ_1, λ_2) *satisfies*

$$(1 - e^{-\lambda_1 d_1^2/8})^c (1 - e^{-\lambda_2 c^2 d_1^2/8}) > 1/2,$$

then $G_{IntDep} = (G_1(\lambda_1, d_1), G_2(\lambda_2, d_2), d_{dep})$ *percolates, where* $c = \lfloor d_2/d_1 \rfloor$, $d_1 \leq d_2$,

and $d_{dep} \geq d_2/2$.

Theorem 4.1 provides a sufficient condition for the percolation of G_{IntDep}. For node densities that satisfy the inequality, an infinite mutual component exists in G_{IntDep}. For the deployment of interdependent networks, if the node densities in the two networks are sufficiently large (characterized by Theorem 4.1), then a large number of nodes in the interdependent networks are functional.

Proof of Theorem 4.1. We first construct a square lattice as follows. Partition the plane into small squares of side length $s = d_1/2\sqrt{2}$. A large square consists of $c \times c$ small squares and has side length cs. The *diagonals* of the large squares form the bonds of a square lattice L, illustrated by the thick line segments in Fig. 4-3.

The state of a bond in L is determined by the point process of G_{IntDep} in the large square that contains the bond. A bond (v_1, v_2) is open if the following conditions are both satisfied.

1. There is at least one node from G_1 in each of the two small squares that contain the ends (v_1 and v_2) of the bond, and they are connected through nodes from G_1, *all within the large square of side length* cs.

2. There is at least one node from G_2 in the large square that contains the bond.

The first condition is satisfied if there exists a sequence of adjacent small squares, each of which contains at least one node in G_1, from the small square that contains v_1 to the small square that contains v_2. (Each small square is *adjacent* to its eight immediate neighbors.) In the example of Fig. 4-3, these sequences include 3-5-7, 3-2-4-7, and 3-6-8-7.

To obtain a closed-form formula, instead of computing the exact probability, we compute a lower bound on the probability that the first condition is satisfied. The probability is lower bounded by the probability that the c small squares that intersect the bond each contain at least one node from G_1, given by

$$p_1 \geq (1 - e^{-\lambda_1 d_1^2/8})^c.$$

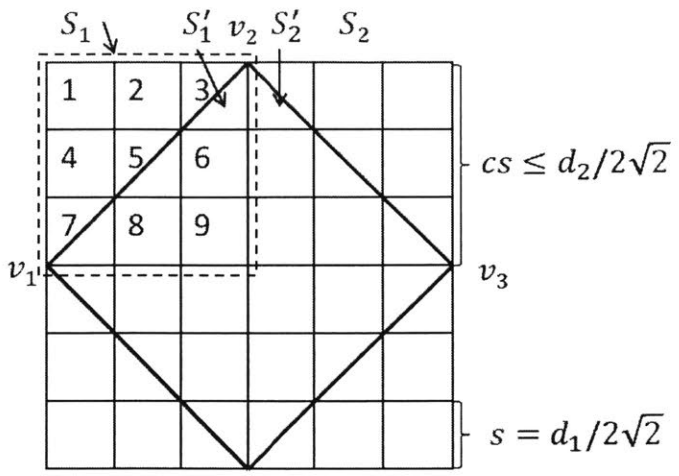


Figure 4-3: Mapping to a square lattice for $c = 3$.

The probability that the second condition is satisfied is

$$p_2 = 1 - e^{-\lambda_2 c^2 d_1^2/8}.$$

Given that the two Poisson point processes in G_1 and G_2 are independent, the probability that a bond is open is $p_1 p_2$.

It remains to prove that the percolation of L implies the percolation of G_{IntDep}. Consider two adjacent open bonds $(v_1, v_2), (v_2, v_3)$ in L. Let S_1 and S_2 denote the two adjacent large squares of side length cs that contain the two open bonds. Let S_1' and S_2' denote two adjacent small squares of side length s that contains v_2, within S_1 and S_2, respectively. See Fig. 4-3 for an illustration. Since $(v_1, v_2), (v_2, v_3)$ are open, under the second condition, nodes of G_2 exist in S_1 and S_2 and they are connected, because they are within distance $2\sqrt{2}cs \leq d_2$. Under the first condition, nodes of G_1 form a connected path from the small square (within S_1, marked as 7 in Fig. 4-3) containing v_1 to S_1', and another path from the small square (within S_2) containing v_3 to S_2'. Moreover, the two paths are joined, because any pair of nodes in S_1' and S_2' are within distance $2\sqrt{2}s = d_1$. Given that any pair of nodes within a large square have distance at most $\sqrt{2}cs \leq d_2/2 \leq d_{\text{dep}}$, all the nodes have at least one supply node inside the large square that contains an open bond. To conclude, if the open

bonds in L form an infinite component, then the nodes in G_{IntDep} form an infinite mutual component.

The event that a bond is open depends on the point processes in the large square that contains the bond, and is independent of whether any other bonds are open. As long as the probability that a bond is open, $p_1 p_2$, is larger than $1/2$, which is the threshold for independent bond percolation in a square lattice [71], G_{IntDep} percolates. □

The bound can be made tighter for any given $c = \lfloor d_2/d_1 \rfloor$, by computing more precisely the probability that the first condition is satisfied. We provide an example to illustrate the computation of an improved upper bound.

Example: Consider an example where $d_1 = 1, d_2 = 2d_{\text{dep}} = 3$. The probability that there is at least one node from G_2 in the large square of side length $3/2\sqrt{2}$ is $p_2 = 1 - e^{-9\lambda_2/8}$.

The probability that a small square of side length $1/2\sqrt{2}$ contains at least one node from G_1 is $p_s = 1 - e^{-\lambda_1/8}$. The probability that the first condition is satisfied is

$$p_1 = p_s^3 + (1 - p_s)p_s^4 + (1 - p_s)p_s^4 - (1 - p_s)p_s^6, \tag{4.1}$$

obtained by considering all the sequences of adjacent small squares. For node densities (λ_1, λ_2) that satisfy $p_1 p_2 > 1/2$, G_{IntDep} percolates. Since p_1 computed by Eq. (4.1) is larger than p_s^3 for any fixed p_s, the bound on λ_2 is smaller for any fixed λ_1.

4.2.3 Large ratio d_2/d_1

In the mapping from G_{IntDep} to the square lattice L, the condition for a bond to be open becomes overly restrictive as d_2/d_1 increases. A path crossing the two large squares that contain two adjacent bonds does not have to cross the small squares that contain the common end of the two bonds. In the following theorem, we give another upper bound on the percolation threshold of G_{IntDep}. This result provides an alternative sufficient condition for the existence of an infinite mutual component in

G_{IntDep}. This upper bound is tighter than the bound in Theorem 4.1 for larger values of d_2/d_1.

Theorem 4.2. *If* (λ_1, λ_2) *satisfies*

$$\left[1 - \frac{4}{3}(m+1)e^{m \log 3(1-p)}\right]\left[1 - \frac{4}{3}(2m+1)e^{m \log 3(1-p)}\right] p' > 0.8639,$$

then $G_{IntDep} = (G_1(\lambda_1, d_1), G_2(\lambda_2, d_2), d_{dep})$ *percolates, where* $p = 1 - e^{-\lambda_1 d_1^2/8}$, $p' = 1 - e^{-2D^2\lambda_2}$, $D = \min(d_2/\sqrt{10}, d_{dep}/\sqrt{5}), m = \lfloor 2D/d_1 \rfloor$, $d_1 \leq d_2$, *and* $d_{dep} \geq d_2/2$.

This upper bound is obtained by mapping G_{IntDep} to a dependent bond percolation model L_D. The mapping from the Poisson boolean model $G(\lambda, d)$ to L_D was first proposed in [69] to study the percolation threshold of $G(\lambda, d)$, and later applied to the study of a random geometric graph under non-uniform node removals [72]. We briefly describe the method in the previous literature that uses L_D to study the percolation of $G(\lambda, d)$, and then prove Theorem 4.2 based on a similar method.

1-dependent bond percolation model L_D

In the standard bond percolation model on a square lattice L, the event that a bond is open is independent of the event that any other bond is open. If in a square lattice L_D, the event that a bond is open may depend on the event that its adjacent bond is open, but is independent of the event that any non-adjacent bond is open, then L_D is a *1-dependent bond percolation model* on a square lattice. With the additional restriction that each bond is open with an identical probability, an upper bound on the percolation threshold of L_D is 0.8639 [69].

The 1-dependent bond percolation model L_D can be used to study the percolation of G' where the points are generated by homogeneous Poisson point processes. To construct a mapping from G' to L_D, consider two adjacent $D \times D$ squares S_1 and S_2 and let R be the rectangle formed by the two squares. A bond (v_1, v_2) that connects the centers of S_1 and S_2 is associated with R. Figure 4-4 illustrates the square lattice formed by the bonds, represented by thick line segments.

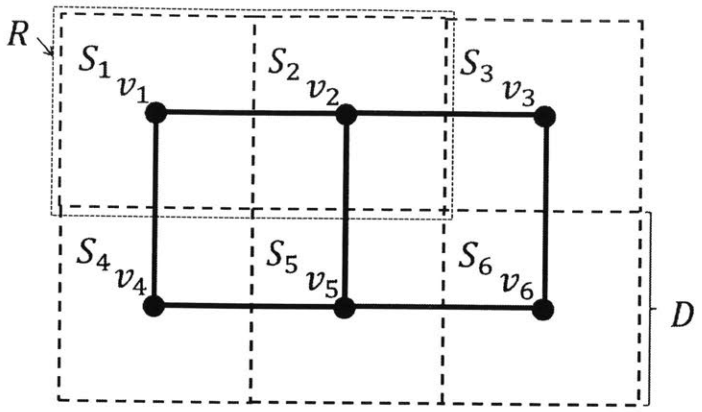


Figure 4-4: Square lattice L_D formed by the bonds (v_i, v_j).

Lemma 4.1. *Let the state of a bond (v_1, v_2) be determined by the homogeneous Poisson point processes of G' inside R, and the conditions for a bond to be open be identical for all bonds. Then the bonds form a 1-dependent bond percolation model L_D with identical bond open probabilities.*

Proof. The event that a bond is open is not independent of the event that its adjacent bond is open, since the two events both depend on the point process in an overlapping square. However, the event that a bond is open is independent of the event that any non-adjacent bond is open, since their associated rectangles do not overlap and the point processes in the two rectangles are independent.

Moreover, a Poisson point process is invariant under translation and rotation. Given that the points in G' are generated by homogeneous Poisson point processes and the conditions for a bond to be open are identical, the probability that a bond is open is identical for all bonds. □

By properly setting the conditions for a bond to be open, the percolation of L_D can imply the percolation of G'. We first look at an example in [71] that studies the percolation of $G(\lambda, d)$, and then extend the technique to study G_{IntDep}.

Example [71]: Let a bond be open if a path in $G(\lambda, d)$ crosses[1] R' horizontally

[1]A path crosses a rectangle $R' = [x_1, x_2] \times [y_1, y_2]$ horizontally if the path consists of a sequence of connected nodes $v_1, v_2, \ldots, v_{n-1}, v_n$, and $v_2, \ldots, v_{n-1}$ are in R', $x(v_1) \leq x_1, x(v_n) \geq x_2$, $y_1 \leq y(v_1), y(v_n) \leq y_2$, where $x(v_i)$ is the x-coordinate of v_i and $y(v_i)$ is the y-coordinate of v_i. A path crosses a rectangle vertically is defined analogously.

and another path in $G(\lambda, d)$ crosses S_1' vertically, where R' is a $(2D - 2d) \times (D - 2d)$ rectangle that has the same center as R, and S_1' is a $(D - 2d) \times (D - 2d)$ square that has the same center as S_1. The reason for considering R' and S_1' is that the existence of the two crossing paths over R' and S_1' is determined by the point process within R, while the existence of links within distance d from the boundaries (and thus the crossings over R) may depend on nodes outside R.

If two adjacent bonds are open, the paths in $G(\lambda, d)$ in the two rectangles are joined. To see this, note that in Fig. 4-5, if the black and blue bonds (same direction) are both open, the crossings 1 and 2 intersect. If the black and red bonds (perpendicular) are both open, the crossings 1 and 3 intersect.

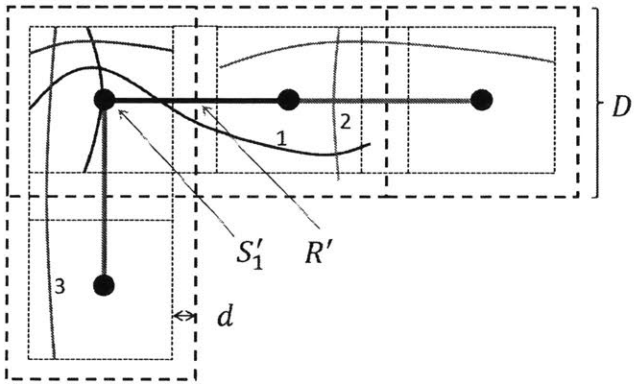


Figure 4-5: Crossings over rectangles associated with two adjacent open bonds are joined.

If the square lattice L_D percolates, open bonds form an infinite component. Paths in $G(\lambda, d)$ across the rectangles associated with the open bonds are connected and form an infinite component. Therefore, a node density above which L_D percolates is an upper bound on the percolation threshold of $G(\lambda, d)$.

Proof of Theorem 4.2

We map G_{IntDep} to L_D by letting a bond in L_D be open if the following three conditions are satisfied in its associated rectangle $R = S_1 \cup S_2$. The size of the rectangle satisfies $D = \min(d_2/\sqrt{10}, d_{\text{dep}}/\sqrt{5}) \geq d_2/2\sqrt{5}$.

1. A path from G_1 crosses R' horizontally, where R' is a $(2D - 2d_1) \times (D - 2d_1)$ rectangle that has the same center as R.

2. A path from G_1 crosses S_1' vertically, where S_1' is a $(D-2d_1) \times (D-2d_1)$ square that has the same center as S_1.

3. There is at least one node from G_2 in R.

To see that the percolation of L_D implies the percolation of G_{IntDep}, consider any two adjacent open bonds in L_D. In the two rectangles associated with the bonds, 1) paths from G_1 that cross one rectangle are joined with paths from G_1 that cross the other rectangle; 2) at least two nodes from G_2, one in each rectangle, are connected by a link in G_2, because any two nodes in adjacent rectangles are within distance $\sqrt{10}D \leq d_2$; 3) every node in G_i has at least one supply node in G_j inside the rectangle ($\forall i, j \in \{1,2\}, i \neq j$), in which the distance between two nodes is no larger than $\sqrt{5}D \leq d_{\text{dep}}$.

If the probability p_{123} that a bond is open is above 0.8639, then L_D percolates and G_{IntDep} also percolates. An upper bound on the percolation threshold of G_{IntDep} is a pair of node densities (λ_1, λ_2) that yields $p_{123} \geq 0.8639$. In the remainder of the proof, we compute p_{123} as a function of (λ_1, λ_2).

To determine the probability that the first and the second conditions are satisfied, we consider a discrete square lattice represented by Fig. 4-6. Bonds of length $d_1/2$ form a square lattice L' in a *finite* $md_1 \times md_1/2$ region, where $m = \lfloor 2D/d_1 \rfloor$. Let a bond in L' be open if there is at least one node from G_1 in the $d_1/2\sqrt{2} \times d_1/2\sqrt{2}$ square that contains the bond (the small square that has dashed boundaries in the figure), which occurs with probability $p = 1 - e^{-\lambda_1 d_1^2/8}$. It is clear that if the open bonds form a horizontal crossing[2] over L', then nodes in G_1 form a horizontal crossing path over R'.

Let $p_x(km, m, p)$ denote the probability that there exists a horizontal crossing

[2]A horizontal crossing of open bonds over a rectangle $R' = [x_1, x_2] \times [y_1, y_2]$ consists of a sequence of adjacent open bonds in the rectangle such that at least one bond has an endpoint with x-coordinate x_1 and at least one bond has an endpoint with x-coordinate x_2. A vertical crossing of open bonds is defined analogously.

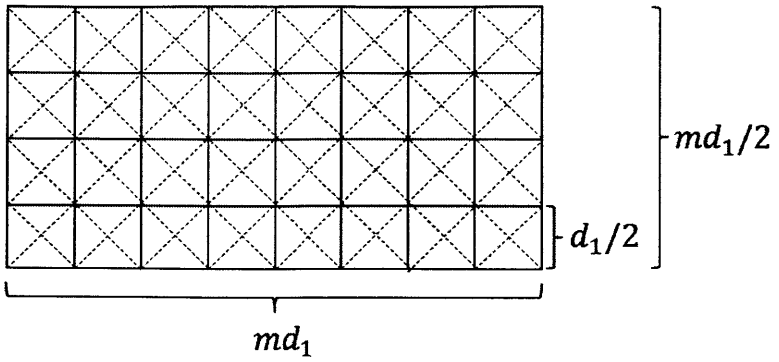


Figure 4-6: Mapping the crossing in G_1 to the crossing in a square lattice L'.

over the $km \times m$ square lattice L' given that each bond is open independently with probability p. A lower bound on $p_x(km, m, p)$, Eq. (4.2), can be derived by a standard technique in percolation theory (*e.g.*, an extension of Proposition 2 in [73]).

$$p_x(km, m, p) \geq 1 - \frac{4}{3}(km + 1)e^{m \log 3(1-p)}. \tag{4.2}$$

The probability that the crossing exists is close to 1 if m is large and $p > 2/3$.

Finally, the probability that the first condition is satisfied is $p_1 \geq p_x(2m, m, p)$. The probability that the second condition is satisfied is $p_2 \geq p_x(m, m, p)$. Given that the existence of the two crossings are positively correlated, by the FKG inequality [71], the probability that both conditions are satisfied is lower bounded by:

$$p_{12} \geq p_1 p_2 \geq p_x(2m, m, p)p_x(m, m, p).$$

The probability that there is at least one node from G_2 in R (*i.e.*, the third condition is satisfied) is $p_3 = 1 - e^{-2D^2\lambda_2}$. Given that the point processes in G_1 and G_2 are independent, the probability that a bond is open is $p_{123} = p_{12}p_3$. As long as $p_{123} > 0.8639$, G_{IntDep} percolates. This completes the proof.

An example of two RGGs with large d_2/d_1

We study two interdependent RGGs G_1 and G_2, which have a finite number of nodes, in order to quantify d_2/d_1 as a function of the number of nodes in the

graph. If $d_2 = \Omega(d_1 \log n_1)$, and $d_{\text{dep}} \geq d_2/2$, then $m = \Omega(\log n_1)$, where n_1 is the expected number of nodes in G_1. As n_1 approaches infinity, the probability $p_x(km, m, p)$ approaches 1 if $p > 2/3$, by Eq. (4.2).

Applying Theorem 4.2, by solving $p = (1 - e^{-\lambda_1 d_1^2/8}) = 2/3$, and $p_3 = 1 - e^{-2D^2\lambda_2} = 0.8639$, we obtain an upper bound on percolation threshold $\lambda_1 = 8.789/d_1^2$, $\lambda_2 = 19.94/d_2^2$. The bounds suggest that if the ratio between the connection distances of two RGGs is very large, the node density in one RGG may *not* affect the minimum node density in the other RGG in order for the giant mutual component to exist in the interdependent RGGs.

We conjecture that as long as the node density of each individual RGG is above the percolation threshold of the single graph, then the interdependent RGGs percolate, if $d_1 \ll d_2$ and $d_{\text{dep}} = (1 + \epsilon)d_2/2$ for $\epsilon > 0$. This can be intuitively explained as below. Let V_2^0 denote the nodes in the giant component of a single graph G_2 without considering the interdependence. Disks of radius $d_2/2$ centered at nodes in V_2^0 are connected. Disks of radius $d_{\text{dep}} > d_2/2$ centered at nodes in V_2^0 are also connected, and this region contains nodes in G_1 that have functional supply nodes. Each disk of radius d_{dep} is so large compared with d_1, that the probability that there is a crossing formed by connected nodes in G_1 along any direction across the disk approaches one[3]. Moreover, the disks of radius d_{dep} have overlaps with width and height at least $\epsilon d_2 \gg d_1$, which are sufficiently large to join the paths in G_1 across two overlapping disks. Thus, a giant component of G_1 exists near the giant component of G_2. Nodes in the two components are interdependent and form a giant mutual component.

4.2.4 Numerical results

We verify the bounds in Theorem 4.1 by simulating G_{IntDep} in a 10×10 square. Table 4.1 illustrates the fraction of nodes from G_i that belong to the largest mutual component, denoted by f_i, $(\forall i \in \{1, 2\})$. The fractions are averaged over 5 instances

[3]If nodes are generated by a Poisson point process with density above the percolation threshold, the probability that there is a horizontal path across a $kl \times l$ rectangle approaches one for any k as $l \to \infty$ [65].

of simulations for each combination of $(\lambda_1, \lambda_2, d_1, d_2, d_{\text{dep}})$ that satisfies the condition in Theorem 4.1. To verify the bounds in Theorem 4.2, we simulate G_{IntDep} in a 30×30 square (to simulate a sufficiently large G_2 under small node densities). Table 4.2 illustrates the average fraction of nodes in the largest mutual component, for $(\lambda_1, \lambda_2, d_1, d_2, d_{\text{dep}})$ given by Theorem 4.2. We observe that most nodes in G_1 and G_2 belong to the largest mutual component, which implies that G_{IntDep} percolates.

Table 4.1: Fraction of nodes in the largest mutual component under the condition of Theorem 4.1

λ_1	λ_2	d_1	d_2	d_{dep}	f_1	f_2
15	1.54	1	3	1.5	1.00	1.00
20	0.92	1	3	1.5	0.99	1.00
25	0.75	1	3	1.5	0.98	1.00
15	2.39	1	2	1	0.99	1.00
20	1.80	1	2	1	1.00	1.00
25	1.58	1	2	1	0.97	1.00

Table 4.2: Fraction of nodes in the largest mutual component under the condition of Theorem 4.2

λ_1	λ_2	d_1	d_2	d_{dep}	f_1	f_2
16	0.190	1	10	7.07	1.00	1.00
17	0.123	1	10	7.07	1.00	1.00
25	0.100	1	10	7.07	1.00	1.00
17	0.385	1	8	5.66	1.00	1.00
18	0.207	1	8	5.66	1.00	1.00
25	0.156	1	8	5.66	0.99	1.00

Remark: We have assumed that $d_{\text{dep}} \geq \max(d_1/2, d_2/2) = d_2/2$ throughout this section. To see that this is a reasonable assumption, note that nodes in G_1 that have at least one functional supply node are restricted in the region R_{dep}, where R_{dep} is the union of disks with radius d_{dep} centered at nodes in the infinite component of G_2. If R_{dep} is fragmented, it is not likely for disks of radius $d_1/2 < d_2/2$ centered at random locations within R_{dep} to overlap, and it is not likely that a functional infinite component will exist in G_1, unless the node density in G_1 is large. Therefore, the interdependent distance d_{dep} should be large enough so that R_{dep} is a connected region, to avoid a large minimum node density in G_1. The region R_{dep} can be made

larger by increasing either λ_2 or d_{dep}. Setting $d_{\text{dep}} \geq d_2/2$ avoids increasing λ_2 high above the percolation threshold of G_2, in order for R_{dep} to be connected. In Section 4.3, we develop a more general approach that does not require this assumption.

4.3 Confidence intervals for percolation thresholds

In this section, we compute confidence intervals for percolation thresholds. The confidence intervals provide interval estimates for the percolation thresholds. If the node densities in G_{IntDep} are below the lower confidence bounds, then there does not exist an infinite mutual component in G_{IntDep} with high confidence. On the other hand, if the node densities are above the upper confidence bounds, then there exists an infinite mutual component in G_{IntDep} with high confidence. Compared with the analytical upper bounds in Section 4.2, the numerical upper confidence bounds are much tighter. Moreover, the techniques in this section apply to G_{IntDep} with general d_1, d_2, d_{dep}.

The mapping to compute confidence intervals is related to the mapping from G_{IntDep} to the 1-dependent bond percolation model L_D in Section 4.2.3. Both mappings satisfy the following properties: 1) the percolation of L_D implies the percolation of G_{IntDep}; 2) the event that determines the state of a bond depends only on the point process within its associated rectangle, thus preserving the 1-dependent property. The probability that the event occurs can be computed or bounded analytically in the previous section. In contrast, in this section, we consider events whose probabilities are larger under the same point processes but can only be evaluated by simulation. Since the events that we consider in this section are more likely to occur under the same point processes, the mappings yield tighter bounds.

Our mappings from G_{IntDep} to L_D extend the mappings from $G(\lambda, d)$ to L_D proposed in [69]. For completeness, we first briefly summarize the mappings in [69] that compute upper and lower bounds on the percolation threshold of $G(\lambda, d)$.

Upper bound for $G(\lambda, d)$ [69]: Recall Fig. 4-4. The event that a bond $(v_1, v_2) \in L_D$ is open is determined by the point process of $G(\lambda, d)$ in the rectangle $R = S_1 \cup S_2$,

where S_1 and S_2 are squares. Let V_i denote the largest component formed by the points of $G(\lambda, d)$ in S_i. If V_i is the *unique* largest component in S_i ($\forall i \in \{1, 2\}$) and V_1 and V_2 are connected, then the bond is open. Otherwise, the bond is closed.

If L_D percolates, open bonds form an infinite component. As a result, the largest components in the squares that intersect the open bonds are connected in $G(\lambda, d)$ and they form an infinite component. Therefore, a node density λ, above which the probability that a bond is open is larger than 0.8639, is an upper bound on the percolation threshold of $G(\lambda, d)$.

Lower bound for $G(\lambda, d)$ [69]: Let the *connection process* of $G(\lambda, d)$ be the union of nodes and links in $G(\lambda, d)$. Let the *complement* of the connection process be the union of the empty space that does not intersect nodes or links. If the complement of the connection process form a connected infinite region, then all the connected components in $G(\lambda, d)$ have finite sizes and $G(\lambda, d)$ does not percolate [69, 74]. Consider the complement of the connection process in rectangle R. Let a bond (in L_D) associated with rectangle R be open if the complement process forms a horizontal crossing[4] over the rectangle R' and a vertical crossing over the square S_1'. Recall that rectangle R' is the $(2D - 2d) \times (D - 2d)$ rectangle that has the same center as R, and square S_1' is the $(D - 2d) \times (D - 2d)$ square that has the same center as S_1, the left square in R. For example, in Fig. 4-7, the two crossings that do not intersect any nodes or links are plotted.

If L_D percolates, the complement process forms an infinite region and $G(\lambda, d)$ does not percolate. To conclude, a node density, under which the probability that the complement process forms the two crossings is above 0.8639, is a lower bound on the percolation threshold for $G(\lambda, d)$.

[4]The complement of a connection process forms a horizontal crossing over a rectangle if a curve in the rectangle touches the left and right boundaries of the rectangle and the curve does not intersect any nodes or links. The vertical crossing of the complement process is defined analogously.

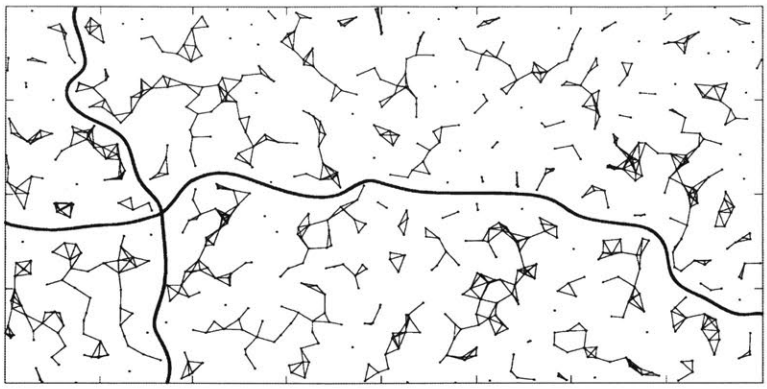


Figure 4-7: The horizontal and vertical crossings from the complement of the connection process over the rectangle.

4.3.1 Upper bounds for G_{IntDep}

In $G(\lambda, d)$, the largest connected component that contains a node b can be computed efficiently by contracting the links (or using a breadth-first-search) starting from b. Two components are connected and form one component if there exists two nodes within distance d, one in each component. We next extend these notions to G_{IntDep}.

Let G_1 and G_2 denote the two graphs in G_{IntDep}. Let $b_1 \in G_1$ and $b_2 \in G_2$ denote two nodes within the interdependent distance d_{dep}. Algorithm 4.1 computes the largest mutual component $M(b_1, b_2)$ that contains b_1 and b_2. The correctness follows from the definition of mutual component.

Algorithm 4.1 Computing the largest mutual component that contains two specified nodes $b_i \in G_i$ within d_{dep} $(\forall i \in \{1, 2\})$.

---

1. Find all the nodes $V_i^0(b_i)$ that are connected to b_i (either directly or through a sequence of links) in G_i $(\forall i \in \{1, 2\})$.

2. Remove nodes in $V_i^0(b_i)$ that do not have any supply nodes in $V_j^0(b_j)$ $(\forall i, j \in \{1, 2\}, i \neq j)$. Among the remaining nodes, find the nodes $V_i^1(b_i) \subseteq V_i^0(b_i)$ that are connected to b_i $(\forall i \in \{1, 2\})$.

3. Repeat step 2 until $V_i^{k+1}(b_i) = V_i^k(b_i)$ $(\forall i \in \{1, 2\})$. Let $M(b_1, b_2) = V_1^k(b_1) \cup V_2^k(b_2)$.

Two mutual components $M = V_1 \cup V_2$ and $\hat{M} = \hat{V}_1 \cup \hat{V}_2$ form one mutual component

if and only if V_i and $\hat{V}_i$ are connected in G_i ($\forall i \in \{1,2\}$). The necessity of the condition is obvious. To see that this condition is sufficient, note that every node in the connected component formed by V_i and $\hat{V}_i$ has at least one supply node that belongs to the connected component formed by V_j and $\hat{V}_j$ ($\forall i,j \in \{1,2\}, i \neq j$). The condition can be generalized naturally for more than two mutual components to form one mutual component.

The method of obtaining an upper bound on the percolation threshold of $G(\lambda,d)$ can be modified to obtain an upper bound on the percolation threshold of G_{IntDep}, by declaring a bond to be open if the unique largest mutual components in the two adjacent $D \times D$ squares S_1 and S_2 are connected. However, computing the largest mutual component of G_{IntDep} in S_i is not as straightforward as computing the largest component of $G(\lambda,d)$ in S_i. In $G(\lambda,d)$, a node belongs to exactly one (maximal) connected component. All the components can be obtained by contracting the links, and the largest component can be obtained by comparing the sizes of the components. However, in G_{IntDep}, a node may belong to multiple mutual components. For example, let b_1 and b_2 be two isolated nodes in G_1, and let b_3 and b_4 be two connected nodes in G_2. If both b_1 and b_2 are within the interdependent distance from b_3 and b_4, $\{b_1,b_3,b_4\}$ and $\{b_2,b_3,b_4\}$ are two mutual components. An algorithm that computes the largest mutual component of G_{IntDep} in a square 1) selects a pair of nodes, one from each graph, and computes the largest mutual component that contains the two nodes by Algorithm 4.1, and then 2) chooses the largest mutual component over all pairs of nodes in the square within the interdependent distance. Thus, it requires much more computation than finding the largest component of $G(\lambda,d)$ in a square.

Instead of optimizing the algorithm and obtaining the largest mutual component in square S, a mutual component $M^{\text{greedy}}(S)$ can be computed by Algorithm 4.2. This algorithm has good performance in finding a large mutual component when the square size is large. In particular, if the square had infinite size, this algorithm would find an infinite mutual component if one exists.

Let a bond (v_1,v_2) in L_D be open if the two components $M^{\text{greedy}}(S_1)$ and $M^{\text{greedy}}(S_2)$ form one mutual component. Since $M^{\text{greedy}}(S_i)$ is unique in any square S_i, a connected

Algorithm 4.2 An algorithm that greedily computes a mutual component $M^{\text{greedy}}(S)$ in region S.

1. Find the largest connected component $V_i^0(S)$ in $G_i(S)$, where $G_i(S)$ consists of the nodes and links of G_i in region S. If there is more than one largest connected component, apply any deterministic tie-breaking rule (*e.g.*, choose the component that contains a nodes with the smallest x-coordinate).
2. Remove nodes in $V_i^0(S)$ that do not have supply nodes in $V_j^0(S)$ $(\forall i, j \in \{1,2\}, i \neq j)$. Find the largest connected component $V_i^1(S)$ formed by the remaining nodes in $V_i^0(S)$ $(\forall i \in \{1,2\})$, and apply the same tie-breaking rule.
3. Repeat step 2 until $V_i^{k+1}(S) = V_i^k(S)$ $(\forall i \in \{1,2\})$. Let $M^{\text{greedy}}(S) = V_1^k(S) \cup V_2^k(S)$.

component in L_D implies that $\{M^{\text{greedy}}(S_i)\}$ form one mutual component in G_{IntDep}, where S_i are the squares that intersect the open bonds in the connected component in L_D. If the probability that a bond is open is larger than 0.8639, L_D percolates and G_{IntDep} also percolates.

An alternative condition for a bond to be open is that nodes in $M^{\text{greedy}}(R)$ form a horizontal crossing over rectangle R' and a vertical crossing over square S_1' in both graphs (recall Fig. 4-5 and the condition for two mutual components to form one mutual component). In order for the existence of the two crossings to only depend on the point processes in R, in the definition of the $(2D - 2d) \times (D - 2d)$ rectangle R' and the $(D - 2d) \times (D - 2d)$ square S_1', $d = \max(d_1, d_2) + d_{\text{dep}}$.

An upper bound on the percolation threshold can be obtained by either approach. The smaller bound obtained by the two approaches is a better upper bound on the percolation threshold for G_{IntDep}.

4.3.2 Lower bounds for G_{IntDep}

In G_{IntDep}, the connection process consists of nodes and links in mutual components. To avoid the heavy computation of mutual components, we study another

model in which the connection process $\tilde{P}_i$ of G_i in the new model *dominates*[5] the connection process P_i of G_i in G_{IntDep} ($\forall i \in \{1,2\}$). As a consequence, the complement of the connection process $\tilde{P}_i^c$ of G_i in the new model is dominated by P_i^c ($\forall i \in \{1,2\}$). If $\tilde{P}_i^c$ percolates, then P_i^c percolates and P_i does not percolate (*i.e.*, all the components in P_i have finite sizes). If either P_1 or P_2 does not percolate, then G_{IntDep} does not percolate. Thus, node densities under which at least one of $\tilde{P}_1^c$ and $\tilde{P}_2^c$ percolates are lower bounds on the percolation thresholds of G_{IntDep}.

The new model can be viewed to have a *relaxed* supply requirement. In this model, every node (as opposed to nodes in the same mutual component) is viewed as a valid supply node for nodes in the other graph. A node b_i in G_i is removed if and only if there is no node in G_j within the interdependent distance d_{dep} from b_i ($\forall i,j \in \{1,2\}, i \neq j$). After all such nodes are removed, the remaining nodes in G_i are connected if their distances are within the connection distance d_i. The computation of the connection process $\tilde{P}_i$ is efficient and avoids the computation of mutual components in G_{IntDep} through multiple iterations.

The connection process $\tilde{P}_i$ in the new model dominates P_i in the original model G_{IntDep}. On the one hand, for any realization, all the links in P_i are present in $\tilde{P}_i$, because all the nodes in a mutual component have supply nodes, and links between these nodes are present in the new model as well. On the other hand, in the new model, nodes in a connected component $\tilde{V}_i$ in G_i may depend on nodes in multiple components in G_j. In contrast, in G_{IntDep}, the nodes in $\tilde{V}_i$ may be divided into several mutual components, and links do not exist between two disjoint mutual components.

An algorithm that computes a lower bound on the percolation threshold of G_{IntDep} is as follows. First, compute the connection process $\tilde{P}_i$ in the new model. Next, in the $2D \times D$ rectangle R, consider the complement of the connection process $\tilde{P}_i^c$. Let p_i denote the probability that there is a horizontal crossing over R' and a vertical crossing over S_1' in the complement process $\tilde{P}_i^c$, where R' and S_1' are the same as before. A lower bound on the percolation threshold of G_{IntDep} is given by node densities under

[5]One connection process dominates another if the nodes and links in the first process form a superset of the nodes and links in the second process, for any realization of G_i.

which $\max(p_1, p_2) \geq 0.8639$.

4.3.3 Confidence intervals

The probability that a bond is open can be represented by an integral that depends on the point processes in the rectangle R. However, direct calculation of the integral is intractable; so instead the integral is evaluated by simulation. In every trial of the simulation, nodes in G_1 and G_2 are randomly generated by the Poisson point processes with densities λ_1 and λ_2, respectively. The events that a bond is open are independent in different trials. Let the probability that a bond is open be p given (λ_1, λ_2). The probability that a bond is closed in k out of N trials follows a binomial distribution. The interval $[0.8639, 1]$ is a 99.5% confidence interval [75] for p, given that $N = 100$ and $k = 5$. If $k < 5$, $p \in [0.8639, 1]$ with a higher confidence. This suggests that if $k \leq 5$, with 99.5% confidence, $p \geq 0.8639$ and the 1-dependent bond percolation model L_D percolates given (λ_1, λ_2).

Based on this method, with 99.5% confidence, an upper bound on the percolation threshold of G_{IntDep} can be obtained by declaring a bond to be open using the method in Section 4.3.1, and a lower bound can be obtained by declaring a bond to be open using the method in Section 4.3.2. For a fixed λ_2^*, a 99% confidence interval for λ_1^* is given by the interval between the upper and lower bounds. Confidence intervals for different percolation thresholds can be obtained by changing the value of λ_2^* and repeating the computation. We make a similar remark as in [69]. The confidence intervals are rigorous, and the only uncertainty is caused by the stochastic point processes in the $2D \times D$ rectangle. This is in contrast with the confidence intervals obtained by estimating whether G_{IntDep} percolates based on extrapolating the observations of simulations in a finite region (which is usually not very large because of limited computational power).

4.3.4 Numerical results

The simulation-based confidence intervals are much tighter than the analytical bounds. Given that $d_1 = d_2 = 2d_{\text{dep}} = 1$, and $\lambda_2^* = 2$, the upper and lower bounds on λ_1^* are 2.25 and 1.80, respectively, both with 99.5% confidence. In contrast, even if $\lambda_2^* \to \infty$, the analytical upper bound on λ_1^* is no less than 3.372, which is the best available analytical upper bound for a single G_1 [68]. Confidence intervals for the percolation thresholds are plotted in Fig. 4-8, where the intervals between bars are 99% confidence intervals.

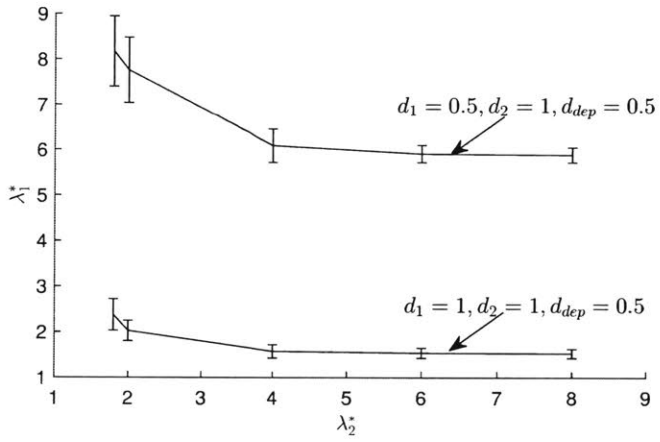


Figure 4-8: The 99% confidence intervals for percolation thresholds of G_{IntDep} with different connection distances.

To verify the confidence intervals, we simulate G_{IntDep} within a 20×20 square, for $d_1 = d_2 = 2d_{\text{dep}} = 1$. Nodes in the largest mutual component are colored black, while the remaining nodes are colored blue. In Fig. 4-9, the node densities are at the upper confidence bound ($\lambda_1 = 2.25, \lambda_2 = 2.00$), and there exists a mutual component that consists of a large fraction of nodes. In Fig. 4-10, the node densities are at the lower confidence bound ($\lambda_1 = 1.80, \lambda_2 = 2.00$), and the size of the largest mutual component is small.

We next study the impact of interdependent distance d_{dep} on the percolation thresholds. Given d_1, d_2, λ_2^*, a smaller d_{dep} leads to a higher λ_1^*, since the probability that a node in G_1 has at least one supply nodes from G_2 decreases for a smaller

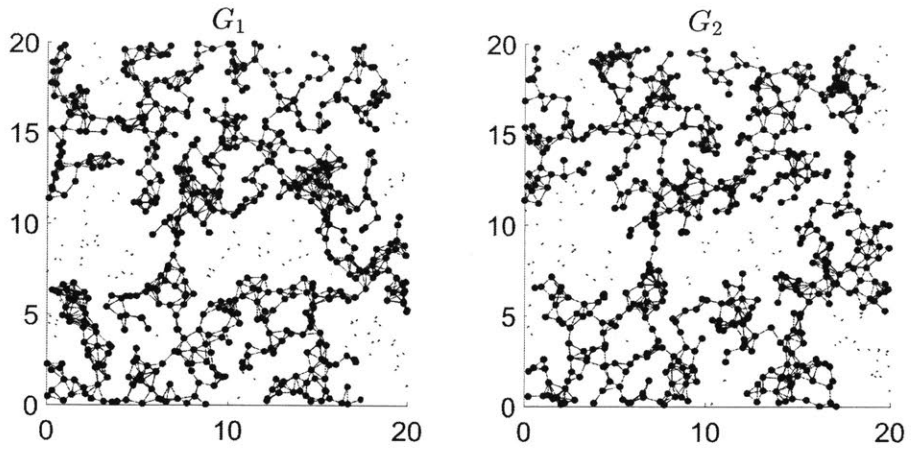


Figure 4-9: The largest mutual component for $\lambda_1 = 2.25, \lambda_2 = 2.00, d_1 = d_2 = 2d_{\text{dep}} = 1$.

d_{dep}. The effect is more significant when the number of nodes in G_2 is small. This is consistent with Fig. 4-11, where the increase of λ_1^* is more significant as d_{dep} decreases when λ_2^* is small.

The confidence intervals confirm that the reduced node density due to a lack of supply nodes is not sufficient to characterize the percolation of one of the interdependent graphs. The average density of nodes in G_1 that have at least one node within d_{dep} in G_2 is $\tilde{\lambda}_1 = \lambda_1(1 - e^{-\lambda_2 \pi d_{\text{dep}}^2})$, given that $e^{-\lambda_2 \pi d_{\text{dep}}^2}$ is the probability that there is no node in G_2 within a disk area πd_{dep}^2. If $\lambda_2^* = 1.8$, with 99% confidence, $\lambda_1^* \in [2.03, 2.72]$ when $d_{\text{dep}} = 0.5$, and $\lambda_1^* \in [7.50, 11.20]$ when $d_{\text{dep}} = 0.25$. We observe that the ranges of $\tilde{\lambda}_1^*$ are different: $\tilde{\lambda}_1^* \in [1.54, 2.06]$ when $d_{\text{dep}} = 0.5$, and $\tilde{\lambda}_1^* \in [2.23, 3.33]$ when $d_{\text{dep}} = 0.25$. Intuitively, nodes in G_1 that have at least one supply nodes are clustered around the nodes in G_2, smaller d_{dep} leads to a more clustered point process. The critical node density of a clustered point process is not the same as the critical node density of the homogeneous Poisson point process for percolation. More detailed study on the percolation of a clustered point process can be found in [76].

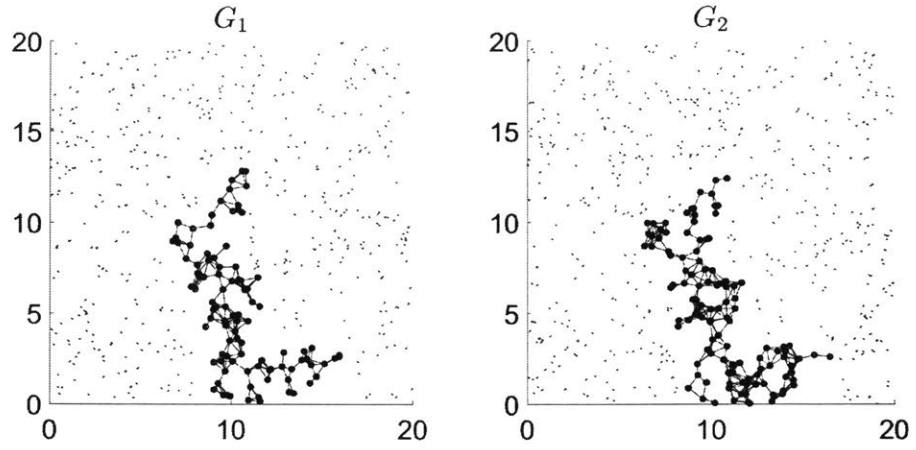


Figure 4-10: The largest mutual component for $\lambda_1 = 1.80, \lambda_2 = 2.00, d_1 = d_2 = 2d_{\text{dep}} = 1$.

4.4 Robustness of interdependent RGGs under random and geographical failures

Removing nodes independently at random with the same probability in one graph is equivalent to reducing the node density of the Poisson point process. To study the robustness of G_{IntDep} under random failures, the first step is to obtain the upper and lower bounds on percolation thresholds. With the bounds, we can determine which graph is able to resist more random node removals, by comparing the gap between the node density λ_i and the percolation threshold λ_i^* given λ_j $(i, j \in \{1, 2\}, i \neq j)$. The graph that can resist a smaller fraction of node removals is the bottleneck for the robustness of G_{IntDep}. Moreover, we are able to compute the maximum fraction of nodes that can be randomly removed from two graphs while guaranteeing G_{IntDep} to be percolated.

We next show that G_{IntDep} still percolates after a geographical attack that removes nodes in a finite connected region, if the node densities of the two graphs before the attack are above any *upper bound* on the percolation thresholds obtained in this chapter (either analytical or simulation-based). Recall that we obtained upper bounds on the percolation thresholds of G_{IntDep} by mapping the percolation of G_{IntDep} to

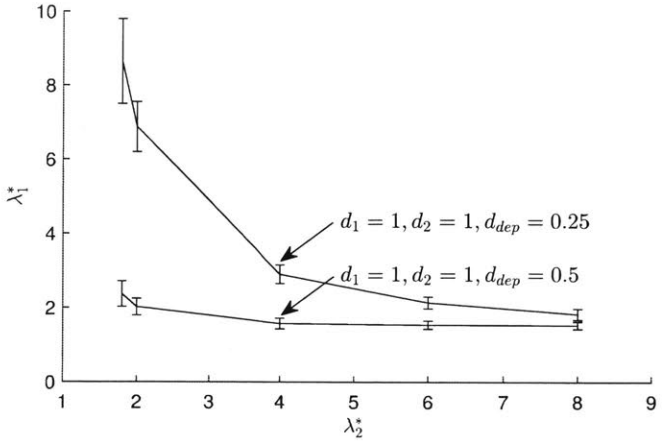


Figure 4-11: The 99% Confidence intervals for percolation thresholds of two G_{IntDep} with different interdependence distances.

either the independent bond percolation on a square lattice L or the 1-dependent bond percolation on a square lattice L_D. Under both mappings, the event that a bond e is open is entirely determined by the point processes in a finite region R_e that contains the bond. After removing nodes of G_{IntDep} in a connected finite geographical region, the state of a bond e may change from open to closed only if R_e intersects the attack region. Let R_f be the union of R_e that intersects the attack region. The region R_f is also a connected finite region. As long as L or L_D still percolates after setting bonds in R_f to be closed, G_{IntDep} percolates.

Results from the percolation theory indeed indicate that setting all the bonds in a finite region R_f to be closed does not affect the percolation of L or L_D. For any percolated L, the probability that there exists a horizontal crossing of open bonds over a $kl \times l$ rectangle approaches 1 for any integer $k > 1$, as $l \to \infty$ (Lemma 8 on Page 64 of [71]). The percolation of L (after setting all bonds in R_f to be closed) is justified by the fact that the connected open bonds across rectangles form a square annulus that does not intersect R_f (shown in Fig. 4-12), which is a standard approach to prove the percolation of L [71]. Moreover, the percolation of L_D after all bonds in R_f are set closed can be proved in the same approach, by noting that the probability that open bonds of L_D form a horizontal crossing over a rectangle approaches 1 as the rectangle size increases to infinity [69].

If the $kl \times l$ rectangle is large but finite, the probability that a horizontal crossing formed by open bonds exists is close to 1 if L or L_D percolates. Therefore, the same analysis demonstrates the robustness of two finite interdependent RGGs under a geographical attack that removes the nodes in a disk region of size βa^2, where $0 < \beta < 1$.

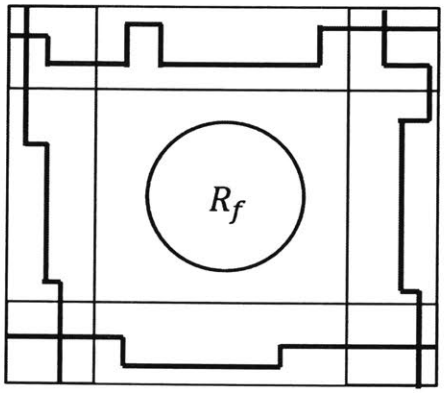


Figure 4-12: Open bonds form a connected path across rectangles around R_f.

The robustness of interdependent RGGs under geographical failures is illustrated in Fig. 4-13. Nodes and links in the giant mutual component are colored black. The interdependent RGGs still percolate after all the nodes in a disk region are removed. This is in contrast with the cascading failures observed in [13] in the interdependent lattice model after an initial disk attack. One reason may be that every node can have more than one supply node in our model, while every node has only one supply node in [13]. The multiple localized interdependence helps the interdependent RGGs to resist geographical attacks.

4.5 Extensions to more general interdependence

In the previous sections, we studied a model where every node in G_i is content to have at least one supply node in G_j in the same mutual component ($\forall i, j \in \{1, 2\}, i \neq j$). The techniques can be extended to study models where every node in G_i must have at least K_j supply nodes from G_j to receive enough supply, where K_j can be either a constant or a random variable ($\forall i, j \in \{1, 2\}, i \neq j$). We briefly discuss

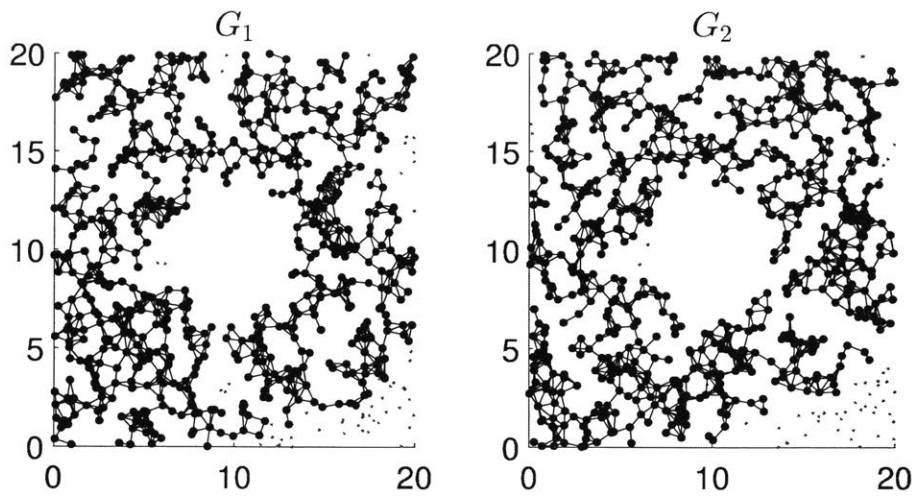


Figure 4-13: Interdependent RGGs with the same connection distance $d_1 = d_2 = 1$ and $d_{\text{dep}} = 0.5$.

the extensions to models with more general supply requirement using the example in Section 4.2.1, where $d_1 = d_2 = 2d_{\text{dep}}$.

4.5.1 Deterministic supply requirement

The extension is straightforward if K_i is a constant, $\forall i \in \{1, 2\}$. By the same discretization technique, the state of a site in the triangle lattice is determined by the point processes in a cell of area A (recall Fig. 4-2). Declare a site to be open if there are at least K_i nodes from G_i in the cell that contains the site $(\forall i, j \in \{1, 2\}, i \neq j)$. For each open site, every node from G_i in the cell has at least K_j supply nodes from G_j in the same cell, satisfying the supply requirement. Following the same analysis as that in Section 4.2.1, the percolation of the triangle lattice implies the percolation of G_{IntDep}.

For a Poisson point process of density λ_j, the probability that there are at least K_j nodes in a cell of area A is $1 - \sum_{l=0}^{K_j-1} (\lambda_j A)^l e^{-\lambda_j A}/l!$. An upper bound on the percolation thresholds is given by (λ_1, λ_2) that satisfies:

$$\left[1 - \sum_{l=0}^{K_1-1} \frac{(\lambda_1 A)^l e^{-\lambda_1 A}}{l!}\right] \left[1 - \sum_{l=0}^{K_2-1} \frac{(\lambda_2 A)^l e^{-\lambda_2 A}}{l!}\right] = \frac{1}{2}.$$

4.5.2 Random supply requirement

Some extra work is necessary if K_i is a random variable, $\exists i \in \{1, 2\}$. For simplicity, we first consider the case where $K_1 \geq 1$ is a constant and K_2 is a discrete random variable with a cumulative distribution function $F_{K_2}(x)$, $x \in \mathbb{N}$. Furthermore, we assume that the number of supply nodes needed by every node in G_1 is independent. After the discretization, a site in the triangle lattice is open if the following two conditions are satisfied for at least one integer-valued $k_2 \geq 1$.

1. There are exactly k_2 nodes from G_2 in the cell.
2. There are at least K_1 nodes from G_1 in the cell, each of which needs no more than k_2 supply nodes.

If both conditions are satisfied, at least K_1 nodes from G_1 and the k_2 nodes from G_2 each have enough supply. It is easy to see that the percolation of the triangle lattice still implies the percolation of G_{IntDep}.

Next we compute the probability that the two conditions are satisfied. The probability that there are k_2 nodes from G_2 in the cell is:

$$\Pr(N_2 = k_2) = (\lambda_2 A)^{k_2} e^{-\lambda_2 A} / k_2!.$$

The probability that there are l nodes from G_1 in the cell is:

$$\Pr(N_1 = l) = (\lambda_1 A)^{l} e^{-\lambda_1 A} / l!.$$

The probability that a node in G_1 needs no more than k_2 supply nodes is $F_{K_2}(k_2)$. Since the number of supply nodes needed by every node in G_1 is independent, the probability that at least K_1 out of the l nodes in G_1 each need no more than k_2 supply nodes is:

$$\Pr(K_2^{(K_1)} \leq k_2 | N_1 = l) = \sum_{t=K_1}^{l} \binom{l}{t} [F_{K_2}(k_2)]^t [1 - F_{K_2}(k_2)]^{l-t},$$

for $K_1 \leq l$, and $\Pr(K_2^{(K_1)} \leq k_2 | N_1 = l) = 0$ for $K_1 > l$. By the law of total probability, for a given k_2, the probability that there exist at least K_1 nodes from G_1 in the cell that each need no more than k_2 supply nodes is:

$$\Pr(K_2^{(K_1)} \leq k_2) = \sum_{l \geq K_1} \Pr(N_1 = l) \Pr(K_2^{(K_1)} \leq k_2 | N_1 = l).$$

Since the events that there are exactly k_2 nodes from G_2 in the cell are mutually exclusive for distinct values of k_2, by the law of total probability, the probability that both conditions are satisfied is:

$$p_{12} = \sum_{k_2 \geq 1} \Pr(N_2 = k_2) \Pr(K_2^{(K_1)} \leq k_2).$$

Any (λ_1, λ_2) that satisfies $p_{12} \geq 1/2$ is an upper bound on the percolation threshold of G_{IntDep}.

Finally, we consider the case where both K_1 and K_2 are discrete random variables. Suppose that N_i nodes from G_i are in the cell of area A. If there exist integers $k_i^* \leq N_i$, such that at least k_i^* nodes from G_i each need no more than k_j^* supply nodes, then the k_i^* nodes from G_i all have enough supply $(\forall i, j \in \{1, 2\}, i \neq j)$. However, it is difficult to obtain a clean formula of the probability that (k_1^*, k_2^*) exists (to satisfy the condition). The events that (k_1^*, k_2^*) exists are not mutually exclusive for distinct values of k_1^* and k_2^*. While it is possible to compute this probability using the inclusion-exclusion formula, the computation is expensive, since the number of choices of (k_1^*, k_2^*) can be large and each term in the inclusion-exclusion formula requires the computation of order statistics.

A practical approach to estimate the probability that nodes have enough supply is by simulation. In each trial of the simulation, N_i nodes are randomly generated in area A, where N_i follows a Poisson distribution of rate $\lambda_i A$ $(\forall i \in \{1, 2\})$. Then, each of the N_i nodes is tagged with a realization of the random variable K_j, which indicates the number of required supply nodes $(\forall i, j \in \{1, 2\}, i \neq j)$. Let I indicate whether there exist (k_1^*, k_2^*) such that at least k_i^* nodes among the N_i nodes all have

tags no more than k_j^* ($\forall i, j \in \{1,2\}, i \neq j$). The value of I can be computed by Algorithm 4.3.

Algorithm 4.3 An algorithm that determines whether nodes have enough supply.

Initialization:

Sort the N_i realizations of the random variable K_j in the ascending order. Let $K_j^{(t)}, t = 1, \ldots, N_i$ be the sorted list ($\forall i, j \in \{1,2\}, i \neq j$). Let $t_1 = t_2 = 1$.

Main loop:

while I is not determined **do**
  $t_1' \leftarrow K_1^{(t_2)}$, $t_2' \leftarrow K_2^{(t_1)}$.
  if $t_1' \leq t_1$ and $t_2' \leq t_2$ **then**
    $I \leftarrow 1$.
  end if
  if $t_1' > N_1$ or $t_2' > N_2$ **then**
    $I \leftarrow 0$.
  end if
  $t_1 \leftarrow \max(t_1, t_1')$, $t_2 \leftarrow \max(t_2, t_2')$.
end while

We now prove the correctness of Algorithm 4.3. For easy presentation, the N_i nodes are referred to as nodes in G_i ($\forall i \in \{1,2\}$). Initially, among the nodes in G_i, the algorithm chooses one node that needs the smallest number of supply nodes. To support this node, at least $t_j' = K_j^{(1)}$ nodes need to be in G_j. If $t_1' \leq 1$ and $t_2' \leq 1$, one node from G_1 and one node from G_2 suffice to support each other. Otherwise, if $t_j' > 1$, at least t_j' nodes need to be in G_j. The t_j' nodes must be supported by $K_i^{(t_j')}$ nodes from G_i. If $K_i^{(t_j')}$ is larger than the total number of nodes in G_i, then there are not enough supporting nodes in G_i and $I = 0$. If $K_1^{(t_2')} \leq t_1'$ and $K_2^{(t_1')} \leq t_2'$, then t_1' nodes from G_1 support t_2' nodes from G_2, and vise versa. Note that t_1' and t_2' never decrease in the iterations, and at least one of them strictly increases in an iteration where I is not determined. If there exists at least one pair (k_1^*, k_2^*), the algorithm terminates with $I = 1$ at the smallest pair for both coordinates, which can be shown by contradiction. If no such pair (k_1^*, k_2^*) exists, the algorithm terminates with $I = 0$.

Given (λ_1, λ_2), by repeating a sufficiently large number of trials, the probability that $I = 1$ can be estimated within a small multiplicative error with high confidence using Monte Carlo simulation. As long as this probability is at least $1/2$, G_{IntDep} percolates with high confidence.

4.6 Summary

We developed an interdependent RGG model for interdependent spatially embedded networks. We obtained analytical upper bounds and confidence intervals for the percolation thresholds. The percolation thresholds of two interdependent RGGs form a curve, which shows the tradeoff between the two node densities in order for the interdependent RGGs to percolate. The curve can be used to study the robustness of interdependent RGGs to random failures. Moreover, if the node densities are above any upper bound on the percolation thresholds obtained in this chapter, then the interdependent RGGs remain percolated after a geographical attack. Finally, we extended the techniques to models with more general interdependence. The study of percolation thresholds can be used to design robust interdependent networks.

Chapter 5

Power grid frequency control using limited communication

Smart grid is one of the most important applications of interdependent networks. Communication network collects measurement data in power grid for the control center, and delivers control messages to power generators. Loss of communication leads to inefficient control on the power grid, and can lead to cascading failures and blackouts. In this chapter, we study power grid frequency control using limited communication.

The integrations of renewable energy resources increase the fluctuations of power supply. To balance the power supply and demand, power generations are controlled using primary, secondary, and tertiary controls under different time scales. The primary control at a power generator, droop control, responds to power flow perturbations within milliseconds to seconds, and re-balances the power supply and demand at the cost of non-zero frequency deviation. The secondary control, Automatic Generation Control (AGC), adjusts generator setpoints to recover the nominal frequency in seconds to minutes. The tertiary control, economic dispatch, minimizes the total power generation cost by scheduling an operating point for each generator, and operates in minutes to an hour.

Communication is essential for frequency regulation and economic dispatch. Both the AGC and the economic dispatch are traditionally implemented using centralized

control. The control center gathers information from all generators and loads, and computes setpoints for generators to adjust to disturbances. Both the information aggregation and setpoints delivery require communication between the control center and controllable nodes.

There have been recent advancement in developing distributed and decentralized frequency control techniques [77–79]. Motivated by the need to adapt to more frequent power fluctuations and faster response, some of these controllers require communication between neighbor nodes, to achieve the objectives of frequency control for power grids with renewable integrations.

There are two major categories of distributed and decentralized frequency control – *primal-dual controller* and *integral controller*. By formulating the frequency control as a convex optimization problem, a primal-dual algorithm was developed in [80] for joint frequency regulation and economic dispatch. The primal-dual controller was extended to handle power transmission line thermal limits and inter-area flow constraints in [81]. By considering frequency regulation and economic dispatch in different time scales, a primal-dual controller under stochastic power demand was developed in [82]. For these primal-dual controllers, communication between adjacent nodes is required to transmit Lagrangian multipliers. Communication between a group of nodes (not necessarily adjacent nodes) is needed to handle more complicated constraints (e.g., inter-area flows).

Integral controller utilizes local frequency deviation information to adjust the controllable power generation or load [78, 83, 84]. In general, a decentralized integral controller is able to recover the nominal frequency based on local measurement, but unable to achieve the optimal operating point where the cost is minimized. By communicating marginal generation costs between nearby controllable nodes, the distributed averaging-based integral control achieves both the frequency regulation and economic dispatch, if all the controllable nodes are connected by a communication network [84].

Economic dispatch or power sharing can be be achieved by a decentralized droop control, under specific droop coefficients [78, 85]. The frequency deviation serves as

a common reference for the power sharing among all generators. The work closest to ours is the study of a decentralized leaky integral control in [86]. The leaky integral control can achieve both power sharing and arbitrarily small frequency deviation in the steady state. In contrast, we study an integral control that recovers the nominal frequency.

Although either frequency regulation or economic dispatch can be achieved by decentralized control [78, 84], communication is required to achieve both objectives. Loss of communication may lead to sub-optimal control. Using power line measurement, a control policy was developed in [87] to withstand any single communication link failure. The impact of communication network topology on power grid control has been studied in [88, 89].

In this chapter, we study the performance of a decentralized integral controller with properly designed controller gains, for minimizing the adjustable power generation cost. We quantify the gap between the cost under the decentralized control and the minimum possible cost, and derive conditions for joint frequency regulation and economic dispatch, based on the DC power flow model. We study the tradeoff between the cost and the convergence time, by changing the parameters of the controller. We also study the effectiveness of communication on reducing the convergence time, and quantify the importance of each individual communication link in a distributed control that require information exchange between neighbors. The method can be generalized to handle arbitrary convex power generation costs and power generation capacity constraints. Moreover, we observe that a delayed integral control scheme achieves near-optimal generation cost using significantly smaller convergence time.

The rest of the chapter is organized as follows. In Section 5.1, we describe the system model. In Section 5.2, we describe the decentralized integral control scheme and study its performance. In Section 5.3, we study the integral control scheme aided by communication between nodes, and characterize the importance of each individual communication link. In Section 5.4, we extend the integral control to handle arbitrary convex costs and generation capacity constraints, and develop a delayed control policy. Section 5.5 presents simulation results. Section 5.6 concludes the chapter.

5.1 Model

The power grid is modeled by a connected graph $G(V, E)$, which has $n = |V|$ nodes and $m = |E|$ edges. Each node represents a bus, which is connected to a generator or a load. Each edge represents a power transmission line. Let $V_G \subset V$ denote the generators and $V_L \subset V$ denote the loads. We assume that lines are lossless and denote the susceptance of power line (j, k) by B_{jk}. We consider an arbitrary orientation of power lines. A positive power flow on a power line indicates a flow in the same orientation as the power line, and a negative power flow indicates a flow in the opposite orientation. Bus voltages are normalized to 1 pu (per unit).

Let ω_j denote the frequency deviation from the nominal frequency at bus j. Let θ_j denote the phase angle with respect to the rotating framework of nominal frequency (i.e., $\theta_j(t) = (\theta_j + 2\pi \cdot 60\text{Hz} \cdot t) \mod 2\pi$). Let p_j denote the unadjustable power generation or load, and let u_j denote the controllable power generation or load, which take a positive value for net generation and a negative value for net load. Before disturbance, $u_j = 0, \forall j \in V$. We consider a DC power flow model. The power dynamics at a generator, which has moment of inertia M_j and droop coefficient D_j, follow the swing equation

$$M_j\dot{\omega}_j = -D_j\omega_j + p_j + u_j - \sum_{k\in V} B_{jk}(\theta_j - \theta_k), \quad \forall j \in V_G. \tag{5.1}$$

The power dynamics at a load, which has a linear frequency-dependent load coefficient D_j, follow the equation

$$0 = -D_j\omega_j + p_j + u_j - \sum_{k\in V} B_{jk}(\theta_j - \theta_k), \quad \forall j \in V_L. \tag{5.2}$$

We study the frequency regulation and economic dispatch problems after a power flow perturbation. The objective of frequency regulation is to recover the nominal frequency at all locations. The objective of economic dispatch is to minimize the total cost of adjustable generation and load. For simplicity, we consider the minimization

of the sum of quadratic cost functions, where a_j is the cost coefficient at j.

$$\min \quad \sum_{j\in V} \tfrac{1}{2} a_j u_j^2 \tag{5.3}$$

$$\text{s.t.} \quad p_j + u_j - \sum_{k\in V} B_{jk}(\theta_j - \theta_k) = 0, \quad j \in V. \tag{5.4}$$

The power balance constraints Eq. (5.4) guarantee frequency recovery. This can be verified by noticing $\omega = 0$ and $\dot{\omega} = 0$ in Eqs. (5.1) and (5.2) if Eq. (5.4) holds in the steady state.

The *marginal cost* of power generation is the rate of change in cost by increasing the net generation. In the optimal solution, the marginal costs of power generation are identical at all locations ($\mathrm{d}(a_j u_j^2/2)/\,\mathrm{d}u_j = a_j u_j = a_k u_k, \forall j, k \in V$). We ignore the thermal limits of power lines and generation capacities of generators for simplicity. In Section 5.4, we generalize the methods to minimize arbitrary convex functions, and consider generator capacity constraints.

5.2 Decentralized integral control

Throughout this chapter, we study the control after a perturbation of power generation or load. We assume that the initial power flows are balanced ($\sum_{j\in V} p_j^0 = 0$). After a perturbation of generation or load, by controlling the adjustable power u, $\sum_{j\in V}(p_j + u_j) = 0$ holds in the steady state. We aim to develop a control policy that achieves both frequency regulation and economic dispatch, by properly setting the adjustable power while adhering to the power flow dynamics Eqs. (5.1) and (5.2).

A decentralized frequency integral controller Eq. (5.5) was studied in [84]. The controller measures the local frequency deviation ω, and adjusts u according to the measurement. It has been shown in [84] that the controller converges to the steady-state and recovers the nominal frequency for any $K_j > 0$, due to the negative feedback loop. We show that by properly setting K_j, the controller achieves near-optimal

economic dispatch.

$$\dot{u}_j \;=\; -K_j\omega_j, \;\; \forall j \in V. \tag{5.5}$$

We prove the following theorem.

Theorem 5.1. *For $K_j = h/a_j$, the steady-state cost under the decentralized control is at most $4(\Delta p)^2 nh/(b\lambda_2)$ more than the optimal cost, where Δp is the initial power change at any bus, λ_2 is the algebraic connectivity of the unweighted graph G, $h > 0$, and b is the minimum absolute value of the power line susceptance.*

Remark 1. The decentralized control achieves frequency regulation and near-optimal economic dispatch, if $K_j = h/a_j$, $h > 0$, and either of the two conditions are satisfied:

1) the absolute values of power line susceptances are large.

2) h is small.

Remark 2. By setting h small, the gap $4(\Delta p)^2 nh/(b\lambda_2)$ becomes small. However, the convergence time increases, because the controller gain in Eq. (5.5) is small. There is a tradeoff between the cost and the convergence time. In Section 5.3, we study the effects of communication in reducing the convergence time.

Remark 3. Previous work [78] studied a method for economic dispatch using decentralized droop control, by properly setting the droop coefficients. There exists a non-zero frequency deviation in the steady state, and the common frequency deviation at all buses serves as a reference for power sharing or cost minimization. Our methods are significantly different from [78]. Instead of using global consensus information (i.e., frequency deviation), we study the properties of power flows in steady state, and utilize the invariance Eq. (5.7) to design the controller gains to minimize the cost.

In the rest of the section, we first present the intuition and preliminaries for the performance analysis of the decentralized controller, and then provide the proof of the theorem.

5.2.1 Preliminary and intuition

Let C be the network incidence matrix, which has n rows and m columns. Suppose that the l-th edge is oriented from node j to node k. Then $C_{jl} = 1$ and $C_{kl} = -1$. Let B be an $m \times m$ diagonal matrix, whose l-th diagonal represents the susceptance of the l-th power line. Let θ^0 denote the initial phase angles before the perturbation, and let θ denote the phase angles in the steady state after the perturbation. In the steady states, the frequency stays fixed at the nominal frequency ($\omega = \dot{\omega} = 0$), and the power flows are balanced at each bus.

$$p^0 = CBC^\top \theta^0; \qquad p + u = CBC^\top \theta.$$

Subtracting the two equations,

$$CBC^\top(\theta - \theta^0) = p + u - p^0. \tag{5.6}$$

The phase angle difference is given by Lemma 5.1.

Lemma 5.1. *The difference of phase angles $\theta - \theta^0$ can be determined up to a constant shift. I.e.,*

$$\theta - \theta^0 = (CBC^\top)^+(p - p^0 + u) + c1_{n\times 1}, \tag{5.7}$$

where $(CBC^\top)^+$ denotes the pseudo-inverse of $CBC^\top$.

Proof. For a connected graph $G(V, E)$, C has rank $n - 1$. Since B is diagonal and positive definite, the graph Laplacian matrix $CBC^\top$ has rank $n - 1$. The nullspace of $CBC^\top$ has dimension 1 and is spanned by the vector $1_{n\times 1}$. To prove that $\theta - \theta^0$ given by Eq. (5.7) is the solution to Eq. (5.6), it suffices to verify that

$$(CBC^\top)(CBC^\top)^+(p - p^0 + u) = p - p^0 + u. \tag{5.8}$$

Using linear algebra techniques,

$$(CBC^\top)(CBC^\top)^+ = I - \frac{1}{n}J, \tag{5.9}$$

where I is an $n \times n$ identity matrix, and J is an $n \times n$ matrix with all one elements. See Lemma 3 in [90] for a proof for unweighted graph Laplacian. The same techniques can be used to prove the weighted graph Laplacian in Eq. (5.9).

Since the power flows are balanced in the steady states, $\sum_{j\in V} p_j^0 = \sum_{j\in V}(p_j + u_j) = 0$. Therefore, $J(p - p^0 + u) = 0$. Since $I(p - p^0 + u) = p - p^0 + u$, we have proved Eq. (5.8). □

Let K be an $n \times n$ diagonal matrix whose j-th diagonal equals K_j. Given the control policy Eq. (5.5), in the steady state, the amount of adjustable power is given by

$$u = -K(\theta - \theta^0). \tag{5.10}$$

If $K_j = h/a_j$, then $a_j u_j = h(\theta_j^0 - \theta_j)$. If the diagonals of B are large, $(CBC^\top)^+(p - p^0 + u)$ is small and $\theta - \theta^0$ is almost equal to $c1_{n\times 1}$. The marginal costs at all generators $a_j u_j$ are almost the same, thus achieving the near-optimal economic dispatch.

5.2.2 Proof of Theorem 5.1

We consider a power perturbation at node k and denote the amount of power change by Δp. Without loss of generality, we assume that $\Delta p < 0$ (i.e., load increase or generation decrease). After the change, the frequency drops below the nominal frequency and $u > 0$. In the steady state after the change, $\sum_{j\in V} u_j = -\Delta p$. The L_1 norm of the vector $p - p^0 + u$ is at most $\sum_{j\in V} |p_j - p_j^0 + u_j| = \sum_{j\in V, j\neq k} u_j + |\Delta p + u_k| \leq 2|\Delta p|$. Suppose that the absolute value of every element of $(CBC^\top)^+$ is at most M. Then, the absolute value of every element in $(CBC^\top)^+(p - p^0 + u)$ is at most $2M|\Delta p|$. Therefore,

$$|(\theta_i - \theta_i^0) - (\theta_j - \theta_j^0)| \leq 4M|\Delta p|, \quad \forall i, j \in V.$$

The difference of the marginal costs at i and j is at most

$$\begin{aligned} |a_i u_i - a_j u_j| &= h|(\theta_i - \theta_i^0) - (\theta_j - \theta_j^0)| \\ &\leq 4hM|\Delta p| \end{aligned}$$

Since the marginal costs differ by at most $4hM|\Delta p|$, and the marginal cost at each node is an increasing function in the generation amount, the cost saving in dispatching one unit power generation to a different node is at most $4hM|\Delta p|$. The total amount of adjustable generation is $\sum_j u_j = |\Delta p|$. Therefore, the generation cost under the control Eq. (5.5) is at most $4hM(\Delta p)^2$ higher than the optimal generation cost.

Next we bound M. By spectral decomposition, the symmetric matrix $CBC^\top = UDU^\top$, where U is an orthonormal matrix and D is a diagonal matrix. The diagonals of D are the eigenvalues of $CBC^\top$. Let λ' denote the set of eigenvalues. Let v_i be the i-th column of D.

$$CBC^\top = \sum_{i=1}^{n} \lambda'_i v_i v_i^\top.$$

Moreover, the graph Laplacian matrix $CBC^\top$ is positive semi-definite and has rank $n-1$, which has the smallest eigenvalue $\lambda'_1 = 0$ and $n-1$ positive eigenvalues. The pseudo-inverse of $CBC^\top$ is given by

$$(CBC^\top)^+ = \sum_{i=2}^{n} (1/\lambda'_i) v_i v_i^\top.$$

Let $L = CC^\top$ be the Laplacian of the unweighted graph G. The second smallest eigenvalue is the algebraic connectivity of G, and is given by

$$\lambda_2 = \min\{\frac{y^\top L y}{y^\top y} | y \neq 0, 1_{1\times n} y = 0\}. \tag{5.11}$$

Let b be the smallest susceptance. $CBC^\top = bL + L'$. The matrix L' can be viewed as the Laplacian of the weighted graph where edge (j,k) has a weight $B_{jk} - b \geq 0$, and is positive semi-definite. The second smallest eigenvalue of $CBC^\top$ is bounded by Eq. (5.15). The vector y^* in Eq. (5.13) is the vector that achieves the minimum in Eq. (5.12). Inequality (5.14) follows from that L' is positive semi-definite. Inequality (5.15) follows from Eq. (5.11).

$$\lambda'_2 = \min\{\frac{y^\top CBC^\top y}{y^\top y} | y \neq 0, 1_{1\times n} y = 0\} \tag{5.12}$$

$$= \frac{y^{*\top}(bL + L')y^*}{y^{*\top}y^*} \tag{5.13}$$

$$\geq \frac{y^{*\top}bLy^*}{y^{*\top}y^*} \tag{5.14}$$

$$\geq b\lambda_2. \tag{5.15}$$

Since $||v_i||_2 = 1$, the absolute value of every element in $v_i v_i^\top$ is at most 1. The absolute value of every element in $(CBC^\top)^+$ is at most $M \leq \sum_{i=2}^{n}(1/\lambda_i') \leq n/(b\lambda_2)$. Therefore, the generation cost under the controller is at most $4(\Delta p)^2 nh/(b\lambda_2)$ higher than the optimal generation cost.

5.3 Distributed control under partial communication

In the previous section, we studied an integral controller that adjusts controllable power based on local measurement and does not require any communication. In this section, we study the benefit of communication in power grid control. Communication is useful to exchange the marginal cost information between the controllable nodes. It reduces the convergence time of the control, by eliminating the need to use a small controller gain for economic dispatch.

Consider a communication network $G'(V', E')$, where $V' = V$ denotes the buses in the power grid, and E' denotes the communication links. It has been shown that by exchanging the marginal costs between neighbors, a distributed averaging-based integral control can achieve both frequency regulation and economic dispatch, if G' is connected [84]. In this section, we develop a control policy under the failures of communication links, and identify important communication links on the control.

Let E^* denote the minimum set of links that are parallel to power lines and merge the disjoint communication components into a connected graph. Let V^* denote the nodes adjacent to E^*. Notice that E^* and V^* are non-empty if and only if $G'(V', E')$ is disconnected. See Fig. 5-1 for an illustration.

We study the performance of the control policy given by Eqs. (5.16) and (5.17),

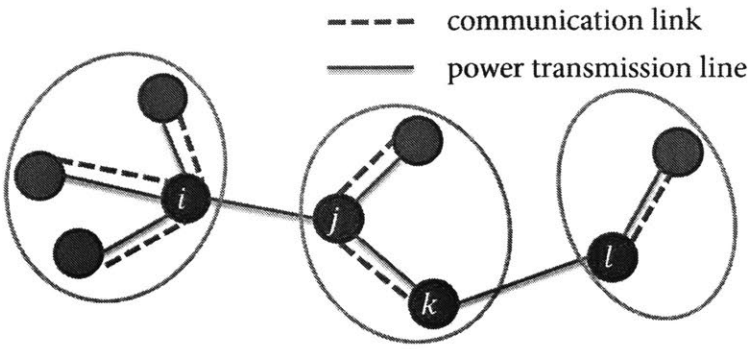


Figure 5-1: Illustration of communication components. Links $E^* = \{(i,j),(k,l)\}$ connect three disjoint communication components. Nodes $V^* = \{i,j,k,l\}$ are their adjacent nodes.

where $K_j = h/a_j$. The integral control in Section 5.2 is applied to nodes V^* (i.e., Eq. 5.16). However, for nodes that are connected by communication links, i.e., $V \backslash V^*$, the distributed averaging controller of [84] is used, and nearby nodes exchange the marginal costs $a_j u_j$. In the steady state, $a_j u_j = a_k u_k$ for j and k in the *same* communication component, by the analysis in [84]. The key is to bound the gap between the marginal costs in different components.

$$\dot{u}_j = -K_j \omega_j, \quad \forall j \in V^*, \tag{5.16}$$

$$\dot{u}_j = -K_j \omega_j - \sum_{(j,k)\in E} (a_j u_j - a_k u_k), \forall j \in V \setminus V^*. \tag{5.17}$$

For simplicity, we assume that the communication network initially has the same topology as the power grid. Suppose that the communication link between i and j fails, and i and j are separated in two communication components. Then, $i, j \in V^*$. By the analysis in Section 5.2, $a_i u_i$ and $a_j u_j$ are given by $h(\theta_i^0 - \theta_i)$ and $h(\theta_j^0 - \theta_j)$, respectively.

By left-multiplying both sides of Eq. (5.7) by $BC^\top$, Eq. (5.18) holds regardless of the control policy on u. Let the i-th diagonal of the diagonal matrix D be $D_i = \sqrt{B_i}$,

which satisfies $B = DD^\top$.

$$\begin{aligned} BC^\top(\theta - \theta^0) &= BC^\top(CBC^\top)^+(p + u - p^0). \\ BC^\top(\theta - \theta^0) &= D(CD)^+(p + u - p^0). \end{aligned} \tag{5.18}$$

Let y denote the $m \times 1$ vector $D(CD)^+(p + u - p^0)$. Recall that C is an adjacency matrix with $C_{il} = 1$ and $C_{jl} = -1$ if the l-th edge is oriented from i to j. The susceptance of the power line that connects i and j is the l-th diagonal value B_l. We obtain

$$(\theta_i - \theta_i^0) - (\theta_j - \theta_j^0) = y_l / B_l.$$

If B_l is large, then $|(\theta_i - \theta_i^0) - (\theta_j - \theta_j^0)|$ is small. Recall Eq. (5.10). Under the integral control Eq. (5.16), the gap between the marginal costs at i and j (i.e., $|a_i u_i - a_j u_j|$) is small. Intuitively, given a bounded power flow on the power line $B_l(\theta_i - \theta_j)$, if the susceptance B_l is large, the difference between phase angles $\theta_i - \theta_j$ is small. Therefore, the difference of phase angles $\theta_i - \theta_i^0$ is close to $\theta_j - \theta_j^0$, which indicates a small gap in the marginal costs at nodes i and j.

To conclude, under the control policy given by Eqs. (5.16) and (5.17), the failure of a communication link has less severe impacts if its associated power line has a large susceptance, which will be further verified using simulation.

5.4 Variations of the integral control

In this section, we extend the the decentralized control in Section 5.2 to handle arbitrary convex costs for adjustable power and generator capacity constraints. Moreover, we study the benefit of delayed control on minimizing the total costs.

5.4.1 Arbitrary convex costs and generator capacity constraints

Let $f_j(u)$ denote the cost of increasing the generation (or decreasing the load) by u at node j. We assume that $f_j(u)$ is strictly convex and differentiable, and attains the minimum at $f_j(0) = 0$, $\forall j \in V$. The derivative $g_j(u) = f_j'(u)$ is monotonically increasing, and the inverse $g_j^{-1}(v)$ is well defined.

We study a control policy given by Eqs. (5.19) and (5.20).

$$\dot{v}_j = -h\omega_j, \quad \forall j \in V. \tag{5.19}$$

$$u_j = g_j^{-1}(v_j), \quad \forall j \in V. \tag{5.20}$$

The controller gain h is positive and identical at all nodes. For the special case of quadratic cost $f_j(u_j) = a_j u_j^2/2$, $g_j(u_j) = a_j u_j$, the control is equivalent to Eq. (5.5) with $K_j = h/a_j$.

The controller measures the local frequency deviation ω_j, and then adjusts a virtual price v_j. The virtual price serves as a reference for the controllable power generation. The marginal cost of power generation at node j is v_j, guaranteed by Eq. (5.20).

If the power line susceptances are large, or the controller gain h is small, the virtual prices v and the marginal costs at different controllable nodes are close. Thus, the total cost is approximately minimized. More precisely,

Corollary 5.1. *For a strictly convex and differentiable cost $f_j(u)$ that attains the minimum at $f_j(0) = 0$, $\forall j \in V$, the steady-state cost under the decentralized control (5.19) and (5.20) is at most $4(\Delta p)^2 nh/(b\lambda_2)$ more than the optimal cost, where Δp is the initial power change at a bus, λ_2 is the algebraic connectivity of unweighted graph G, $h > 0$, and b is the minimum absolute value of the power line susceptance.*

Proof. The function $g_j(u_j) = f_j'(u_j)$ denotes the rate of cost change at node j as the amount of net adjustable generation increases. We aim to prove that the difference

of the marginal costs at i and j is

$$|f_i'(u_i) - f_j'(u_j)| \;=\; h|(\theta_i - \theta_i^0) - (\theta_j - \theta_j^0)|, \tag{5.21}$$

where θ^0 are the phase angles before perturbation and θ are the phase angles in the steady state after the perturbation. The rest of the proof follows from the proof of Theorem 5.1.

Under Eq. (5.19), $v_j = -h(\theta_j - \theta_j^0)$. We obtain

$$|v_i - v_j| \;=\; h|(\theta_i - \theta_i^0) - (\theta_j - \theta_j^0)|.$$

Since $f_j(u)$ is strictly convex and differentiable, $g_j(u) = f_j'(u)$ is monotonically increasing, and is a bijection function. According to Eq. (5.20),

$$f_j'(u_j) = g_j(u_j) = v_j, \quad \forall j \in V.$$

Therefore, we have proved Eq. (5.21). □

To handle the power generation capacity constraints, it suffices to replace Eq. (5.20) by the following equation.

$$u_j = \max(c_j^1, \min(c_j^2, g_j^{-1}(v_j)), \quad \forall j \in V,$$

where $[c_j^1, c_j^2]$ is range of controllable net generation at node j. Under the same analysis, the total cost is approximately minimized, and the frequency is recovered to the nominal frequency in the steady state, as long as it is feasible to balance the power generation and load under the capacity constraints.

5.4.2 Delayed control

The integral of frequency deviation is utilized at each controllable node to serve as a reference for the marginal cost of adjustable power. In previous sections, we studied conditions for the references to be nearly identical at all locations in order

for economic dispatch. Next, we study a controller that only adjusts the controllable power using frequency deviation information after a timeout period T, given by Eqs. (5.22) and (5.23).

$$\dot{u}_j(t) = 0, \qquad t \leq T, \forall j \in V. \tag{5.22}$$

$$\dot{u}_j(t) = -K_j\omega_j, \qquad t > T, \forall j \in V. \tag{5.23}$$

The intuition is that, after some time T without any control on u, the frequency deviations at all nodes become almost identical. The deviations could serve as references to adjust u. In the numerical result section, we observe significant cost savings by the delayed control, at similar convergence time compared with the original integral control.

5.5 Numerical results

In this section, we verify the performance of the controllers using a simple example with 10 nodes and 10 edges. The network topology (Fig. 5-2) and the data are identical to those in [87]. We study the control after a perturbation of 5 units load increase at node 3. The minimum sum of quadratic costs shown in Eq. (5.3) is 23.27. For completeness, the data are presented below.

For 10 nodes (white numbers indicate node ID),
Inertia $M = \{0.01, 0.02, 0.01, 0.1, 0.05, 0.8, 0.05, 1, 0.1, 0.01\}$.
Initial power $p = \{1, 5, -2, 6, -5, -10, -4, 8, 5, -4\}$.
Droop coefficient $D = \{0.33, 1.67, 0.67, 2.00, 1.67, 3.33, 1.33, 2.67, 1.67, 1.33\}$.
Cost coefficient $a = \{20, 20, 200, 200, 10, 20, 14, 18, 10, 20\}$.

For 10 power lines (black numbers indicate line ID),
Susceptance $B = \{1.00, 0.50, 0.33, 1.00, 0.20, 0.25, 0.17, 1.00, 0.11, 1.00\}$.

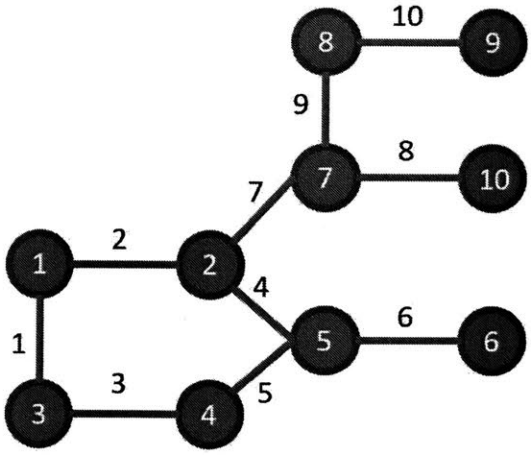


Figure 5-2: Topology of the power grid.

5.5.1 Cost vs. controller gain and power line susceptance

We evaluate the total costs in the steady state after the perturbation, for different values of controller gains. The costs are compared in Fig. 5-3. We observe that as h decreases, the cost under the integral control Eq. (5.5) approaches the optimal cost. The near-linear dependence on h matches the predictions in Theorem 5.1.

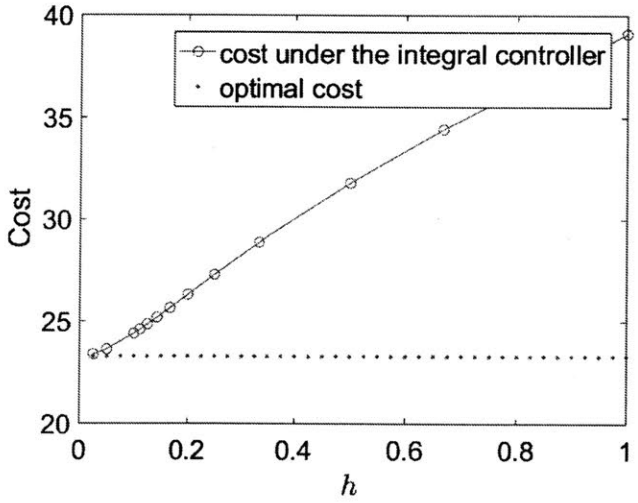


Figure 5-3: Cost decreases as the controller gain decreases.

By dividing all line susceptances by the values in the x-axis, the cost decreases and follows the same curve as Fig. 5-3. This can be explained analytically. From Eq.

(5.10), we obtain

$$\theta - \theta^0 = -K^{-1}u.$$

Moreover,

$$p + u - p^0 = (CBC^\top)(\theta - \theta^0).$$

Therefore,

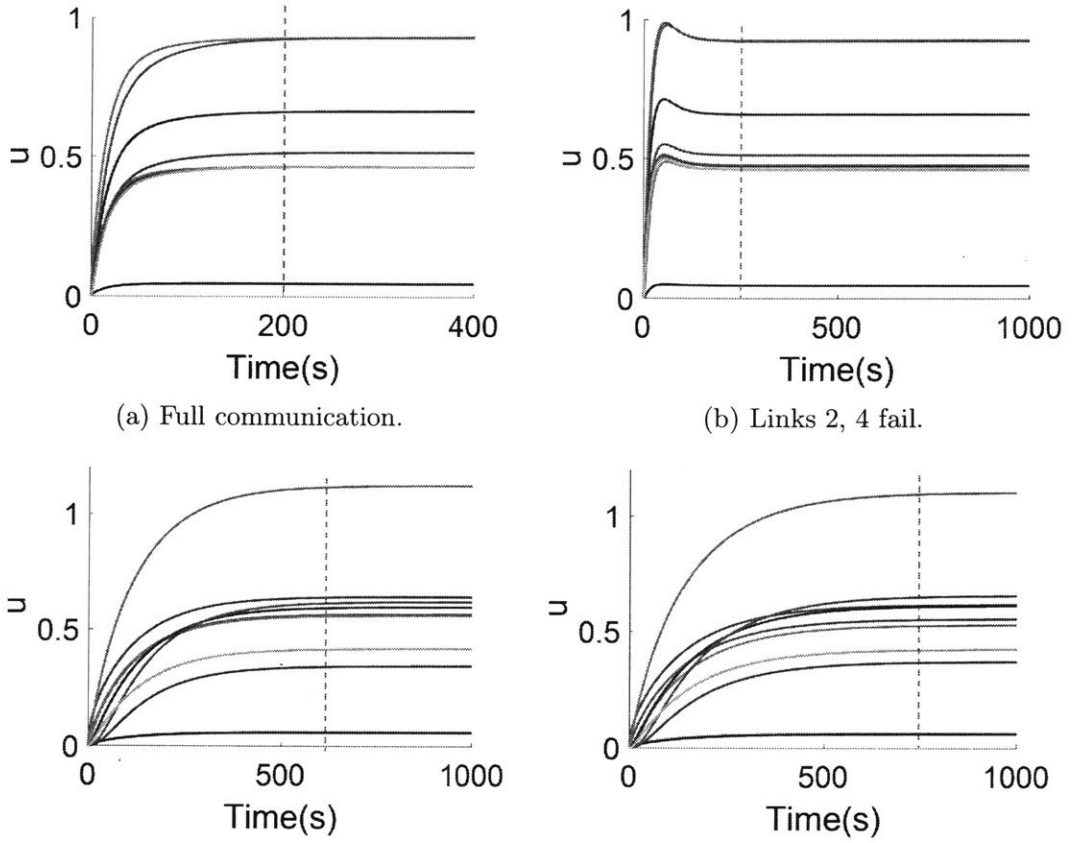
$$(I + CBC^\top K^{-1})u = p^0 - p,$$

where I is the identity matrix. Since $C(\alpha B)C^\top K^{-1} = CBC^\top (K/\alpha)^{-1} = \alpha CBC^\top K^{-1}$, the adjustable power u are identical under 1) power line susceptance αB and controller gain h; 2) power line susceptance B and controller gain h/α, for any positive scaler α.

5.5.2 Reducing convergence time using communication

We study the role of communication on reducing the convergence time to reach the steady state while guaranteeing a low cost. We consider a connected communication network that has the same topology as the power grid. Figure 5-4 illustrates the change of adjustable power at all nodes as time increases, under the control Eqs. (5.16) and (5.17). The four figures correspond to the scenario where there is no communication link failure, links $\{2, 4\}$ failure, links $\{2, 4, 9\}$ failure, and all links failure, respectively.

The cost under the control with a connected communication network (Figs. 5-4a) is 23.27, with convergence time around 200 seconds under $h = 1$. The costs for the other scenarios under communication link failures are set to be around 24.43, which is 5% higher than the optimal cost. In order to achieve the target cost, h is set to be $1/1.7, 1/8.5, 1/9.8$, respectively, and the convergence times are around 250, 600, and 750 seconds, respectively, for Figs. 5-4b, 5-4c, and 5-4d. We observe that the convergence time increases as there are more communication link failures, in order to guarantee the same target cost.



(a) Full communication.

(b) Links 2, 4 fail.

(c) Links 2, 4, 9 fail.

(d) No communication.

Figure 5-4: Convergence time with and without communication. Curves illustrate the adjustable power at all nodes.

5.5.3 Importance of each individual communication link

We verify that the failures of communication links have more significant impact on the cost, if the corresponding power lines have small susceptances. For $h = 1$, we study the control Eqs. (5.16) and (5.17) under three communication link failures. In the left figure, nodes adjacent to links $(1, 2), (2, 5)$ are controlled by Eq. (5.16). The corresponding power lines have larger susceptances 0.5 and 1. In the right figure, nodes adjacent to links $(4, 5), (7, 8)$ are controlled by Eq. (5.16). The corresponding power lines have smaller susceptances 0.2 and 0.1. The total costs in the steady states are 26.17 and 34.36, for the left and right figures, respectively. We observe that the cost is higher if communication fails between nodes connected by power lines with

smaller susceptances.

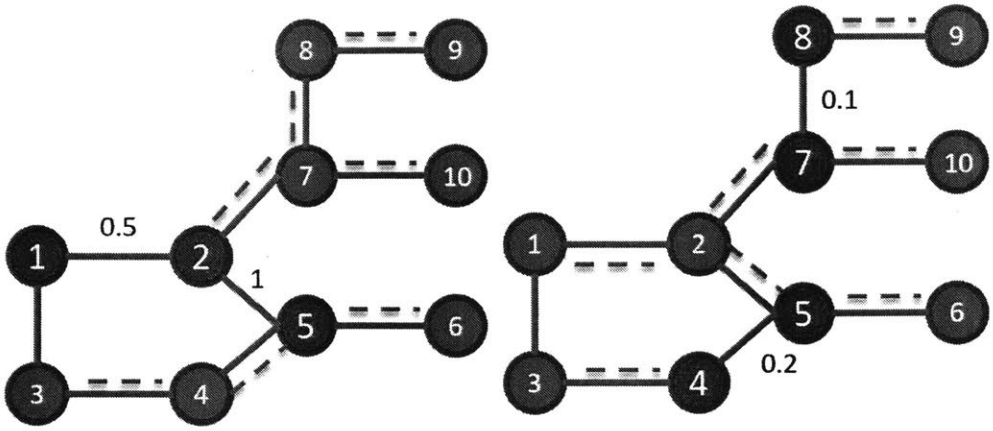


Figure 5-5: Communication link failures.

5.5.4 Delayed control

We study the performance of delayed control Eqs. (5.22) and (5.23). By fixing $h = 1$, in Fig. 5-6, the adjustable power generation under the control without delay ($T = 0$) is illustrated by the left figure, with total cost 39.11. The control with delay $T = 30$ seconds is illustrated by the right figure with total cost 27.50. The convergence times are close (differ by 30 seconds), while the cost under the delayed control is 30% lower than the cost under the control without delay.

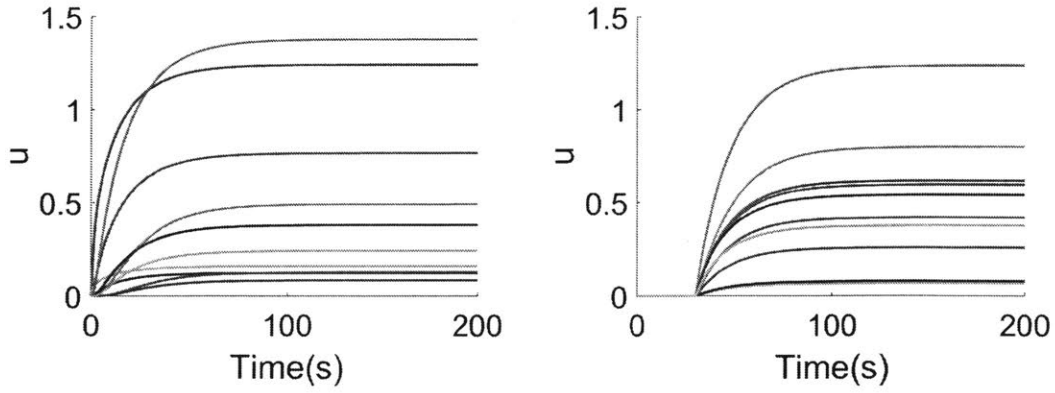


Figure 5-6: Delayed control (left: $T = 0$, right: $T = 30$ s).

5.5.5 General convex cost function

We evaluate the performance of the control Eqs. (5.19) and (5.20), for a cubic cost function $f_j(u_j) = a_j|u_j|^3/3$. The optimal cost is 8.84. The costs obtained by the decentralized integral controller for controller gains h are illustrated in Fig. 5-7. The curve is similar to the curve in Fig. 5-3 for a quadratic cost. The results show that the integral control can be applied to arbitrary convex cost function.

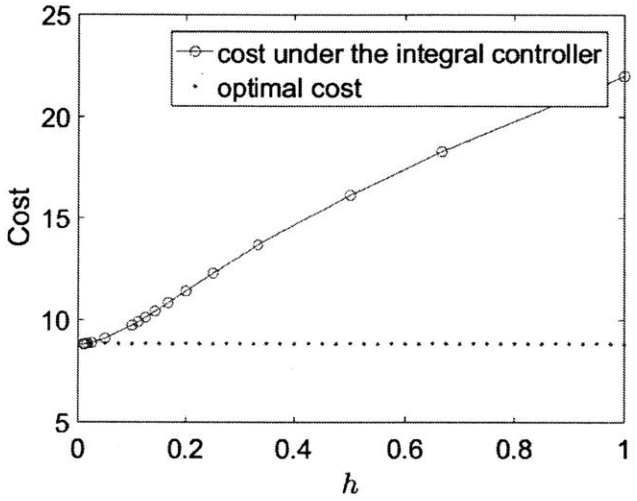


Figure 5-7: Cost for the control under a cubic cost function.

5.6 Summary

We studied a decentralized integral control for joint frequency regulation and economic dispatch. We derived conditions for the control to achieve near-optimal cost, and observed a tradeoff between the cost and the convergence time. We studied the role of communication in reducing the convergence time. Moreover, we extend the control to handle arbitrary convex costs and power generation capacity constraints. Numerical results show that a delayed control reduces the cost significantly with similar convergence time.

Chapter 6

Robustness of distributed computing networks

Cloud computing has been growing rapidly in recent years. For example, over one millon servers have been deployed for Amazon Web Service, which generates billions of revenue each year and grew by over 40 percent in revenue in 2018. Cloud networks, and computing networks in general, facilitate agile, reliable and cost effective implementations for a variety of applications. The robustness of computing networks is essential for web access, online database, video streaming, among other applications deployed in the cloud.

Network flows in computing networks rely on both communication resources for transmission and computation resources for processing. The unavailability of either type of resources may lead to the failure of supporting network flows. In this chapter, we study the robustness of computing networks under the failures of network resources and flow interdiction problems.

The amount of flows supported by a computing network depends on network topology and the allocation of computation resources. The problem of maximizing flow by allocating computation resources was studied in [91, 92]. Under stochastic traffic arrivals, routing and scheduling algorithms were developed in [93,94] to support the maximum flow rates in a computing network.

Network flow interdiction problems have been extensively studied in the previ-

ous literature. The problem of minimizing the maximum flow by removing network links within a budget is strongly NP-hard. Integer linear programs were developed to compute the optimal interdiction [95]. Approximation hardness results and a $2(n-1)$-approximation algorithm was developed in [96]. A pseudoapproximation algorithm based on linear programming relaxation was developed in [97]. NP-hardness result and a polynomial time approximation scheme were developed for network flow interdiction on planer graphs [98,99].

In a traditional flow network, the maximum flow between a source-destination $(s-t)$ pair equals the minimum cut, which is the minimum-capacity link removals that disconnect the $s-t$ pair [100]. In a computing network, we show that there is a non-zero gap between the maximum flow and the minimum cut. The gap implies that a flow may require more communication resources compared with a flow in a traditional communication network, since a flow in a computing network need to be transmitted to computation nodes for processing before delivered to the destination.

The main contributions of this chapter are as follows. We develop a model for a computing network, and formulate cut metrics to study its robustness. We develop efficient algorithms to compute the maximum flow supported by a computing network. We prove the complexity of computing the minimum cuts, and developed integer linear programs and approximation algorithms to compute the minimum cuts. Moreover, we formulate a maximum flow interdiction problem, where the objective is to minimize flow by removing communication and computation resources within a given budget, and develop an integer linear program to compute the optimal interdiction.

The rest of this chapter is organized as follows. In Section 6.1, we introduce the model for a distributed computing network, and define cut metrics to evaluate the network robustness. In Section 6.2, we develop algorithms to evaluate the maximum flow and minimum cuts. In Section 6.3, we formulate and solve a budgeted maximum flow interdiction problem. Section 6.4 provides numerical results. Section 6.5 summarizes the chapter.

6.1 Model

In this section, we develop a model for a distributed computing network, and define metrics for the evaluation of network robustness.

A distributed computing network is modeled by a directed graph $G(V, E)$, where V denotes the set of routers and computation nodes, and E denotes the set of communication links. Computation nodes can process and forward flows, while routers can only forward flows. Let μ_u denote the processing capacity at node u. Let μ_{uv} denote the transmission capacity at link (u, v).

Unlike the traditional data network where flows require minimal fixed computation tasks such as routing table lookup and checksum, flows in the distributed computing network can require vastly different computation resources, and hence computation capacities at servers (as well as communication bandwidth) are essential to process traffic. The classical robustness metric such as minimum cut is not able to capture the robustness of such a computing network. We extend classical flow and cut metrics to computing networks, to characterize the need to incorporate both communication and computation resources in network operation.

We first define *computation path* which supports both the processing and the delivery of data packets in the network.

Definition 6.1. A *computation path* (P, w) from a source s to a destination t is characterized by a sequence of connected edges and nodes P that start at s and end at t, and includes a computation node $w \in P$.

Network flows are transmitted and processed by computation paths. In order to reduce the flow carried by a computation path to zero, either any communication link or the computation resource in the path should be removed. Note that we consider the removal of computation resources without removing the node, *i.e.*, the node can still forward packets without processing them.

In general, there are multiple computation paths from a source to a destination. To interdict the flow, a combination of communication and computation resources can be removed. We next define cuts that measure the connectivity of a pair of nodes

in a computing network.

Definition 6.2. A *communication cut* is a set of communication links E_c such that the flow is reduced to zero after removing E_c.

Definition 6.3. A *computation cut* is a set of computation nodes V_c such that the flow is reduced to zero after removing the computation resources at V_c.

Definition 6.4. A *joint communication and computation cut* (abbr. *joint cut*) is a set of communication links E_c and computation nodes V_c such that the flow is reduced to zero after removing E_c and computation resources at V_c.

We illustrate these cuts using an example in Fig. 6-1, where computation nodes are illustrated by squares. Edges $\{(u_1, v_1), (u_2, v_2)\}$ form a communication cut, since s and t are disconnected after removing these two edges. Nodes $\{u_1, u_2, u_3, u_4\}$ form a computation cut, since no flow can be processed after removing the computation resources at the four computation nodes. The union of edge $\{(u_1, u_2)\}$ and nodes $\{u_3, u_4\}$ is a joint cut, since the upper path is disconnected after removing edge (u_1, u_2), and the lower path cannot process flow after removing the computation resources at nodes $\{u_3, u_4\}$.

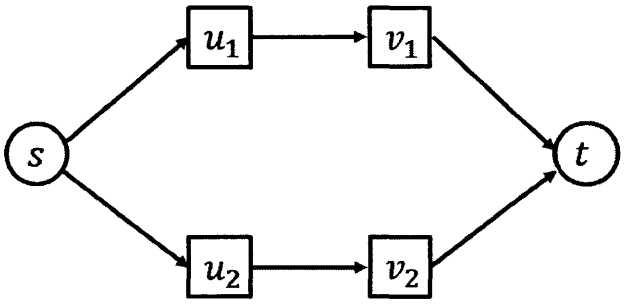


Figure 6-1: Illustration of cuts.

To simplify the analysis for network robustness, we assume that *every unit flow requires the same amount of communication and computation resources.* The identical resource requirement of flows can be justified by the statistical multiplexing of individual flows in networks, although individual flows for different applications may have different resource requirements. For example, video streaming is communication

intensive, while search is computation intensive. By normalizing units, we further assume that *every unit flow requires one unit computation resource for processing, and outputs one unit processed flow.* Under this assumption, one unit flow on a computation path occupies one unit communication resource at every link along the path, and a total of one unit computation resource at one or more computation nodes along the path. Moreover, we ignore flow scaling due to processing.

Before developing algorithms to evaluate the maximum flow and the minimum cuts, we prove the complexity of computing the cut metrics. First, we show the hardness of evaluating the minimum communication cut, whose proof is in the Appendix.

Lemma 6.1. *Computing the minimum communication cut for an $s-t$ pair is NP-hard, if there is more than one computation node.*

Lemma 6.1 implies that computing the minimum joint cut is NP-hard, since the minimum communication cut can be viewed as a special case of the minimum joint cut when the computation resources are abundant at all nodes.

Theorem 6.1. *Computing the minimum joint communication and computation cut for an $s-t$ pair is NP-hard, if there is more than one computation node.*

6.2 Computation of max-flow and min-cuts

In this section, we study the computation of the maximum flow and minimum cuts for a source-destination pair. We develop polynomial-time algorithms to evaluate the maximum flow and the minimum computation cut, and integer programs to evaluate the minimum communication cut and the minimum joint cut. In Section 6.2.1, we develop mathematical programs to evaluate the maximum flow and the minimum cut using path-based formulation, which is intuitive but has an exponential number of variables or constraints. In Section 6.2.2, we develop a layered graph representation to simplify their computations, and develop mathematical programs of polynomial sizes. Finally, in Section 6.2.3, we study the gap between the maximum flow and the minimum cut.

6.2.1 Path-based formulations

We first develop mathematical programs to compute the maximum flow and minimum cuts using path-based formulations. Although the formulations have an exponential number of variables or constraints, they illustrate the connections between flow and cuts in a computing network and those in a classical flow network.

We formulate a linear program to compute the maximum flow in a computing network. Let $\mathcal{P}$ denote the set of $s-t$ paths. Let $x_{P,w}$ denote the amount of flow transmitted through path P and processed at a computation node $w \in P$. The maximum flow can be computed by the following linear program.

$$\max \quad \sum_{P\in\mathcal{P}, w\in P} x_{P,w} \tag{6.1}$$

$$\text{s.t.} \quad \sum_{P\in\mathcal{P}, w\in P:(u,v)\in P} x_{P,w} \leq \mu_{uv}, \quad \forall (u,v) \in E, \tag{6.2}$$

$$\sum_{P\in\mathcal{P}:w\in P} x_{P,w} \leq \mu_w, \quad \forall w \in V, \tag{6.3}$$

$$x_{P,w} \geq 0, \quad \forall P \in \mathcal{P}, w \in P.$$

The communication capacity constraints are guaranteed by (6.2), and the computation capacity constraints are guaranteed by (6.3), by restricting the total amount of flow that is transmitted by a link or processed at a computation node.

We then develop an integer program to evaluate the minimum joint communication and computation cut using the path-based formulation. Indicator variable y_{uv} represents whether link (u,v) is removed. Indicator variable y_w represents whether the computation resource at node w is removed. Constraint (6.5) guarantees that for each path, either one of the link is removed, or all the computation resources are removed.

$$\min \quad \sum_{(u,v)\in E} \mu_{uv} y_{uv} + \sum_{w\in V} \mu_w y_w \tag{6.4}$$

$$\text{s.t.} \quad \sum_{(u,v)\in P} y_{uv} + y_w \geq 1, \quad \forall P \in \mathcal{P}, w \in P \tag{6.5}$$
$$y_{uv} \in \{0,1\}, \quad \forall (u,v) \in E$$
$$y_w \in \{0,1\}, \quad \forall w \in V.$$

A minimum communication cut can be obtained by the same integer program (6.4), by setting $y_w = 0, \forall w \in V$. A minimum computation cut can be obtained by setting $y_{uv} = 0, \forall (u,v) \in E$.

The number of paths $|\mathcal{P}|$ can be exponential in the size of the network. Both the linear program (6.1) and the integer program (6.4) has exponential sizes. Compared with the classical maximum flow and minimum cut formulations, the main difference is that a computation path in the computing network depends on a computation node in addition to a sequence of connected links. The coupling of constraints by the computation nodes brings challenges to the evaluation of the metrics.

6.2.2 Layered graph formulations

We then develop a layered graph representation to simplify the evaluation of flow and cuts. Based on the layered graph, in Sections 6.2.2 and 6.2.2, we develop modified mathematical programs with a polynomial number of variables and constraints to evaluate the maximum flow and the minimum cuts, respectively.

We consider a two-layer graph, where every layer has the same topology as the original graph. An edge connects the two copies of each computation node across the two layers. Unprocessed flows are transmitted thought links in the upper layer $G(V,E)$, while processed flows are transmitted in the lower layer $G'(V',E')$. Flows across the two layers represent processing at computation nodes. For example, in Fig. 6-2, a flow is transmitted through (s,u), processed at u, and then transmitted through (u,v) and (v,t). In the layered graph, unprocessed flow is transmitted through (s,u) in the upper layer, then transmitted through (u,u'), which represents the processing at u, and finally transmitted through (u',v') and (v',t') in the lower layer. Every flow from s to t and processed at computation nodes in the original graph can be

represented by a flow from s to t' in the layered graph.

Lemma 6.2. *Let S be an $s-t$ cut in the computing network. In the layered graph, removing edges $S' = \{(u,v),(u',v')|(u,v) \in S\} \cup \{(w,w')|w \in S\}$ disconnects s and t'.*

Proof. We prove by contradiction. Suppose that a path P exists between s and t' in the layered graph after removing S'. The path P contains a link from the upper layer to the lower layer, denoted by (a,a'). There is a path P_1 from s to a in the upper layer, and a path P_2' from a' to t' in the lower layer. Let $P_2 = \{(u,v)|(u',v') \in P_2'\}$. Since none of the edges $P_1 \cup P_2'$ belong to cut S', none of the edges in $P_1 \cup P_2$ belong to cut S.

In the computing network, there is a path P_1 from s to a, and a path P_2 from a to t. Moreover, the computation resource at a is not removed, since (a,a') remains. The path $(P_1 \cup P_2, a)$ is a computation path from s to t, which contradicts with the fact that S be an $s-t$ cut. □

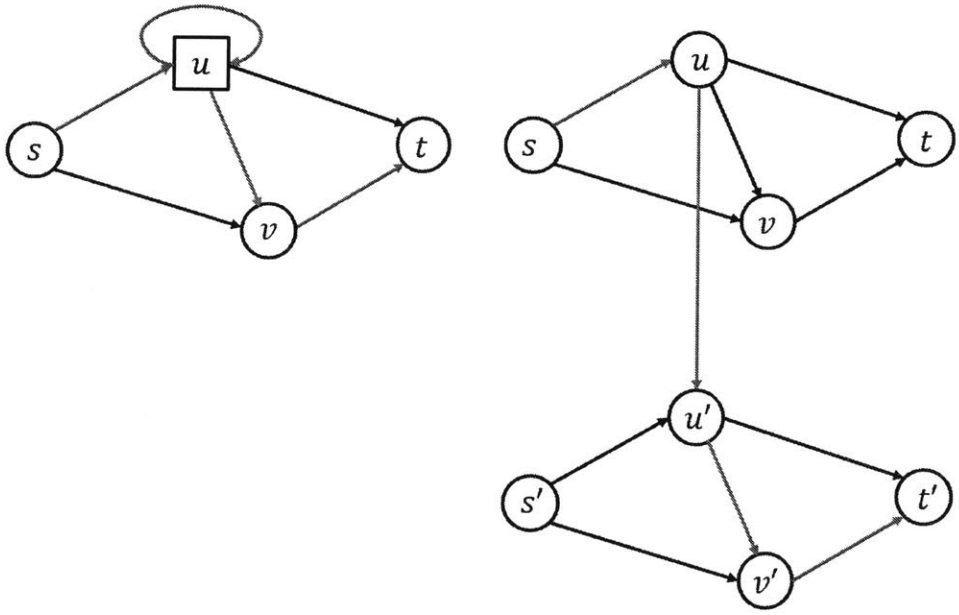


Figure 6-2: Flows in the original and layered graphs.

Evaluation of maximum flow

Flow conservation holds in the layered graph, since communication and computation units are normalized and flow scalings are ignored. The maximum flow that can be delivered from s to t while processed can be represented by a flow in the layered graph from s to t'. The difference of a flow in the layered graph from the classical network flow is that the sum of flows on the two copies of a link should not exceed the transmission capacity. Let $\tilde{E} = E \cup E' \cup (t', s)$ denote the union of the set of edges in the layered graph and an edge from t' to s that has an infinite capacity. Let $\tilde{V} = V \cup V'$ denote the set of nodes in the layered graph. Let f_e denote the amount of flow on link e. The maximum flow from s to t' can be computed using the following linear program.

$$
\begin{aligned}
\max \quad & f_{t's} \\
\text{s.t.} \quad & \sum_{u \in \tilde{V}:(u,v) \in \tilde{E}} f_{uv} - \sum_{w \in \tilde{V}:(v,w) \in \tilde{E}} f_{vw} = 0, \forall v \in \tilde{V}, && (6.6) \\
& f_{ww'} \leq \mu_w, \quad \forall w \in V, && (6.7) \\
& f_{uv} + f_{u'v'} \leq \mu_{uv}, \quad \forall (u,v) \in E, && (6.8) \\
& f_{uv} \geq 0, f_{u'v'} \geq 0, \quad \forall (u,v) \in E, \\
& f_{ww'} \geq 0, \quad \forall w \in V.
\end{aligned}
$$

Flow conservation constraints are guaranteed by (6.6). Computation capacity constraints are guaranteed by (6.7) for each computation node. Communication capacity constraints are guaranteed by (6.8) for each communication link. The linear program has $O(|E|)$ variables and $O(|E|)$ constraints, which has a significantly smaller size compared with the path-based linear program formulation. To conclude, the maximum flow can be computed by the linear program in polynomial time.

Evaluation of minimum cuts

Recall that an $s-t$ flow can be interdicted by removing either communication or computation resources, or a combination of both. We first develop an integer program to compute the minimum joint communication and computation cut, which can be easily modified to compute the minimum communication cut and the minimum computation cut. The formulation is based on disconnecting s and t' in the layered graph, which equivalently reduces the $s-t$ flow to zero in the original graph by Lemma 6.2.

Let y_{uv} indicate whether link (u,v) is removed. Let z_w indicate whether the computation resource at node w is removed. Let p_v be the potential of a node, where potential never decreases along a connected path, guaranteed by constraints (6.10), (6.11), and (6.12) when $y_{uv}=0$ and $z_w=0$. However, disconnected nodes may have different potentials, guaranteed by the same constraints when $y_{uv}=1$ or $z_w=1$. Constraint (6.13) guarantees that s has higher potential than t'. If all the constraints are satisfied, s and t' are disconnected, since the potential cannot decrease through a connected path. The cut include the communication links where $y_{uv}=1$ and computation nodes where $z_w=1$. Notice that if link (u,v) is removed, no flow can pass through either (u,v) or (u',v').

$$\min \quad \sum_{(u,v)\in E} \mu_{uv} y_{uv} + \sum_{w\in V} \mu_w y_w \tag{6.9}$$

$$\text{s.t.} \quad p_v - p_u + y_{uv} \geq 0, \quad \forall (u,v)\in E, \tag{6.10}$$

$$p_{v'} - p_{u'} + y_{uv} \geq 0, \quad \forall (u,v)\in E, \tag{6.11}$$

$$-p_w + p_{w'} + y_w \geq 0, \quad \forall w \in V, \tag{6.12}$$

$$p_s - p_{t'} \geq 1, \tag{6.13}$$

$$y_{uv} \in \{0,1\}, \quad \forall (u,v)\in E,$$

$$y_w \in \{0,1\}, \quad \forall w \in V.$$

To obtain the minimum computation cut, it suffices to set $y_{uv} = 0$ for all $(u, v) \in E$, and then compute the optimal solution to the integer program. The pathes from s to t' are disconnected by removing computation resources, represented by the links across two layers of graphs. To obtain the minimum communication cut, it suffices to set $z_w = 0$ for all $w \in V$, and then compute the optimal solution to the integer program.

Since it is inefficient to compute the optimal solution of an integer program, we next develop a polynomial time algorithm for evaluating the minimum computation cut, and approximation algorithms for evaluating the minimum communication cut and the joint cut.

Minimum computation cut: Since a flow needs to be processed by computation nodes along the paths from the source to the destination, removing all the computation resources along $s - t$ paths is sufficient and necessary to reduce the flow to zero. Such computation resources can be identified by computing the intersection of the set of nodes reachable from the source and the set of nodes that can reach the destination. Both sets can be computed by depth first search. The algorithm is summarized as follows, with time complexity $O(|E|)$.

Algorithm 6.1 Algorithm for evaluating the minimum computation cut for an $s - t$ pair

---

1. Find the set of nodes V_s such that there exists at least one path from s to every node in V_s.
2. Find the set of nodes V_t such that there exists at least one path from every node in V_t to t.
3. The minimum computation cut for the st pair is $V_s \cap V_t$.

Minimum communication cut: If there is a single computation node u, then the minimum communication cut is the minimum of 1) the minimum cut that disconnects s and u, and 2) the minimum cut that disconnects u and t.

However, if there is more than one computation node, computing the minimum communication cut is NP-hard (Lemma 6.1). Besides the integer program (6.9),

we develop a 2-approximation algorithm, which outputs a communication cut whose value is at most twice the minimum communication cut.

Algorithm 6.2 Approximation algorithm for the minimum communication cut for an $s-t$ pair

---

1. Construct a layered graph. Assign an arbitrarily high cost to every link across two layers. Assign μ_{uv} cost to links (u, v) and (u', v').
2. Compute a minimum cut C that separates s and t'.
3. The communication cut include links $\{(u, v)|(u, v) \in C \text{ or } (u', v') \in C\}$.

Theorem 6.2. *The communication cut obtained by Algorithm 6.2 is at most twice the minimum communication cut.*

Proof. Let S^* be the minimum $s-t$ communication cut, which has value w. By Lemma 6.2, in the layered graph, removing edges $S' = \{(u, v), (u', v')|(u, v) \in S^*\}$ disconnects s and t'. The cost of S' in the layered graph is at most $2w$.

The minimum communication cut C obtained by Algorithm 6.2 has value at most $2w$, since C is the minimum cut in the layered graph and is no larger than S'. The cost of removing links $L = \{(u, v)|(u, v) \in C \text{ or } (u', v') \in C\}$ is no more than the cost of removing C. Therefore, the value of L is at most twice the value of the minimum communication cut. □

Minimum joint communication and computation cut: Algorithm 6.2 can be modified to compute a joint cut whose value is at most twice the minimum joint cut. In the first step of Algorithm 6.2, instead of assigning an arbitrarily high cost to links across two layers, μ_w cost is assigned to link (w, w'). Using a similar proof to the proof of Theorem 6.2, we obtain the performance of the modified algorithm.

Theorem 6.3. *The joint communication and computation cut obtained by the modified algorithm is at most twice the minimum joint cut.*

6.2.3 Relationship between max-flow and min-cuts

The classical max-flow min-cut theorem states that the maximum amount of flow from s to t equals the value of the minimum cut that separates s and t. In a computing network, we study the connections between flow and various types of cuts. Since either communication or computation can be the bottleneck to support a flow, the gap between the maximum flow and the minimum communication cut or the minimum computation cut can be unbounded. For example, Fig. 6-3 illustrates that the gap between the minimum communication cut and the maximum flow can grow arbitrarily large as the communication bandwidth increases while the computation units stay the same, where the numbers adjacent to links and nodes represent the communication capacity and computation capacity, respectively. Similarly, Fig. 6-4 illustrates that the gap between the minimum computation cut and the maximum flow can be arbitrarily large.

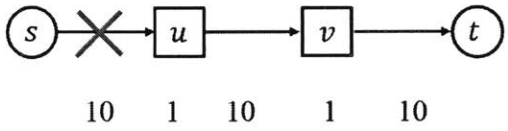


Figure 6-3: Gap between the maximum flow and minimum communication cut: maximum flow = 2, minimum communication cut = 10.

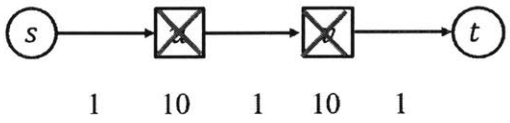


Figure 6-4: Gap between the maximum flow and minimum computation cut: maximum flow = 1, minimum computation cut = 20.

Since the joint communication and computation cut include pure communication cut and pure computation cut as special cases, the minimum joint cut is at most the smaller of the two pure cuts. In Fig. 6-3, the minimum joint cut is 2, by removing the two units computation resources, while in Fig. 6-4, the minimum joint cut is 1, by removing either of the two communication links. Note that the joint cut can be smaller than both pure cuts. For example, consider two paths in parallel between s

and t, illustrated by Figs. 6-3 and 6-4, respectively. The minimum joint cut is 3, while the minimum communication cut is 11 and the minimum computation cut is 22.

The following theorem bounds the gap between the maximum flow and the minimum joint cut.

Theorem 6.4. *The minimum joint communication and computation cut is at most twice the maximum flow between a source-destination pair.*

Proof. In the layered graph, the sum of flows on two copies of a communication link should not exceed the capacity of the link. By relaxing the capacity constraints, and restricting that the flow on each copy of the link should not exceed the capacity of the link, we obtain a *modified* layered graph. Since the sum of flows in the two copies of a link is at most twice the link capacity, the capacity constraints in the original graph are satisfied by reducing the flow by half in the modified layered graph. Therefore, the maximum flow in the modified layered graph is at most twice the maximum flow in the original graph.

The minimum cut in the modified layered graph is the same as the maximum flow in the modified layered graph. The minimum joint cut in the original graph is at most the minimum cut in the modified layered graph, since removing two copies of a link incurs double cost in the modified layered graph and a single cost in the original graph. Therefore, the minimum joint cut in the original graph is at most twice the maximum flow in the original graph. □

The gap is shown to be tight by the example in Fig. 6-5. In this computing network, each link has capacity 2. Node v is the only computation node with processing capacity 2. The maximum $s-t$ flow is 1, while the minimum $s-t$ joint cut is 2.

6.3 Flow interdiction

In this section, we study a network flow interdiction problem in a computing network. The objective is to minimize the flow by removing communication links and

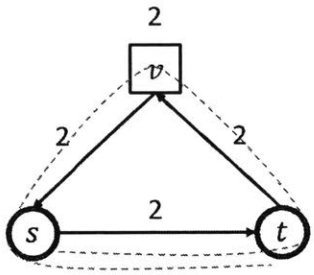


Figure 6-5: Gap between the maximum flow and minimum joint cut: maximum flow = 1, minimum joint cut = 2.

computation resources using a given budget. We formulate mathematical programs to compute an optimal interdiction strategy that minimizes the maximum $s-t$ flow in the general case.

Using linear programming duality, the maximum flow is equivalent to the minimum cut without integrality constraints. Let z_{uv} indicate whether link (u,v) is removed. Let z_w indicate whether the computation resource at node w is removed. Let c_{uv} denote the cost of removing link (u,v). Let c_w denote the cost of removing the computation resource at node w. Let B denote the interdiction budget. The objective (6.14) minimizes the maximum flow after interdiction. The budget constraint is guaranteed by Eq. (6.15). The detailed derivation of this formulation is in the Appendix.

$$\min \quad \sum_{(u,v)\in E} \mu_{uv}\beta_{uv} + \sum_{w\in V} \mu_w \beta_w \tag{6.14}$$

$$\begin{aligned}
\text{s.t.} \quad & p_v - p_u + \beta_{uv} + z_{uv} \geq 0, \quad \forall (u,v) \in E \\
& p_{v'} - p_{u'} + \beta_{uv} + z_{uv} \geq 0, \quad \forall (u,v) \in E \\
& -p_w + p_{w'} + \beta_w + z_w \geq 0, \quad \forall w \in V \\
& p_s - p_{t'} \geq 1, \\
& \sum_{(u,v)\in E} c_{uv} z_{uv} + \sum_{w\in V} c_w z_w \leq B, \qquad (6.15) \\
& 0 \leq \beta_{uv} \leq 1, z_{uv} \in \{0,1\}, \quad \forall (u,v) \in E,
\end{aligned}$$

$$0 \leq \beta_w \leq 1, z_w \in \{0,1\}, \quad \forall w \in V.$$

Network flow interdiction problem is NP-hard in general even for the classical flow network. Clearly, the interdiction problem for a computing network is also NP-hard, since it includes the classical interdiction problem as a special case. The problem becomes tractable for the classical flow network when the interdiction cost equals link capacity, and fractional interdiction is allowed. The optimal solution is to compute the minimum cut and remove full or partial links in the cut. However, such an algorithm would not work for a computing network, since the maximum flow no longer equals the minimum cut, and removing two links with the identical capacity in an $s-t$ cut may reduce the maximum $s-t$ flow by different values.

6.4 Numerical results

In this section, we provides numerical examples based on the Abilene network topology to illustrate the network robustness by applying our algorithms to evaluate the robustness metrics. Since we study directed graphs throughout the chapter, we consider two directed links (in both directions) parallel to each link in Fig. 6-6.

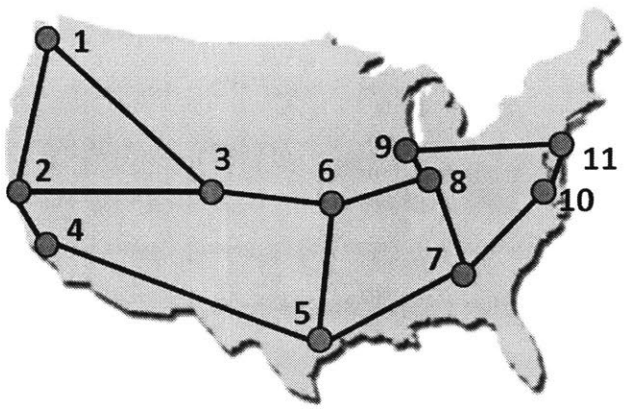


Figure 6-6: Abilene network topology.

6.4.1 Max-flow and min-cuts

The maximum flow equals the minimum computation cut if computation resource is the bottleneck to support a network flow. Suppose that each directed link has transmission capacity 1, and that each of nodes 6 and 11 has processing capacity 0.5. The maximum flow between each pair of nodes is 1, which matches the value of minimum computation cut (*i.e.*, removing both computation resources at 6 and 11).

There is a non-zero gap between the maximum flow and the minimum communication cut if communication resource is the bottleneck. Suppose that each processing capacity of nodes 6 and 11 is increased to 5. The minimum cut for $s = 8, t = 7$ is 3, while the maximum flow is 2.5. The flow can be decomposed as follows. One unit flow is transmitted through $8-6-5-7$ and processed at 6. One unit flow is transmitted through $8-9-11-10-7$ and processed at 11. Half unit flow is transmitted through $8-7-5-6-8-7$ (or $8-7-10-11-9-8-7$) and processed at 6 (or 11). Part of the flow has to traverse the same link $8-7$ twice, once before processing and once after processing.

In the above two examples, the minimum joint cut equals the minimum of the pure communication cut and pure computation cut. By setting the processing capacity of nodes 6 and 11 to be 5 and 0.5, respectively, for $s = 8, t = 7$, the minimum joint cut is 2.5, smaller than both the minimum communication cut 3 and the minimum computation cut 5.5. In this example, the maximum $s-t$ flow is 2.25. One feasible decomposition of the flow is one unit flow through $8-6-5-7$ processed at 6, half unit flow through $8-9-11-10-7$ processed at 11, half unit flow through $8-9-11-10-7-5-6-9-8-7$ processed at 6, and 0.25 unit flow through $8-7-5-6-9-8-7$ processed at 6.

6.4.2 Flow interdiction

We then study flow interdiction using randomly generated capacities, in order to illustrate the properties of network flow interdiction and discuss instances with different levels of interdiction difficulty. For simplicity, the capacity of each link

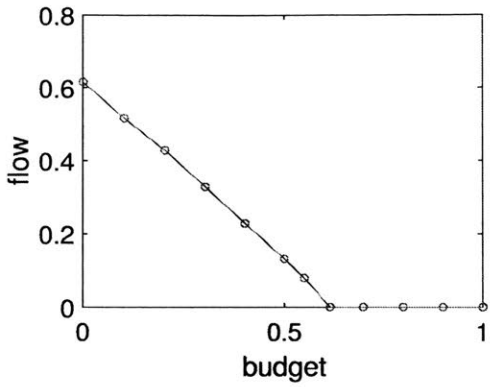
is independently and uniformly chosen from $(0, 1)$. The capacity of each node is independently and uniformly chosen from $(0, 0.1)$.

First, we consider the network flow interdiction problem where the cost of interdiction equals the capacity. For $s = 1, t = 2$, the values of flows after optimal interdictions are represented by Fig. 6-7a. The curve is smooth, because computation resource is the bottleneck for the flow from node 1 to node 2 and computation capacity has finer granularity due to the small random number generations range. For $s = 1, t = 10$, the values of flows after optimal interdictions are represented by Fig. 6-7b. The curve is non-smooth, because communication resource is the bottleneck for the flow from node 1 to node 10 and the cost of removing a link is relatively high. The steps in the curve illustrates that the interdiction problem has the same nature as the knapsack problem where the knapsack size represents the budget and item sizes represent link capacities.

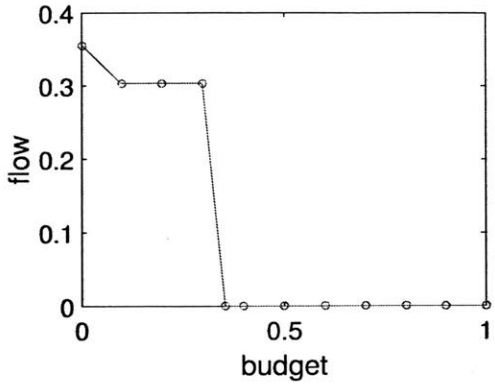
Then, we consider interdiction costs that are independent of the capacities. The cost of removing each link is independently and uniformly chosen from $(0, 1)$. The cost of removing the computation resource at each node is independently and uniformly chosen from $(0, 0.1)$. For $s = 1, t = 2$, the values of flows after optimal interdictions are represented by Fig. 6-7c. The curve is steeper for small budgets compared with Fig. 6-7a, since it is possible to remove large computation resource at small cost due to the independence between cost and capacity. For $s = 1, t = 10$, the values of flows after optimal interdictions are represented by Fig. 6-7d.

6.5 Summary

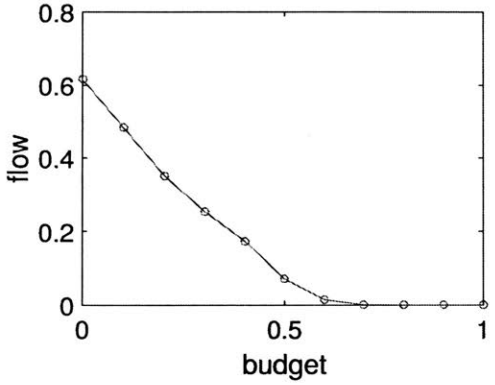
We studied the robustness of distributed computing networks where flows require communication and computation resources to be transmitted and processed. We defined cut metrics to evaluate network robustness under the failures of communication and computation resources. We developed algorithms to evaluate the max-flow and the min-cuts, and showed a non-zero gap between them. Moreover, we developed algorithms for optimal flow interdiction by removing communication and computation



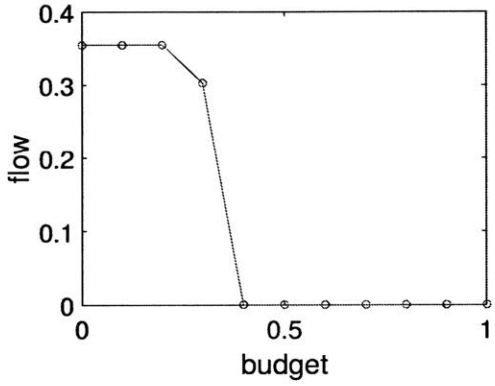
(a) Cost equals capacity, $s = 1, t = 2$.



(b) Cost equals capacity, $s = 1, t = 10$.



(c) Cost independent of capacity, $s = 1, t = 2$.



(d) Cost independent of capacity, $s = 1, t = 10$.

resources within a given budget.

6.6 Chapter appendix

6.6.1 Computational complexity

Proof of Lemma 6.1. We first prove that obtaining the minimum $s-t$ communication cut is NP-hard if there are two computation nodes, by a reduction from exact cover by 3-sets. The reduction follows a similar proof in [101] that shows multicut is NP-hard.

The exact cover by 3-sets problems is as follows. Given a set X of $3q$ elements, and a collection C of 3-element subsets of X, is there a subset $K \subseteq C$, such that every element in X appears in exactly one member of K?

We construct a graph from an instance of the exact cover by 3-sets problem. For each 3-set $c_i \in C$, there is a path $s_1 \rightarrow u_i \rightarrow v_i \rightarrow t_1$ from s_1 to t_1. The capacities of links $(s_1, u_i), (u_i, v_i), (v_i, t_1)$ are $k, 2, 1$, respectively. For each element $x \in X$, there is a path from s_2 to t_2. The path contains an edge (u_i, v_i) if the 3-set s_i contains x. All the other edges in the path from s_2 to t_2 have capacity k, except the edges (u_i, v_i).

Finally, the source node s is connected to each of s_1 and s_2 through a link of capacity k. Each of the two nodes t_1 and t_2 is connected to the destination t through a link of capacity k. The only two computation nodes are s_2 and t_1.

Suppose the links adjacent to s and t are not removed. In order for a computation path to connect s and t, either s_1 is connected to t_1, or s_2 is connected to both t_1 and t_2. If there exists an exact cover $K \subseteq C$ for X, a cut S_c can be constructed as follows. The edge (u_i, v_i) is in the cut if $s_i \in K$. The edge (v_j, t_1) is in the cut if $s_j \notin K$. The value of the cut S_c is $2q + (m - q) = m + q$, where $m = |C|$. This is the minimum cut that separates s_1 and t_1, and s_2 and $\{t_1, t_2\}$, for $k \geq 2m$. Therefore, S_c is the minimum communication cut that disconnect all computation paths from s to t.

To conclude, the minimum communication $s - t$ cut is $m + q$ if and only if there exist exact cover by 3-sets for X. The reduction can be done in polynomial time, since there are $O(q + m)$ edges and vertices. The computation of the minimum communication $s - t$ cut is NP-hard.

We illustrate the reduction using an example. Consider an exact cover by 3-sets problem where $X = \{1, 2, 3, 4, 5, 6\}$, $C = \{c_1 = \{1, 2, 3\}, c_2 = \{1, 2, 4\}, c_3 = \{3, 5, 6\}\}$. In this example, $m = 3, q = 2$. There exist an exact cover $K = \{c_2, c_3\}$ for X. The corresponding computing network is shown by Fig. 6-8. The path $s_1 \rightarrow u_i \rightarrow v_i \rightarrow t_1$ corresponds to the 3-set c_i, $\forall i \in \{1, 2, 3\}$. The path $s_2 \rightarrow u_1 \rightarrow v_1 \rightarrow u_2 \rightarrow v_2 \rightarrow t_2$ corresponds to elements 1 and 2 that appear in c_1 and c_2. The path $s_2 \rightarrow u_1 \rightarrow v_1 \rightarrow u_3 \rightarrow v_3 \rightarrow t_2$ corresponds to element 3 that appears in c_1 and c_3. The path $s_2 \rightarrow u_2 \rightarrow v_2 \rightarrow t_2$ corresponds to element 4 that appears in c_2. The path $s_2 \rightarrow u_3 \rightarrow v_3 \rightarrow t_2$ corresponds to elements 5 and 6 that appear in c_3. The thick edges each have capacity k. The numbers adjacent to the other edges indicate their capacity.

The red edges $\{(u_2, v_2), (u_3, v_3), (v_1, t_1)\}$ illustrate the minimum computation cut. The value of the minimum computation cut is $5 = m + q$. □

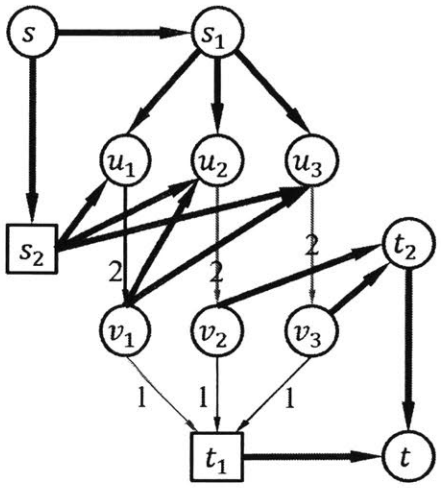


Figure 6-8: Reduction from exact cover by 3-sets to minimum communication cut.

6.6.2 Correctness of the integer linear program for flow interdiction

We prove the correctness of the integer linear program formulation for flow interdiction. The dual of the maximum flow (linear program (6.1)) is equivalent to the minimum cut (integer program (6.4)) without integrality constraints. Let z_{uv} indicate whether link (u, v) is removed. Let z_w indicate whether the computation resource at node w is removed. The maximum flow after removing links where $z_{uv} = 1$ and computation resources at nodes where $z_w = 1$ is represented by Eq. (6.16). The mathematical program (6.16) computes the maximum flow after the optimal interdiction with budget B.

$$\min \quad \sum_{(u,v)\in E} \mu_{uv}(1 - z_{uv})y_{uv} + \sum_{w\in V} \mu_w(1 - z_w)y_w \tag{6.16}$$

$$\text{s.t.} \quad p_v - p_u + y_{uv} \geq 0, \quad \forall (u, v) \in E,$$

$$p_{v'} - p_{u'} + y_{uv} \geq 0, \quad \forall (u, v) \in E, \tag{6.17}$$

$$
\begin{aligned}
& -p_w + p_{w'} + y_w \geq 0, \quad \forall w \in V, \\
& p_s - p_{t'} \geq 1, \\
& \sum_{(u,v)\in E} c_{uv} z_{uv} + \sum_{w \in V} c_w z_w \leq B, \\
& 0 \leq y_{uv} \leq 1, z_{uv} \in \{0,1\}, \quad \forall (u,v) \in E, \\
& 0 \leq y_w \leq 1, z_w \in \{0,1\}, \quad \forall w \in V.
\end{aligned}
$$

Since z_{uv} and z_v are binary, the objective can be equivalently represented by Eq. (6.18), by adding constraints Eqs. (6.19), (6.20), (6.21), (6.22). To see this, note that if $z_{uv} = 0$, $\mu_{uv}\beta_{uv} \geq \mu_{uv} y_{uv}$. In the optimal solution to the integer linear program (6.18), $\mu_{uv}\beta^*_{uv} = \mu_{uv} y^*_{uv}$, since $\mu_{uv} \geq 0$. If $z_{uv} = 1$, $y_{uv} - z_{uv} \leq 0$, and $\mu_{uv}\beta^*_{uv} = 0$ in the optimal solution. In both cases, $\beta^*_{uv} \leq 1$. Therefore, the objective $(1 - z_{uv})y_{uv}$ can be transformed to $\mu_{uv}\beta_{uv}$. Similarly, the objective $(1 - z_w)y_w$ can be transformed to $\mu_w \beta_w$. The objective Eq. (6.18) exactly matches the objective Eq. (6.16).

$$
\begin{aligned}
\min \quad & \sum_{(u,v)\in E} \mu_{uv}\beta_{uv} + \sum_{w \in V} \mu_w \beta_w && (6.18) \\
\text{s.t.} \quad & p_v - p_u + y_{uv} \geq 0, \quad \forall (u,v) \in E, \\
& p_{v'} - p_{u'} + y_{uv} \geq 0, \quad \forall (u,v) \in E, \\
& -p_w + p_{w'} + y_w \geq 0, \quad \forall w \in V, \\
& p_s - p_{t'} \geq 1, \\
& \sum_{(u,v)\in E} c_{uv} z_{uv} + \sum_{w \in V} c_w z_w \leq B, \\
& \beta_{uv} \geq y_{uv} - z_{uv}, \quad \forall (u,v) \in E, && (6.19) \\
& \beta_w \geq y_w - z_w, \quad \forall v \in V, && (6.20) \\
& 0 \leq \beta_{uv} \leq 1, \quad \forall (u,v) \in E, && (6.21) \\
& 0 \leq \beta_w \leq 1, \quad \forall w \in V, && (6.22) \\
& 0 \leq y_{uv} \leq 1, z_{uv} \in \{0,1\}, \quad \forall (u,v) \in E,
\end{aligned}
$$

$$0 \leq y_w \leq 1, z_w \in \{0,1\}, \quad \forall w \in V.$$

Finally, we show that the integer linear program (6.18) has the same optimal solution, if the constraints (6.19) and (6.20) are replaced by equality constraints.Suppose that in an optimal solution, $y^*_{uv} - z^*_{uv} \geq 0$. Then $\beta^*_{uv} = y^*_{uv} - z^*_{uv}$ holds in the optimal solution. If $y^*_{uv} - z^*_{uv} < 0$, y^*_{uv} can be increased to z^*_{uv} without violating any constraint and achieves the same cost, where $\beta^*_{uv} = 0$. Therefore, the constraint (6.19) can be replaced by an equality constraint. The same analysis holds for replacing constraint (6.20) by an equality constraint. By replacing $y_{uv} = \beta_{uv} + z_{uv}$ in all the constraints, we obtain the integer linear programming formulation (6.14).

Remark. The network flow interdiction problem in the classical communication network was formulated as an integer linear program in [95]. We follow a similar approach that use linear programming duality to transform a minimax problem to a minimization problem. The key difference is that the classical minimum cut polytope is integral, and thus it is possible to restrict values of p_v, y_{uv} to be binary in [95]. However, the polytope of Integer program (6.9) is not integral. Thus, p_v, y_{uv} may take fractional values, which complicates our analysis and makes it non-trivial to extend this formulation to study fractional interdiction problems where a fraction of link or node capacity can be removed.

Chapter 7

Concluding remarks

In this thesis, we developed theories on interdependent networks modeling, analysis, and control, and studied their applications to smart grid and edge cloud computing. The new models and analytical tools can be applied to design robust interdependent networks that can withstand failures and attacks, enabling the deployment and operation of future large-scale reliable cyber-physical systems.

In Chapter 2, we developed a layered graph model to represent interdependent networks with arbitrary topology and dependence. We defined supply node connectivity as a robustness metric for interdependent networks, developed algorithms to evaluate the connectivity, and interdependence assignment algorithms to maximize the connectivity. We extended graph algorithms and performance metrics to interdependent networks, to analyze the network robustness under failures.

In Chapter 3, we studied robust routing in interdependent networks. We developed approximation algorithms to evaluate the path failure probability under correlated node failures, and developed algorithms to compute the most reliable path. We also studied diverse routing, which significantly improves the routing reliability over single-path routing. The routing algorithms can be used to reliably transmit information or commodities through interdependent networks.

In Chapter 4, we developed an interdependent random geometric graph (RGG) model for large-scale interdependent networks. We derived the first analytical bounds on percolation thresholds of interdependent RGGs, and obtain 99% confidence inter-

vals for the percolation thresholds. As the first study on percolation of interdependent spatial network models using a mathematically rigorous approach, this new model and analytical tools provide a framework for robustness evaluation of interdependent networks under uniform random node failures, geographical attacks, and degree-dependent failures that capture non-uniform vulnerabilities of network components.

In Chapter 5, we studied power grid frequency control with limited communication in smart grids, which is an application of interdependent networks. A decentralized integral controller, with properly designed controller grains, achieves both frequency regulation and near-optimal economic dispatch without communication. We studied the benefit of communication in reducing the convergence time of the control, and quantify the importance of each individual communication link. This chapter serves as an example of the control of interdependent networks under failures.

In Chapter 6, we studied network flow interdiction problems in distributed computing networks, where flows require both communication and computation resources. We defined cut metrics that characterize the network vulnerability under the failure of network resources. We develop algorithms to compute the maximum flow and the minimum cuts in a computing network, and optimal interdiction to minimize the flow using a given budget.

Finally, we comment on future research topics in interdependent networks. We developed simple and analytically tractable models to capture key properties of interdependent networks. The models can be extended to characterize heterogeneity in network structure and dependence. For example, key nodes for network connectivity, such as nodes in a small network cut, can be supported by more supply nodes to enhance their reliability. The amount of interdependence can be characterized in finer granularity, where some nodes require a larger number of supply nodes to be functional. The intermediate functional states between fully functional and failure enrich model representability. The network structure for dependence (*e.g.*, supply nodes should include at least one node from each component) can be further studied. In summary, the future interdependent networks require the joint optimization of in-

tra and inter connections of networks, which motivates a rich class of network design problems.

The continued growth of cyber-physical systems and the increasing role of communication in system monitoring and control make it an exciting time to study the interdependence between systems and networks. The importance of identifying the vulnerability and reinforcing interdependent networks is signified by the trend of large-scale deployment of cyber-physical systems. Future works include developing protection techniques for interdependent networks, including augmentation and reinforcement for both dependence and connectivity, to contribute to the operation and maintenance of highly reliable interdependent networks.

Appendix A

Optimal control of distributed computing networks with mixed-cast traffic flows

In the appendix, we present a dynamic routing and scheduling policy for cloud networks, where traffic flows are processed by a chain of service functions. The dependencies between service functions arise in applications such as video streaming and virtual reality, where the input of one function is the output of another function. We develop stochastic control policies for packets to be processed by the functions, and delivered from sources to destinations.

Distributed cloud (computing) networks, tasked with both packet transmission and processing, require the joint optimization of communication and computation resources. Given that internet traffic is increasingly a diverse mix of unicast and multicast flows, we address the design of throughput-optimal dynamic packet processing and routing policies for mixed-cast (unicast and multicast) service chains in distributed computing networks. Our proposed control policy handles flow scaling, a prominent characteristic of traffic flows in distributed computing networks, where a flow may expand or shrink due to service function processing.

We develop a dynamic control policy that determines both routes and processing locations for packets upon their arrival at a distributed computing network. The

proposed policy, referred to as Universal Computing Network Control (UCNC), guarantees that packets i) are processed by a specified chain of service functions, ii) follow cycle-free routes between consecutive functions, and iii) are delivered to their corresponding set of destinations via proper packet duplications. UCNC is shown to be throughput-optimal for any mix of unicast and multicast traffic, and is the first throughput-optimal policy for non-unicast traffic in distributed computing networks with both communication and computation constraints. Moreover, simulation results suggest that UCNC yields substantially lower average packet delay compared with existing control policies for unicast traffic.

The rest of the chapter is organized as follows. We introduce the model in Section A.1, and characterize the capacity region in Section A.2. In Section A.3, we develop a routing policy to stabilize a virtual queuing system. In Section A.4, we prove that the same routing policy, along with a proper packet scheduling policy, is throughput-optimal for the associated computing network. Section A.5 presents numerical simulations. Section A.6 presents extensions. Section A.7 summarizes the chapter.

A.1 Model

In this section, we present models for distributed computing networks, service function chains, and mixed-cast traffic.

A.1.1 Computing network model

We consider a distributed computing network modeled as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ links. A node may represent a router, which can forward packets to neighboring nodes, or a distributed computing location, which, in addition, can host service functions for flow processing. When network flows go through a service function at a computation node, they consume computation resources (*e.g.*CPUs). We denote by μ_u the processing capacity of node $u \in \mathcal{V}$. A link represents a network connection between two nodes. When network flows go

through a link, they consume communication resources (*e.g.*bandwidth). We denote by μ_{uv} the transmission capacity of link $(u, v) \in \mathcal{E}$.

A.1.2 Service model

A service $\phi \in \Phi$ is described by a chain of M_ϕ functions (ϕ, i), $i \in \{1, \ldots, M_\phi\}$. Each function (ϕ, i) is characterized by its computation requirement $r^{(\phi,i)}$, indicating that $r^{(\phi,i)}$ computation resource units are required to process a unit input flow. Function (ϕ, i) is also characterized by a flow scaling factor $\xi^{(\phi,i)}$, indicating that the average flow rate at the output of function (ϕ, i) is $\xi^{(\phi,i)}$ times the average input flow rate. We assume function (ϕ, i) is available at a subset of computation nodes $\mathcal{N}_{(\phi,i)} \subseteq \mathcal{V}$. A flow that requires service ϕ must be processed by the functions (ϕ, i), $i \in \{1, \ldots, M_\phi\}$ in order.

Figure A-1 illustrates an example of a service function chain for video streaming. The first function in the chain is a firewall, with computation requirement $r^{(\phi,1)} = 0.1$ and flow scaling $\xi^{(\phi,1)} = 1$. The second function in the chain is a transcoding function, with computation requirement $r^{(\phi,2)} = 2$ and flow scaling $\xi^{(\phi,2)} = 0.8$. The numbers above the links indicate the flow rates at each stage of the service chain, and the numbers above the functions indicate the computation rates required to process the incoming flow.

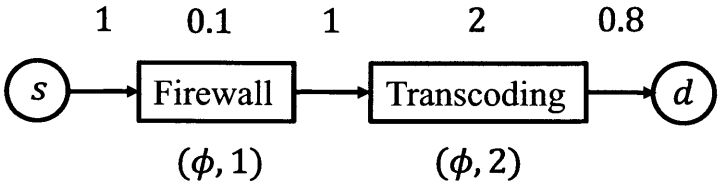


Figure A-1: An illustration of a service function chain with different function computation requirements and flow scaling.

A.1.3 Traffic model

A commodity-(c, ϕ) flow is specified by a source node s_c, a set of destination nodes $\mathcal{D}_c$, and a service ϕ. Packets of commodity-(c, ϕ) flow enter the network at s_c and exit

the network for consumption at $\mathcal{D}_c$ after being processed by the service functions in ϕ. A flow is *unicast* if $\mathcal{D}_c$ contains a single node in $\mathcal{V}$, denoted by d_c, and is *multicast* if $\mathcal{D}_c$ contains more than one node in $\mathcal{V}$. We denote by $(\mathcal{C}, \Phi)$ the set of all commodities.

We consider a time slotted system with slots normalized to integral units $t \in \{0, 1, 2, \dots\}$. We denote by $A^{(c,\phi)}(t)$ the number of exogenous arrivals of commodity-(c, ϕ) packets at node s_c during time slot t, and by $\lambda^{(c,\phi)}$ its expected value, referred to as the average arrival rate, where we assume that $A^{(c,\phi)}(t)$ is independently and identically distributed (i.i.d.) across time slots. The vector $\boldsymbol{\lambda} = \{\lambda^{(c,\phi)}, (c, \phi) \in (\mathcal{C}, \Phi)\}$ characterizes the arrival rates to the network.

A.2 Policy space and capacity region

We address the *mixed-cast service chain control problem*, where both unicast and multicast packets must be processed by a specified chain of service functions before being delivered to their associated destinations. The goal is to develop a control policy that maximizes network throughput under both communication and computation constraints.

We first transform the original problem that has both communication and computation constraints into a network flow problem in a graph that only has link capacity constraints. The transformation simplifies the representation of a flow. We then limit the routing policy space without reducing the capacity region. Finally, we characterize the network capacity region.

A.2.1 Transformation to a layered graph

Following the approach of [102], we model the flow of packets through a service chain via a layered graph, with one layer per stage of the service chain. Let $G^{(\phi)}_{\text{IntDep}} = (G^{(\phi,0)}_{\text{IntDep}}, \dots, G^{(\phi,M_\phi)}_{\text{IntDep}})$, with edge set $\mathcal{E}^{(\phi)}$ and vertex set $\mathcal{V}^{(\phi)}$, denote the layered graph associated with service chain ϕ. Each layer $G^{(\phi,i)}_{\text{IntDep}}$ is an exact copy of the original graph G_{IntDep}, used to represent the routing of packets at stage i of service ϕ, *i.e.*the routing of packets that have been processed by the first i functions of service ϕ. Let

$u^{(\phi,i)}$ denote the copy of node u in $G^{(\phi,i)}_{\text{IntDep}}$, and edge $(u^{(\phi,i)}, v^{(\phi,i)})$ the copy of link (u,v) in $G^{(\phi,i)}_{\text{IntDep}}$. Across adjacent layers, a directed edge from $u^{(\phi,i-1)}$ to $u^{(\phi,i)}$ for all $u \in \mathcal{N}_{(\phi,i)}$ is used to represent the computation of function (ϕ,i). See Fig. A-2 for an example of the layered graph.

Proposition A.1. *There is a one-to-one mapping between a flow from $s^{(\phi,0)}$ to $\mathcal{D}^{(\phi,M_\phi)}$ in $G^{(\phi)}_{IntDep}$ and a flow from s to $\mathcal{D}$ processed by ϕ in G_{IntDep}.*

Proof. Let a flow be processed by function (ϕ,i) at node $u \in \mathcal{N}_{(\phi,i)} \subseteq \mathcal{V}$. Then, by construction of the layered graph, an equivalent flow must traverse link $(u^{(\phi,i-1)}, u^{(\phi,i)}) \in \mathcal{E}^{(\phi)}$. Similarly, let a flow that has been processed by the first i functions of service ϕ traverse link $(u,v) \in \mathcal{E}$. Then, an equivalent flow must traverse link $(u^{(\phi,i)}, v^{(\phi,i)}) \in \mathcal{E}^{(\phi)}$. Under this mapping, every flow processed by ϕ in G_{IntDep} corresponds to a flow in $G^{(\phi)}_{\text{IntDep}}$, and vice versa. □

We now state generalized flow conservations laws in the layered graph that readily apply to the original graph by Proposition A.1.

Let $f_{u^{(\phi,i)}v^{(\phi,i)}}$ denote the flow rate on link $(u^{(\phi,i)}, v^{(\phi,i)})$, *i.e.*the rate of *stage-i* packets on link (u,v), where a stage-i packet is a packet that has been processed by the first i functions in ϕ, and not by functions $(\phi,i+1),\ldots,(\phi,M_\phi)$. Similarly, $f_{u^{(\phi,i-1)}u^{(\phi,i)}}$ denotes the flow rate on link $(u^{(\phi,i-1)}, u^{(\phi,i)})$, *i.e.*the computation rate at node u for processing stage-$(i-1)$ packets into stage-i packets via function (ϕ,i).

We first focus on unicast traffic, where no packet duplication is required.[1] Note that due to non-unit computation requirements and flow scalings, traditional flow conservation does not hold even for unicast traffic. For a given node $u^{(\phi,i)} \in G^{(\phi,i)}_{\text{IntDep}}$, the following *generalized flow conservation* law holds:

$$\sum_{v^{(\phi,i)} \in \mathcal{V}^{(\phi)}} f_{v^{(\phi,i)}u^{(\phi,i)}} + \frac{\xi^{(\phi,i)}}{r^{(\phi,i)}} f_{u^{(\phi,i-1)}u^{(\phi,i)}}$$

[1]Packet duplication is different from flow scaling. Flow scaling is a result of service function processing. An expanded flow, which is a function output, contains different packets. Packet duplication makes identical copies of a packet, which may be forwarded along different routes to reach different destinations.

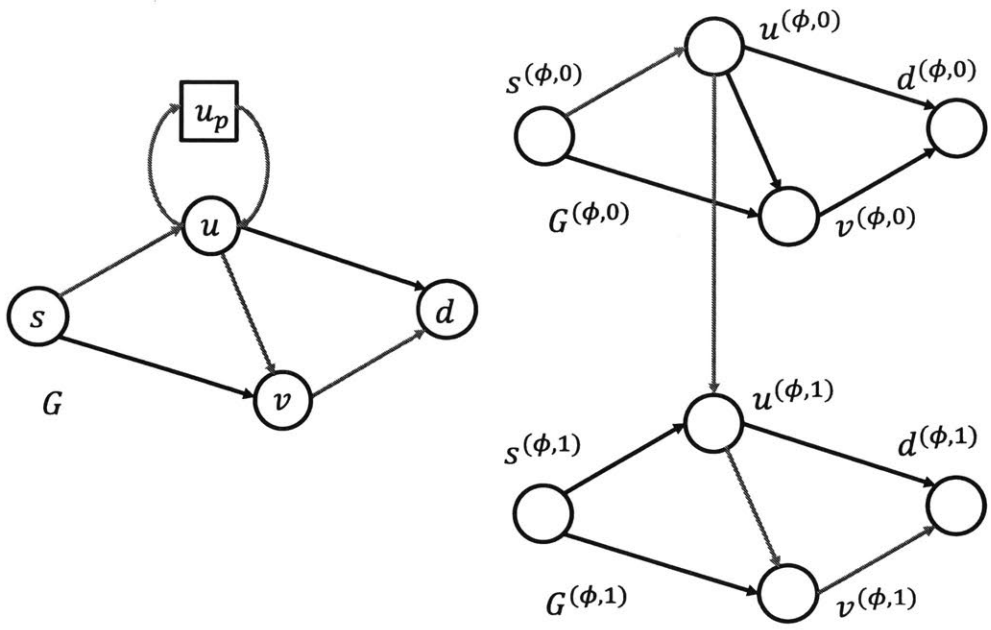


Figure A-2: The left figure is the original graph G_{IntDep}, where u is the only computation node for the single function in ϕ. A dummy node u_p and connections to u are added to illustrate the availability of service function processing at node u. The right figure is the layered graph $G^{(\phi)}_{\text{IntDep}}$.

$$= \sum_{v^{(\phi,i)} \in \mathcal{V}^{(\phi)}} f_{u^{(\phi,i)}v^{(\phi,i)}} + \frac{1}{r^{(\phi,i+1)}} f_{u^{(\phi,i)}u^{(\phi,i+1)}}. \tag{A.1}$$

In the case of multicast traffic, packet duplication is necessary for a packet to reach multiple destinations. Packet duplications can happen at any stage of a service chain. Suppose that a stage-i packet is duplicated. Then, all the copies must be processed by functions $(\phi, i+1), \ldots, (\phi, M_\phi)$ before reaching destinations in $\mathcal{D}$. Equivalently, in the layered graph $G^{(\phi)}_{\text{IntDep}}$, if a packet is duplicated at a node in $G^{(\phi,i)}_{\text{IntDep}}$, then all the copies need to travel through the links that cross the remaining $M_\phi - i$ layers before reaching a node in $\mathcal{D}^{(\phi,M_\phi)}$. The *generalized flow conservation and packet duplication* law states that generalized flow conservation (A.1) holds at the nodes where there is no packet duplication.

Given the flow rates in the layered graph and the mapping of Proposition A.1, the flow rates in the original graph can be easily derived. The communication rate on link $(u, v) \in G_{\text{IntDep}}$, computed as the sum over the flow rates on links $(u^{(\phi,i)}, v^{(\phi,i)})$,

$\forall \phi \in \Phi, i \in \{0, \ldots, M_\phi\}$, and the computation rate at node $u \in G_{\text{IntDep}}$, computed as the sum over the flow rates on links $(u^{(\phi,i-1)}, u^{(\phi,i)})$, $\forall \phi \in \Phi, i \in \{1, \ldots, M_\phi\}$ are subject to communication and computation capacity constraints:

$$\sum_{\phi \in \Phi, i \in \{0,\ldots,M_\phi\}} f_{u^{(\phi,i)}v^{(\phi,i)}} \leq \mu_{uv},$$

$$\sum_{\phi \in \Phi, i \in \{1,\ldots,M_\phi\}} f_{u^{(\phi,i-1)}u^{(\phi,i)}} \leq \mu_u.$$

A.2.2 Policy space

An admissible policy π for the mixed-cast service chain control problem consists of two actions at every time slot t.

1. **Route selection:** For a commodity-(c, ϕ) packet that originates at s_c and is destined for $\mathcal{D}_c$, choose a set of links $E^{(c,\phi)} \subseteq \mathcal{E}^{(\phi)}$, and assign a number of packets[2] on each link that satisfies the generalized conservation law for unicast traffic and the generalized conservation and duplication law for multicast traffic.

2. **Packet scheduling:** Transmit packets through every link in $\mathcal{E}$ according to a schedule that respects capacity constraints.

The set of all admissible policies is denoted by Π. The set Π includes policies that may use past and future arrival and control information.

Let $\mathcal{P}^{(c,\phi),\pi}(t)$ denote the packets that are originated at s_c, processed by ϕ, and delivered to every node in $\mathcal{D}_c$ under policy π up to time t. Let $R^{(c,\phi),\pi}(t) = |\mathcal{P}^{(c,\phi),\pi}(t)|$ denote the number of such packets. The number of packets received by a node in $\mathcal{D}_c$ is at least $\prod_{i=1}^{M_\phi} \xi^{(\phi,i)} R^{(c,\phi),\pi}(t)$ due to flow scaling. We characterize the network throughput using *arrival rates*. A policy π *supports* an arrival rate vector $\boldsymbol{\lambda}$ if

$$\liminf_{t\to\infty} \frac{R^{(c,\phi),\pi}(t)}{t} = \lambda^{(c,\phi)}, \quad \forall (c, \phi) \in (\mathcal{C}, \Phi), \text{ w.p. 1.} \tag{A.2}$$

[2]Recall that a commodity-(c, ϕ) input packet can be expanded to multiple packets due to flow scaling and packet duplication.

The network layer capacity region is the set of all supportable arrival rates.

$$\Lambda(G_{\text{IntDep}}, \mathcal{C}, \Phi) = \{\boldsymbol{\lambda} \in \mathbb{R}_+^{|\mathcal{C}||\Phi|} : \exists \pi \in \Pi \text{ supporting } \boldsymbol{\lambda}\} \tag{A.3}$$

We next restrict the set of admissible routes without reducing the capacity region. A route is *efficient* if every packet never visits the same node in $G_{\text{IntDep}}^{(\phi)}$ more than once. For example, if there is no flow scaling, a unicast packet is transmitted through a path from the source to the destination, without cycles, and a multicast packet is transmitted and duplicated through a tree that connects the source and the set of destinations. It suffices to consider efficient routes, by Lemma A.1, whose proof is in Appendix A.8.1.

Lemma A.1. *Any arrival rate $\boldsymbol{\lambda}$ in the capacity region can be supported by a policy that only uses efficient routes.*

Moreover, we further restrict the route of a unicast packet to be a *service chain path*, and the route of a multicast packet to be a *service chain Steiner tree*, without reducing the capacity region. Note that under flow scaling, one commodity-(c, ϕ) packet that originates at s_c is scaled to $\prod_{j=1}^{i-1} \xi^{(\phi,j)}$ packets at stage-$(i-1)$. To process them, function (ϕ, i) requires $x^{(\phi,i)} = r^{(\phi,i)} \prod_{j=1}^{i-1} \xi^{(\phi,j)}$ computation resource units, and outputs $w^{(\phi,i)} = \prod_{j=1}^{i} \xi^{(\phi,j)}$ packets. Let $w^{(\phi,0)} = \xi^{(\phi,0)} = 1$.

Definition A.1. A commodity-(c, ϕ) unicast packet is routed over a *service chain path* $T^{(c,\phi)}$, if

1. $T^{(c,\phi)}$ is a path from $s_c^{(\phi,0)}$ to $d_c^{(\phi,M_\phi)}$ in $G_{\text{IntDep}}^{(\phi)}$;
2. $w^{(\phi,i)}$ packets are routed over a link in $T^{(c,\phi)}$ that belongs to $G_{\text{IntDep}}^{(\phi,i)}$;
3. $x^{(\phi,i)}$ packets are routed over a link in $T^{(c,\phi)}$ that connects $G_{\text{IntDep}}^{(\phi,i-1)}$ and $G_{\text{IntDep}}^{(\phi,i)}$.

It is easy to verify that the generalized flow conservation law holds in a service chain path. Clearly, a service chain path is an efficient route, since every node in $G_{\text{IntDep}}^{(\phi)}$ is visited only once by the same packet. However, an efficient route does

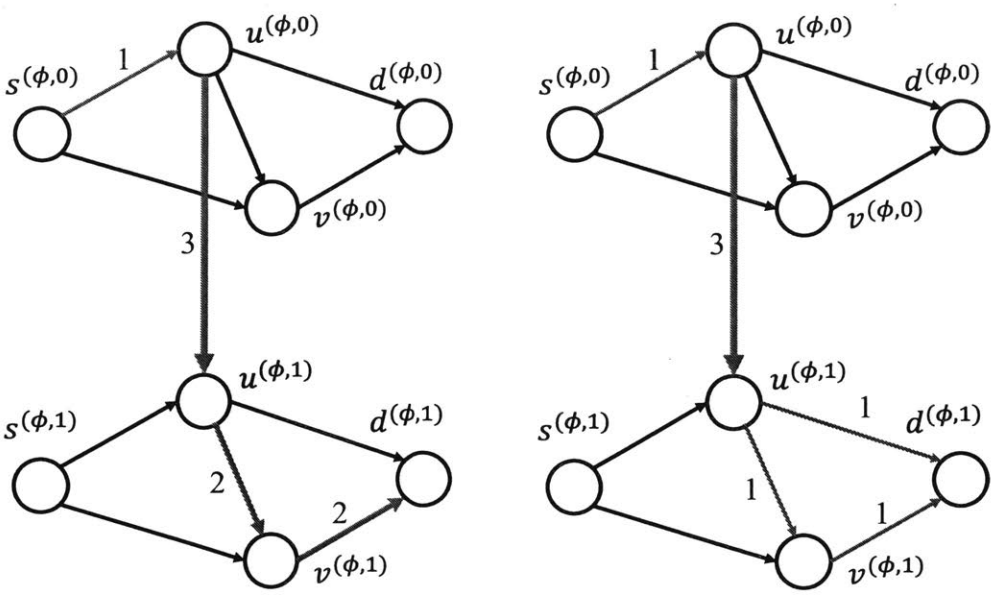


Figure A-3: The left figure illustrates a service chain path, and the right figure illustrates an alternative efficient route that is not a service chain path. The number adjacent to a link indicates the number of packets on the link. Scaling factors: $x^{(\phi,1)} = 3$; $w^{(\phi,1)} = 2$.

not have to be a service chain path. If a packet is expanded into two packets via intermediate service processing, the two packets can take different paths without violating route efficiency. For example, in Fig. A-3, the left figure illustrates a service chain path, while the right figure illustrates an efficient route that is not a service chain path.

Definition A.2. A commodity-(c, ϕ) multicast packet is routed over a *service chain Steiner tree* $T^{(c,\phi)}$, if

1. $T^{(c,\phi)}$ is a Steiner tree (arborescence) that is rooted at $s_c^{(\phi,0)}$ and connected to $\mathcal{D}_c^{(\phi,M_\phi)}$ in $G_{\text{IntDep}}^{(\phi)}$;

2. $w^{(\phi,i)}$ packets are routed over a link in $T^{(c,\phi)}$ that belongs to $G_{\text{IntDep}}^{(\phi,i)}$;

3. $x^{(\phi,i)}$ packets are routed over a link in $T^{(c,\phi)}$ that connects $G_{\text{IntDep}}^{(\phi,i-1)}$ and $G_{\text{IntDep}}^{(\phi,i)}$.

If a packet is routed over a service chain Steiner tree $T^{(c,\phi)}$, then packet duplications occur at every node that has more than one outgoing edge in $T^{(c,\phi)}$. The number

of packet duplications at a node equals its number of outgoing edges in $T^{(c,\phi)}$ minus one. The generalized flow conservation holds at all other nodes.

We conclude this section with Theorem A.1, whose proof is in Appendix A.8.1.

Theorem A.1. *There exists a policy that chooses a convex combination of service chain paths for each incoming unicast packet, and a convex combination of service chain Steiner trees for each incoming multicast packet, to support any arrival rate* $\boldsymbol{\lambda}$ *in the capacity region.*

Due to Theorem A.1, in the following, we restrict our attention to routing policies that use service chain paths or service chain Steiner trees to route incoming packets, without reducing network throughput.

A.2.3 Capacity region

For any arrival rate $\boldsymbol{\lambda} \in \Lambda(G_{\text{IntDep}}, \mathcal{C}, \phi)$, there exists an admissible policy π that takes restricted routes and supports $\boldsymbol{\lambda}$. Let $\mathcal{T}^{(c,\phi)}$ denote the set of all service chain paths (or Steiner trees) for commodity-(c,ϕ) packets. By taking the time average over the actions of π, for each commodity (c,ϕ), there exists a randomized flow decomposition and routing on $\mathcal{T}^{(c,\phi)}$. Let $\lambda_k^{(c,\phi)}$ be the average (arrival) flow rate of commodity-(c,ϕ) packets over $T_k^{(c,\phi)} \in \mathcal{T}^{(c,\phi)}$.

$$\lambda^{(c,\phi)} = \sum_{T_k^{(c,\phi)} \in \mathcal{T}^{(c,\phi)}} \lambda_k^{(c,\phi)}, \quad \forall (c,\phi) \in (\mathcal{C}, \Phi). \tag{A.4}$$

Moreover, flows should satisfy communication and computation capacity constraints. Commodity-(c,ϕ) flow contributes a rate $w^{(\phi,i)}\lambda_k^{(c,\phi)}$ on communication link (u,v) if $(u^{(\phi,i)}, v^{(\phi,i)}) \in T_k^{(c,\phi)}$, and a rate of $x^{(\phi,i)}\lambda_k^{(c,\phi)}$ on computation node u if $(u^{(\phi,i-1)}, u^{(\phi,i)}) \in T_k^{(c,\phi)}$. Let $\mathcal{S}_{uv} = \{(k,i,c,\phi) : (u^{(\phi,i)}, v^{(\phi,i)}) \in T_k^{(c,\phi)}, T_k^{(c,\phi)} \in \mathcal{T}^{(c,\phi)}, i \in \{0,\ldots,M_\phi\}, (c,\phi) \in (\mathcal{C},\Phi)\}$ denote the set of commodities that use link (u,v). Let $\mathcal{S}_u = \{(k,i,c,\phi) : (u^{(\phi,i-1)}, u^{(\phi,i)}) \in T_k^{(c,\phi)}, T_k^{(c,\phi)} \in \mathcal{T}^{(c,\phi)}, i \in \{1,\ldots,M_\phi\}, (c,\phi) \in (\mathcal{C},\Phi)\}$ denote the set of commodities that use node u. The communication and computation capacity constraints are represented by (A.5) and (A.6), respectively.

$$\sum_{(k,i,c,\phi)\in\mathcal{S}_{uv}} w^{(\phi,i)}\lambda_k^{(c,\phi)} \leq \mu_{uv}, \quad \forall (u,v)\in\mathcal{E}, \tag{A.5}$$

$$\sum_{(k,i,c,\phi)\in\mathcal{S}_{u}} x^{(\phi,i)}\lambda_k^{(c,\phi)} \leq \mu_{u}, \qquad \forall u\in\mathcal{V}. \tag{A.6}$$

To conclude, the capacity region is characterized by the arrival rates $\boldsymbol{\lambda} = \{\lambda^{(c,\phi)} : (c,\phi)\in(\mathcal{C},\Phi)\}$ that satisfy constraints (A.4), (A.5), and (A.6).

A.3 Dynamic routing in a virtual system

In this section, we study a virtual queueing system for a distributed computing network, whose simplified dynamics allows us to develop a dynamic routing algorithm that guarantees that the average arrival rate at a virtual link is no more than its service rate. We then formalize the connection between the virtual and physical systems in Section A.4.

We consider a virtual queueing system $\{\tilde{Q}_{uv}(t), \forall (u,v)\in\mathcal{E}\}$ and $\{\tilde{Q}_u(t), \forall u\in\mathcal{V}\}$ for network G_{IntDep}. We then define virtual queues for the links in the layered graphs $G^{(\phi)}_{\text{IntDep}}, \forall\phi\in\Phi$ such that the queue length of the communication links $(u^{(\phi,i)}, v^{(\phi,i)})$, $\forall\phi\in\Phi, i\in\{0,\ldots,M_\phi\}$ is equal to $\tilde{Q}_{uv}(t)$ for all t, and the queue length of the computation links $(u^{(\phi,i-1)}, u^{(\phi,i)})$, $\forall\phi\in\Phi, i\in\{1,\ldots,M_\phi\}$ is equal to $\tilde{Q}_u(t)$ for all t.

In contrast to the physical system, in which packets travel through the links in its route sequentially, in the virtual system, a packet immediately enters the virtual queues of all the links in its route, upon arrival at the network. The number of packets that arrive at the communication queue $\tilde{Q}_{uv}$ at time t, denoted by $A_{uv}(t)$, is the sum of the number of packets routed on $(u^{(\phi,i)}, v^{(\phi,i)})$, $\forall\phi\in\Phi, i\in\{0,\ldots,M_\phi\}$ at time t. Similarly, the number of packets $A_u(t)$ that arrive at the computation queue $\tilde{Q}_u$ at time t is the sum of the number of packets routed on $(u^{(\phi,i-1)}, u^{(\phi,i)})$, $\forall\phi\in\Phi, i\in\{1,\ldots,M_\phi\}$ at time t. The value $A_{uv}(t)$ indicates the total number of packets that *will be transmitted* through link (u,v), in order to serve the packets (and

their associated packets after processing) that arrive at time t, based on the routing decision. The value $A_u(t)$ indicates the total amount of computation that node u will use to process these packets. The departure rate of the packets in $\tilde{Q}_{uv}$ is equal to the transmission capacity of link (u, v), μ_{uv}, and the departure rate of the packets in $\tilde{Q}_u$ is equal to the processing capacity of node u, μ_u.

We study the queueing dynamics under a policy that routes all the packets that belong to the same commodity and arrive at the same time, through a service chain path or service chain Steiner tree. Let $A^{(c,\phi)}(t)$ be the number of commodity-(c, ϕ) packets that arrive at the network at time t. Let $T^{(c,\phi),\pi}$ denote the path or tree chosen under policy π at time t. Let $A_{uv}^{(c,\phi),\pi}(t)$ denote the number of packets that arrive at the virtual communication queue (u, v) at time t. Recall that $w^{(\phi,i)}$ and $x^{(\phi,i)}$ were defined before Definition A.1 in Section A.2.

$$A_{uv}^{(c,\phi),\pi}(t) = \sum_{(u^{(\phi,i)},v^{(\phi,i)})\in T^{(c,\phi),\pi}} w^{(\phi,i)} A^{(c,\phi)}(t). \tag{A.7}$$

Let $A_u^{(c,\phi),\pi}(t)$ denote the number of packets that arrive at the virtual computation queue at u at time t.

$$A_u^{(c,\phi),\pi}(t) = \sum_{(u^{(\phi,i-1)},u^{(\phi,i)})\in T^{(c,\phi),\pi}} x^{(\phi,i)} A^{(c,\phi)}(t). \tag{A.8}$$

The virtual queue lengths $\tilde{Q}_{uv}(t)$ and $\tilde{Q}_u(t)$ evolve according to the following recursion, where $(a)^+ = \max(a, 0)$.

$$\begin{aligned}
\tilde{Q}_{uv}(t+1) &= \left(\tilde{Q}_{uv}(t) + \sum_{(c,\phi)\in(\mathcal{C},\Phi)} A_{uv}^{(c,\phi),\pi}(t) - \mu_{uv}\right)^+, \\
\tilde{Q}_u(t+1) &= \left(\tilde{Q}_u(t) + \sum_{(c,\phi)\in(\mathcal{C},\Phi)} A_u^{(c,\phi),\pi}(t) - \mu_u\right)^+.
\end{aligned}$$

Dynamic routing policy π^*: When $A^{(c,\phi)}(t)$ packets arrive at time t, policy π^*

chooses a route $T^{(c,\phi),\pi^*}$ by minimizing

$$\sum_{(u,v)\in\mathcal{E}} \tilde{Q}_{uv}(t)A_{uv}^{(c,\phi),\pi}(t) + \sum_{u\in\mathcal{V}} \tilde{Q}_u(t)A_u^{(c,\phi),\pi}(t)$$
$$= A^{(c,\phi)}(t)\Bigg(\sum_{(u^{(\phi,i)},v^{(\phi,i)})\in\mathcal{E}^{(\phi)}} w^{(\phi,i)}\tilde{Q}_{uv}(t)1\{(u^{(\phi,i)},v^{(\phi,i)})\in T^{(c,\phi),\pi}\}$$
$$+ \sum_{(u^{(\phi,i-1)},u^{(\phi,i)})\in\mathcal{E}^{(\phi)}} x^{(\phi,i)}\tilde{Q}_u(t)1\{(u^{(\phi,i-1)},u^{(\phi,i)})\in T^{(c,\phi),\pi}\}\Bigg). \tag{A.9}$$

Let the length of link $(u^{(\phi,i)}, v^{(\phi,i)})$ be $w^{(\phi,i)}\tilde{Q}_{uv}(t)$, and the length of link $(u^{(\phi,i-1)}, u^{(\phi,i)})$ be $x^{(\phi,i)}\tilde{Q}_u(t)$. For unicast traffic, the optimal path is the *shortest path* from $s^{(\phi,0)}$ to $d^{(\phi,M_\phi)}$. For multicast traffic, the optimal tree is the *minimum Steiner tree* from $s^{(\phi,0)}$ to $\mathcal{D}^{(\phi,M_\phi)}$.

Policy π^* stabilizes the virtual system for any arrival rate in the interior of the capacity region.

Theorem A.2. *Under routing policy π^*, the virtual queue process $\{\tilde{Q}(t)\}_{t\geq 0}$ is strongly stable for any arrival rate that is in the interior of the capacity region.* I.e.

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\Bigg(\sum_{(u,v)\in\mathcal{E}} \mathbb{E}\tilde{Q}_{uv}(t) + \sum_{u\in\mathcal{V}} \mathbb{E}\tilde{Q}_u(t)\Bigg) < \infty.$$

The proof of Theorem A.2 is based on Lyapunov drift analysis and can be found in Appendix A.8.2. The queue stability implies that the arrival rate at each virtual queue is no more than its service rate.

A.4 Control of the physical network

In this section, we formalize the connection between the virtual system and the physical system, and develop a throughput-optimal control policy for a distributed computing network. Recall that an admissible policy consists of two actions at every time slot: 1) route selection, 2) packet scheduling.

The route selection for an incoming packet to the network is identical to the route

selection π^* in the virtual system. Suppose that a packet is served (*i.e.*both processed by all the service functions and delivered to the destination) by the network. The amount of traffic that the packet contributes to a physical queue Q_{uv} (or Q_u) is the same as the amount of traffic that it contributes to the virtual queue $\tilde{Q}_{uv}$ (or $\tilde{Q}_u$). Strong stability of virtual queues implies that the average arrival rate is at most the service rate of each virtual queue under π^*. Therefore, by applying the same routing policy to the physical system, the average arrival rate (or offered load) is at most the service rate for each physical queue. The statement is made precise in the proof of Theorem A.3.

A packet scheduling policy chooses a packet to transmit over a link or to process at a node, when there are more than one packet awaiting service. It was proved in [103, 104] that an extended nearest-to-origin (ENTO) policy guarantees queue stability, as long as the average arrival rate is no more than the service rate at each queue. The ENTO policy gives higher priority to packets that have traveled a smaller number of hops (*i.e.*closer to their origins). A duplicated packet (in multicast) inherits the hop count of the original packet. In the proof of Theorem A.3, we show that this policy guarantees the stability of physical queues even with flow scaling (*i.e.*one packet processed by a first queue may enter a second queue in the form of multiple packets).

The resulting routing and scheduling policy, referred to as Universal Computing Network Control (UCNC), is summarized in Algorithm A.1.

In Step 2, a commodity-(c, ϕ) packet enters the physical network and will be transmitted and processed in G_{IntDep} according to $T^{(c,\phi),\pi^*} \subseteq G^{(\phi)}_{\text{IntDep}}$ by the mapping in Proposition A.1. To implement the algorithm, the packet stores $T^{(c,\phi),\pi^*}$. At time slot $t' \geq t$, if it has been processed by the first i functions and is at node u, then it enters the physical queue for link (u, v) if $(u^{(\phi,i)}, v^{(\phi,i)}) \in T^{(c,\phi),\pi^*}$. It enters the computation queue at node u if $(u^{(\phi,i)}, u^{(\phi,i+1)}) \in T^{(c,\phi),\pi^*}$. The packet is duplicated (for multicast) if $u^{(\phi,i)}$ has more than one outgoing edge in $T^{(c,\phi),\pi^*}$.

Theorem A.3. *Under UCNC, all physical queues are rate stable for any arrival rate*

Algorithm A.1 Universal Computing Network Control (UCNC).

Initialization: $\tilde{Q}_{uv}(0) = \tilde{Q}_u(0) = 0, \forall (u,v) \in \mathcal{E}, u \in \mathcal{V}$.

At each time slot t:

1. **Preprocessing**. For an incoming commodity-(c,ϕ) packet, construct a layered graph $G^{(\phi)}_{\text{IntDep}}$. Let the cost of link $(u^{(\phi,i)}, v^{(\phi,i)})$ be $w^{(\phi,i)}\tilde{Q}_{uv}(t)$, and the cost of link $(u^{(\phi,i-1)}, u^{(\phi,i)})$ be $x^{(\phi,i)}\tilde{Q}_u(t)$.

2. **Route Selection (π^*)**. Compute a minimum-cost route $T^{(c,\phi),\pi^*}$ for a commodity-(c,ϕ) incoming packet. The packet will follow $T^{(c,\phi),\pi^*}$ for transmission and processing.

3. **Packet Scheduling (ENTO)**. Each physical link transmits packets and each computation node processes packets according to the ENTO policy.

4. **Virtual Queues Update**.

$$\begin{aligned}
\tilde{Q}_{uv}(t+1) &= \Big(\tilde{Q}_{uv}(t) + \sum_{(c,\phi)\in(\mathcal{C},\Phi)} A^{(c,\phi),\pi^*}_{uv}(t) - \mu_{uv}\Big)^+; \\
\tilde{Q}_u(t+1) &= \Big(\tilde{Q}_u(t) + \sum_{(c,\phi)\in(\mathcal{C},\Phi)} A^{(c,\phi),\pi^*}_{u}(t) - \mu_{u}\Big)^+.
\end{aligned}$$

in the interior of the capacity region. I.e.

$$\begin{aligned}
\lim_{t\to\infty} \frac{Q_{uv}(t)}{t} &= 0, \quad w.p.\ 1,\ \forall (u,v) \in \mathcal{E}; \\
\lim_{t\to\infty} \frac{Q_u(t)}{t} &= 0, \quad w.p.\ 1,\ \forall u \in \mathcal{V}.
\end{aligned}$$

The proof can be found in Appendix A.8.3 and consists of two parts. The first part is to prove that the average arrival rate is no more than the service rate of every link and every computation node. The second part is to prove that under this condition, the physical queues are stable under the ENTO policy. Using standard queue stability analysis (*e.g.* [103]), we conclude that the policy is throughput-optimal.

A.5 Simulation results

In this section, we evaluate the performance of UCNC in a distributed computing network based on the Abilene network topology in Fig. A-4. For simplicity, we assume that each link is bidirectional and has unit transmission capacity in each direction. We evaluate the performance of UCNC for unicast traffic in Section A.5.1, and for multicast traffic in Section A.5.2. In Sections A.5.1 and A.5.2, we consider a small number of commodities, and assume that nodes 3 and 8 have unit computation capacity and that all the other nodes have zero computation capacity. In Section A.5.3, we consider a larger number of commodities with a mix of unicast and multicast.

For unicast traffic, we compare UCNC with the backpressure-based algorithm in [93]. While both algorithms are throughput-optimal, UCNC yields much shorter packet delay. We also compare UCNC with heuristic policies such as choosing the closest server to process the service functions, and observe that the heuristic policies are not always throughput-optimal. This demonstrates the importance of joint optimization of communication and computation resources.

For multicast traffic, we illustrate the performance of UCNC, and compare the capacity region under multicast traffic with the capacity region when multicast flows are treated as multiple unicast flows. Numerical results indicate the ability to deliver higher rates when multicast traffic can be served via proper packet duplications, as opposed to creating independent copies for each destination. This confirms the importance of the first throughput-optimal algorithm for multicast traffic in distributed computing networks.

We compare different policies using the average delay metric. Note that we did not claim any theoretical delay guarantee of UCNC (other than $o(t)$ delay with probability 1 due to Little's law and Theorem A.3). Nevertheless, the delay metric is important for quality of service. Moreover, queue lengths can be inferred from delay information. Small delays indicate short queue lengths and therefore stable queues. Thus, we can infer the capacity region under different policies using delay information.

A.5.1 Unicast traffic

Comparison with backpressure-based algorithm

We consider two commodities of unicast traffic. The first commodity originates at node 1 and is destined for node 11. The second commodity originates at node 4 and is destined for node 7. Packets in both commodities are processed by two functions in a service chain. Let λ_1 and λ_2 denote the expected arrival rates of the two commodities, respectively. Ignoring all the scalings ($\xi = r = 1$), the computation resource constraints are tight to support $\lambda_1 + \lambda_2 = 1$. Thus, the capacity region is $\lambda_1 + \lambda_2 \leq 1$. Figure A-5 compares the average packet delays under UCNC and the backpressure-based algorithm, for different arrival rates that satisfy $\lambda_1 = \lambda_2$. We observe that the average packet delays under UCNC are significantly lower than the delays under the backpressure-based algorithm.

Comparison with nearest-to-destination service function placement

We compare the performance of UCNC with the heuristic of placing the service functions in the computation node that is nearest to the destination. For a fair comparison, the processing capacity of a single node should be sufficient. We consider a single unicast commodity from node 2 to node 7. The service chain ϕ has a single function $(\phi, 1)$ with flow scaling factor $\xi^{(\phi,1)} = 1/3$ and computation requirement $r^{(\phi,1)} = 1/3$. The heuristic policy routes the packets from node 2 to node 8, which is the closest computation node to node 7, processes the packets at node 8, and routes the processed packets from node 8 to node 7. The average packet delays under both algorithms are compared in Fig. A-6. Due to communication constraints, the maximum rate that UCNC can support is $\lambda = 3$, while the maximum rate that the heuristic policy can support is $\lambda = 2$. The heuristic policy fails to be throughput-optimal when there is flow scaling (shrinkage) due to processing. This demonstrates the importance of jointly optimizing communication and computation resources.

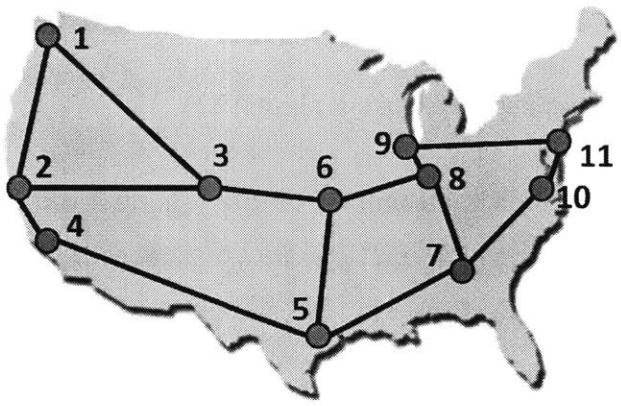


Figure A-4: Abilene network topology.

Comparison with nearest-to-source service function placement

Placing a service function at the nearest-to-source computation node may decrease the supportable service rate, when there is flow expansion. We consider a single commodity from node 2 to node 7. The service chain ϕ has a single function $(\phi, 1)$ with flow scaling factor $\xi^{(\phi,1)} = 3$ and computation requirement $r^{(\phi,1)} = 1$. The heuristic policy routes the packets from node 2 to node 3, which is the closest computation node to the source, processes the packets at node 3, and then routes the processed packets from node 3 to node 7. The maximum flow rate from node 3 to node 7 is two. Thus, the maximum supportable service rate is $\lambda = 2/3$, which expands to a flow of rate two after processing. In contrast, illustrated in Fig. A-7, UCNC is able to support a service rate $\lambda = 1$. This, again, demonstrates the need to jointly optimize communication and computation resources.

A.5.2 Multicast traffic

We next study a multicast flow from node 1 to nodes 7 and 11. Suppose that the service chain has two functions and that all the scaling factors ξ, r are one. The optimal policy is to process the packets at both nodes 3 and 8, and then duplicate the processed packets and route them to the two destinations. The maximum supportable service rate is $\lambda = 1$ for both destinations. In contrast, if the multicast flow is treated as two unicast flows, then the sum of the service rates to both destinations is one.

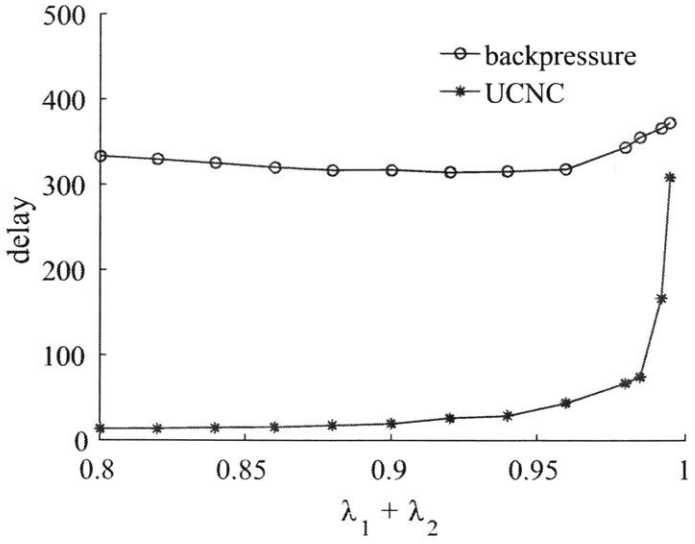


Figure A-5: Comparison of average packet delay under UCNC and the delay under the backpressure algorithm.

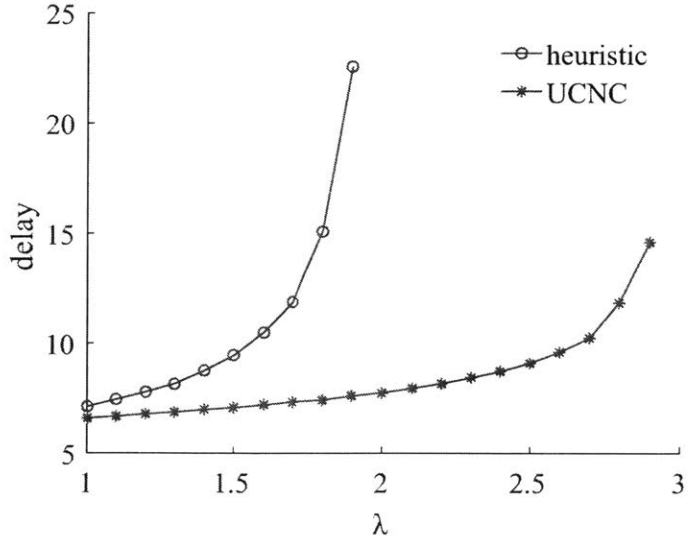


Figure A-6: Comparison of average packet delay under UCNC and the nearest-to-destination service function placement heuristic.

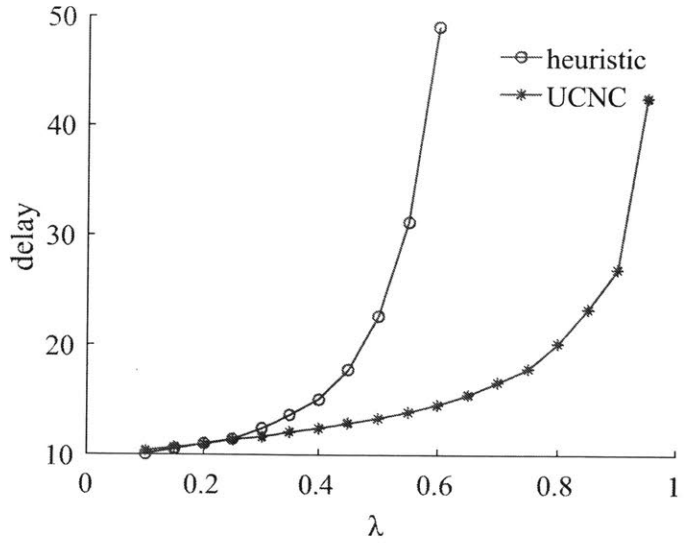


Figure A-7: Comparison of average packet delay under UCNC and the nearest-to-source service function placement heuristic.

Thus, multicasting improves the performance of the distributed computing network. As shown in Fig. A-8, UCNC is throughput-optimal for multicast traffic, and the average packet delays are small.

A.5.3 Large scale simulation

We evaluate the performance of UCNC under a large number of commodities. We consider three service chains $\Phi = \{\phi_1, \phi_2, \phi_3\}$. Services ϕ_1, ϕ_2 have two functions each, and ϕ_3 has three functions. The scaling factors ξ, r are chosen independently from a uniform distribution in $[0.5, 2]$. Each service chain processes four unicast flows and two multicast flows, where the source and the destination(s) of each flow are randomly chosen among all nodes that are at least two hops away. Thus, there are a total of 18 commodities. Each function can be computed at four randomly chosen computation nodes, each of which has unit capacity.

The average packet delays under the 18 mixed-cast commodities are shown in Fig. A-9, where all commodities have identical arrival rate λ. We observe that UCNC is able to support rate $\lambda = 0.12$. In contrast, when each multicast flow is treated as

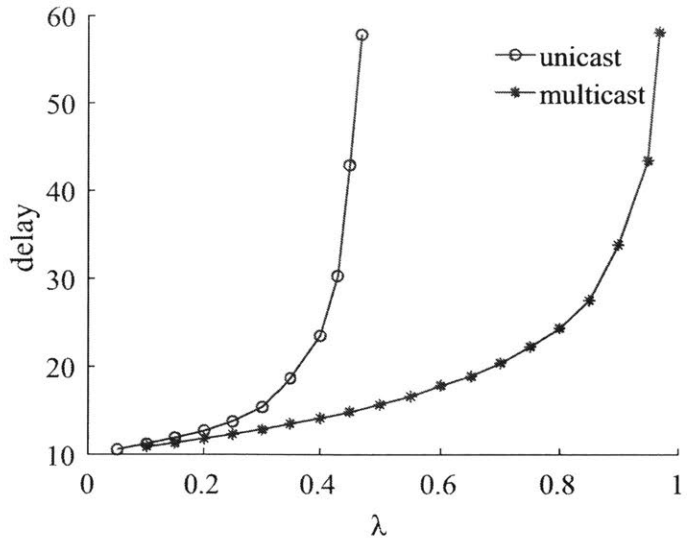


Figure A-8: Average packet delay of multicast traffic and when multicast is treated as multiple unicast traffic.

multiple unicast flows, for a total of 24 commodities, the maximum supportable rate is around $\lambda = 0.09$. This demonstrates the importance of optimal control for multicast traffic. The average packet delays under the backpressure-based algorithm, with multicast flows treated as multiple unicast flows, are over 1000 for $\lambda \in [0.01, 0.09]$, substantially higher than under UCNC, and hence ommitted in the figure.

Finally, we also evaluated the performance of an algorithm that uses the routing policy π^* and the First-In-First-Out (FIFO) scheduling policy for the physical queues. Numerical results demonstrate that the average packet delays are close to the delays under the ENTO scheduling policy, and are omitted for brevity. Thus, for practical purpose of dynamic control in distributed computing networks, FIFO scheduling policy could also be used.

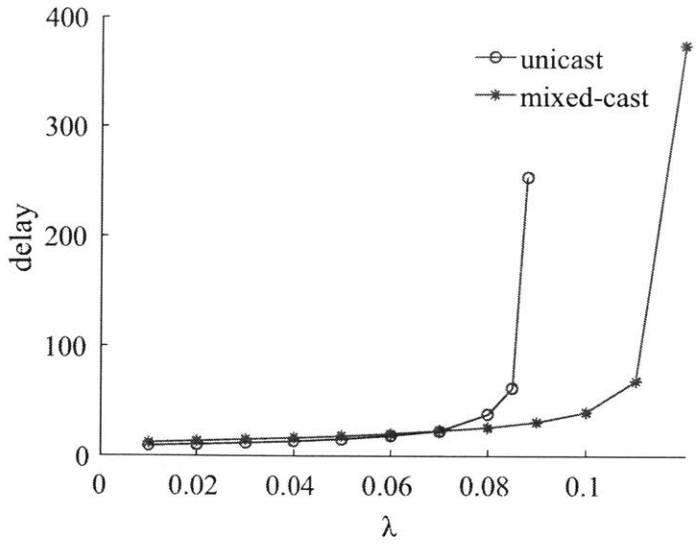


Figure A-9: Average packet delay of mixed-cast traffic and when multicast is treated as multiple unicast traffic.

A.6 Extensions

A.6.1 Undirected network

In an undirected network where the sum of transmission rates in both directions over a link is limited by the link capacity, the virtual queue updates should be modified while all the other steps in the algorithm remain the same.

Each undirected link (u, v) is associated with a virtual queue. A packet contributes an arrival to the queue if it plans to travel either from u to v, or from v to u. With this modified queue evolution, under the routing policy π^*, which routes a unicast packet over a shortest path and routes a multicast packet over a minimum Steiner tree, all the virtual queues are strongly stable for any arrival rate in the interior of the capacity region. Thus, the sum of the average packet arrival rates to a link through both directions is no more than the transmission capacity of the link. ENTO scheduling policy still guarantees the stability of physical queues when the link is undirected. Thus, the same routing and scheduling policy, with modified queue evolutions, is throughput-optimal.

A.6.2 Network throughput under approximate routing

The computation overhead will be reduced by allowing approximation in route selection. Consider a routing policy π^{α} that finds a path for a unicast packet and a Steiner tree for a multicast packet whose cost is at most $\alpha > 1$ times the minimum cost. The routing policy π^{α} and the ENTO scheduling policy are able to support arrival rate vector $\boldsymbol{\lambda}/\alpha$, where $\boldsymbol{\lambda}$ is in the interior of the stability region.

The proof follows a similar approach, by comparing the Lyapunov drift under π^{α} with the (scaled) drift under a randomized policy that supports $\boldsymbol{\lambda}/\alpha$.

A.6.3 Broadcast and anycast traffic

The broadcast traffic is a special case of the multicast traffic, where the destination nodes of a commodity include all the nodes in $\mathcal{V}$. At each time t, for a commodity-(c,ϕ) packet, the routing policy π^* computes a minimum Steiner tree that is rooted at $s_c^{(\phi,0)}$ and connected to $\mathcal{V}^{(\phi,M_\phi)}$.

For anycast traffic, where a commodity-(c,ϕ) packet is originated at s_c and destined for any node in $\mathcal{D}_c$, a dummy node $d_c^{(\phi,M_\phi)}$ is added in $G_{\text{IntDep}}^{(\phi,M_\phi)}$. Links of zero cost are added from $\mathcal{D}_c^{(\phi,M_\phi)}$ to $d_c^{(\phi,M_\phi)}$. The routing policy π^* computes a shortest path from $s_c^{(\phi,0)}$ to $d_c^{(\phi,M_\phi)}$.

A.6.4 Location-dependent computation requirements

UCNC can be extended to handle the problem where a service function (ϕ,i) may have different computation resource requirements at different computation nodes. For route selection, the cost of an edge $(u^{(\phi,i-1)},u^{(\phi,i)})$ at time t is modified to $x_u^{(\phi,i)}\tilde{Q}_u(t)$, where $\tilde{Q}_u(t)$ is the virtual queue length of u and $x_u^{(\phi,i)} = r_u^{(\phi,i)}\prod_{j=1}^{i-1}\xi^{(\phi,j)}$. The $r_u^{(\phi,i)}$ denotes the computation resource requirement to process each unit of input flow by function (ϕ,i) at node u.

A.7 Summary

We characterized the capacity region and developed the first throughput-optimal control policy (UCNC) for unicast and multicast traffic in a distributed computing network. UCNC handles both communication and computation constraints, flow scaling through service function chains, and packet duplications. Simulation results suggest that UCNC has superior performance compared with existing algorithms.

A.8 Proof of theorems

A.8.1 Restricted routes do not reduce the capacity region

Proof of Lemma A.1: We prove that, any packet that can be transmitted from the source to the destination(s) by time t under a policy π that uses arbitrary routes, can also be transmitted from the source to the destination(s) by time t under a policy π' that only uses efficient routes. Then, by Eq. (A.2), any rate $\boldsymbol{\lambda}$ that is supported by π can also be supported by π'. By Eq. (A.3), any rate in the capacity region can be supported by a policy that only uses efficient routes.

Consider a policy π that transmits the same packet to a node in $G_{\text{IntDep}}^{(\phi)}$ more than once. For unicast traffic, where there is no packet duplication, the packet travels through one or more cycles. Moreover, each cycle must be in one layer of $G_{\text{IntDep}}^{(\phi)}$ and the packet can not be processed while traveling through the cycle, since there is no edge from $G_{\text{IntDep}}^{(\phi,j)}$ to $G_{\text{IntDep}}^{(\phi,i)}$ for $i < j$. Construct a policy π' that removes all the cycles and transmission schedules on the cycle links. Any packet that arrives at a node (*e.g.*the destination) by time t under π can also arrive at the same node by time t under π'.

For multicast traffic, if a packet visits the same node $u^{(\phi,i)} \in G_{\text{IntDep}}^{(\phi)}$ more than once under policy π, then there are two possibilities.

1. The packet travels through one or more cycles in $G_{\text{IntDep}}^{(\phi,i)}$.

2. A packet is duplicated at another node v and more than one copy has traveled

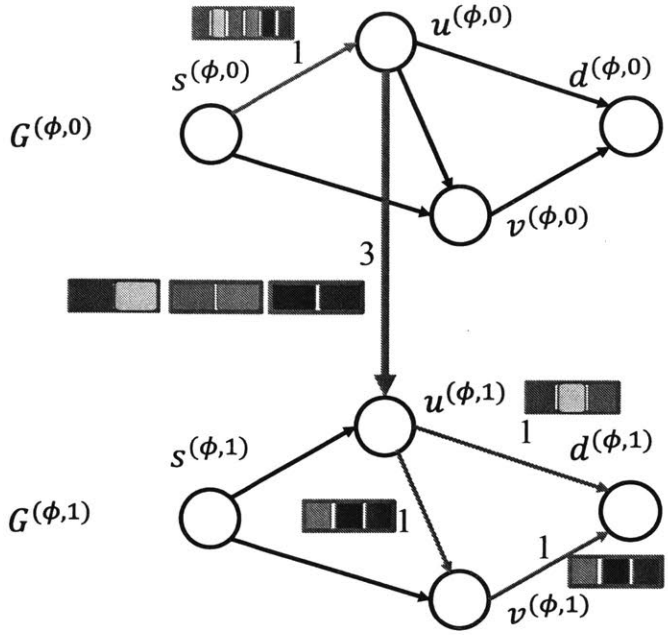


Figure A-10: Micro packets in an efficient route. The sizes of a micro packet on a link in $G^{(\phi,0)}_{\text{IntDep}}$, link $(u^{(\phi,0)}, u^{(\phi,1)})$, and a link in $G^{(\phi,1)}_{\text{IntDep}}$ are 1/6, 1/2, and 1/3, respectively. Scaling factors: $x^{(\phi,1)} = 3$; $w^{(\phi,1)} = 2$.

through some links and reached $u^{(\phi,i)}$.

To construct a policy π' that only uses efficient routes, we handle the first case in the same manner as the unicast case. *I.e.,* remove all the cycles and the transmission schedules of the packet on the cycles. For the second case, policy π' only keeps the routing and scheduling of the packet that first arrives at $u^{(\phi,i)}$, and removes all the duplications that arrive later. If the packet needs to be transmitted through more than one outgoing link from $u^{(\phi,i)}$ under π, then duplications occur at $u^{(\phi,i)}$ and the duplicated copies follow the same routes and schedules as π.

It is easy to check that the time that a packet visits a node under π' is the first time that the packet visits the node under π. By repeating the process until no packet visits the same node more than once, the policy π' only uses efficient routes.

Remark: If all scaling factors w, x are one, then an efficient route for a unicast packet is a path from the source to the destination. An efficient route for a multicast packet is a Steiner tree from the source to the destinations.

Proof of Theorem A.1: If $w^{(\phi,i)} = x^{(\phi,i)} = 1$, $\forall \phi \in \Phi, i \in \{1, \ldots, M_\phi\}$, the theorem

follows immediately from Lemma A.1. Next we study arbitrary (rational) scaling factors. We divide a packet into *micro packets*, and represent the routes of a packet by the composition of paths (or Steiner trees) of micro packets.[3]

Unicast traffic: Consider a policy π' that chooses an efficient route E_p for a unicast packet of commodity (c, ϕ). We assume that a rational number of packets are routed in each link. A micro packet is designed such that it changes size as it travels through the layered graph. Let $w^{(\phi,i)}/z$ be the size of the micro packet on a link in $G^{(\phi,i)}_{\text{IntDep}}$. Let $x^{(\phi,i)}/z$ be the size of the micro packet on a link that connects $G^{(\phi,i-1)}_{\text{IntDep}}$ and $G^{(\phi,i)}_{\text{IntDep}}$. The choice of z satisfies the following two constraints.

1. All the links in E_p carry an integer number of micro packets.

2. Every packet is divided to an integer number of micro packets.[4]

Fig. A-10 illustrates the decomposition into micro packets.

Due to the generalized flow conservation law for unicast traffic and the choice of micro packet sizes, the total number of incoming micro packets to a node equals the total number of outgoing micro packets from a node. In other words, every outgoing micro packet can be associated with an incoming micro packet, and they can be viewed to have the same *identity*. The links that carry micro packets with the same identity form a path. Since the routing is efficient, the path is acyclic. The route E_p can be viewed as a *convex combination* of service chain paths. More precisely, let p_k be the number of micro packets with different identities that travel through a path $T^{(c,\phi)}_k$. Let $\mathcal{T}^{(c,\phi)}$ be the set of paths from $s^{(\phi,0)}_c$ to $d^{(\phi,M_\phi)}_c$. The number of packets that travel through a link $(u^{(\phi,i)}, v^{(\phi,i)})$ is

$$\sum_{T^{(c,\phi)}_k \in \mathcal{T}^{(c,\phi)}} w^{(\phi,i)} \frac{p_k}{z} 1\{(u^{(\phi,i)}, v^{(\phi,i)}) \in T^{(c,\phi)}_k\}.$$

[3]The decomposition into micro packets is mostly useful for the analysis of multicast flow and can be bypassed in the analysis of unicast flow. However, we choose to use the decomposition for both unicast and multicast flows, for a unified treatment of the two cases.

[4]The second constraint is necessary only for multicast flows.

The number of packets that travel through a link $(u^{(\phi,i-1)}, u^{(\phi,i)})$ is

$$\sum_{T_k^{(c,\phi)} \in \mathcal{T}^{(c,\phi)}} x^{(\phi,i)} \frac{p_k}{z} 1\{(u^{(\phi,i-1)}, u^{(\phi,i)}) \in T_k^{(c,\phi)}\}.$$

Remark: Two micro packets with different identities are distinct (*i.e.*they carry different information). Micro packets that have the same identity can be viewed to carry the same (raw) information. More specifically, we assume that a function (ϕ, i) takes a micro packet as an input and then outputs a micro packet with the same identity. Equivalently, in $G_{\text{IntDep}}^{(\phi)}$, when a micro packet travels through a link in $G_{\text{IntDep}}^{(\phi,i-1)}$, a link that connects $G_{\text{IntDep}}^{(\phi,i-1)}$ and $G_{\text{IntDep}}^{(\phi,i)}$, and a link in $G_{\text{IntDep}}^{(\phi,i)}$, it has the same identity with possibly different sizes on the three links. In Fig. A-10, each color represents an identity.

Multicast traffic: The sizes of a micro packet are determined in the same manner as in the unicast case. Consider a node where there is no packet duplication. The total number of incoming micro packets equals the total number of outgoing micro packets, and every outgoing micro packet can be associated with an incoming micro packet. Both have the same identity. Consider a node where some packets P are duplicated. All the micro packets that are contained in P are duplicated as well. A duplication of a micro packet inherit the same identity as the original micro packet. Every micro packet follows the same route as the packet that contains the micro packet. The key observation is that *a micro packet is never split* under this construction, because all the links carry an integer number of micro packets and every packet contains an integer number of micro packets. Due to the efficient routing assumption, the micro packets that have the same identity never visit the same node in $G_{\text{IntDep}}^{(\phi)}$ more than once. Thus, the route of the micro packets that have the same identity form a service chain Steiner tree. The multicast flow can be viewed as a convex combination of service chain Steiner trees.

A.8.2 Stability of the virtual queues

Proof of Theorem A.2: We consider a quadratic Lyapunov function $L(\tilde{Q}(t)) = \sum_{(u,v)\in\mathcal{E}} \tilde{Q}_{uv}^2(t) + \sum_{u\in\mathcal{V}} \tilde{Q}_u^2(t)$. The following inequality holds for all $\tilde{Q}$.

$$\begin{aligned}\tilde{Q}^2(t+1) &\leq (\tilde{Q}(t) + A^{(c,\phi),\pi}(t) - \mu)^2 \\ &\leq \tilde{Q}^2(t) + (A^{(c,\phi),\pi}(t))^2 + \mu^2 \\ &+ 2\tilde{Q}(t)A^{(c,\phi),\pi}(t) - 2\tilde{Q}(t)\mu.\end{aligned}$$

Let $A_{uv}^{\pi}(t) = \sum_{(c,\phi)\in(\mathcal{C},\Phi)} A_{uv}^{(c,\phi),\pi}(t)$; $A_u^{\pi}(t) = \sum_{(c,\phi)\in(\mathcal{C},\Phi)} A_u^{(c,\phi),\pi}(t)$. The Lyapupov drift $\Delta^{\pi}(t)$ is upper bounded by

$$\begin{aligned}\Delta^{\pi}(t) &\stackrel{\text{def}}{=} \mathbb{E}(L(\tilde{Q}(t+1)) - L(\tilde{Q}(t))|\tilde{Q}(t)) \\ &\leq B + 2\sum_{(u,v)\in\mathcal{E}} \tilde{Q}_{uv}(t)\left(\mathbb{E}(A_{uv}^{\pi}(t)|\tilde{Q}(t)) - \mu_{uv}\right) \\ &\quad +2\sum_{u\in\mathcal{V}} \tilde{Q}_u(t)\left(\mathbb{E}(A_u^{\pi}(t)|\tilde{Q}(t)) - \mu_u\right),\end{aligned} \tag{A.10}$$

where

$$\begin{aligned}B &= \sum_{(u,v)\in\mathcal{E}}\left(\mathbb{E}(A_{uv}^{\pi}(t))^2 + \mu_{uv}^2\right) \\ &\quad +\sum_{u\in\mathcal{V}}\left(\mathbb{E}(A_u^{\pi}(t))^2 + \mu_u^2\right) \\ &\leq \sum_{(u,v)\in\mathcal{E}}\sum_{(c,\phi)\in(\mathcal{C},\Phi)}\sum_{i\in\{0,\ldots,M_\phi\}} (w^{(\phi,i)})^2\mathbb{E}(A^{(c,\phi)}(t))^2 \\ &\quad +\sum_{u\in\mathcal{V}}\sum_{(c,\phi)\in(\mathcal{C},\Phi)}\sum_{i\in\{1,\ldots,M_\phi\}} (x^{(\phi,i)})^2\mathbb{E}(A^{(c,\phi)}(t))^2 \\ &\quad +\sum_{(u,v)\in\mathcal{E}}\mu_{uv}^2 + \sum_{u\in\mathcal{V}}\mu_u^2.\end{aligned}$$

For finite second moment of exogenous arrivals $\mathbb{E}(A^{(c,\phi)}(t))^2$ and finite scaling factors, B is finite.

We next prove that the drift $\Delta^{\pi}(t)$ is negative for sufficiently large $\tilde{Q}(t)$, by comparing $\Delta^{\pi}(t)$ with the drift of a randomized policy. For any $\bar{\boldsymbol{\lambda}}$ in the interior of the capacity region, there exists $\epsilon > 0$ such that $(1+\epsilon)\bar{\boldsymbol{\lambda}} \in \Lambda(G_{\text{IntDep}}, \mathcal{C}, \Phi)$. The rate $\boldsymbol{\lambda} = (1+\epsilon)\bar{\boldsymbol{\lambda}}$ satisfies the constraints (A.4), (A.5) and (A.6). Let $\lambda_k^{(c,\phi)} = (1+\epsilon)\bar{\lambda}_k^{(c,\phi)}$, $\forall k, c, \phi$.

$$
\begin{aligned}
(1+\epsilon)\bar{\lambda}^{(c,\phi)} &= \sum_{T_k^{(c,\phi)} \in \mathcal{T}^{(c,\phi)}} (1+\epsilon)\bar{\lambda}_k^{(c,\phi)}, \quad \forall (c,\phi) \in (\mathcal{C}, \Phi), \\
\sum_{(k,i,c,\phi)\in\mathcal{S}_{uv}} w^{(\phi,i)}(1+\epsilon)\bar{\lambda}_k^{(c,\phi)} &\le \mu_{uv}, \qquad \forall (u,v) \in \mathcal{E}, \\
\sum_{(k,i,c,\phi)\in\mathcal{S}_{u}} x^{(\phi,i)}(1+\epsilon)\bar{\lambda}_k^{(c,\phi)} &\le \mu_{u}, \qquad \forall u \in \mathcal{V}.
\end{aligned}
$$

The randomized policy routes each incoming commodity-(c,ϕ) packet along $T_k^{(c,\phi)} \in \mathcal{T}_k^{(c,\phi)}$ with probability $\bar{\lambda}_k^{(c,\phi)}/\bar{\lambda}^{(c,\phi)}$, $\forall k, c, \phi$. The expected arrival rates to $\tilde{Q}_{uv}$ and $\tilde{Q}_u$ at every time t are

$$
\mathbb{E}A_{uv}^{\text{rand}}(t) = \sum_{(k,i,c,\phi)\in\mathcal{S}_{uv}} w^{(\phi,i)}\bar{\lambda}_k^{(c,\phi)} \le \mu_{uv}/(1+\epsilon);
$$

$$
\mathbb{E}A_{u}^{\text{rand}}(t) = \sum_{(k,i,c,\phi)\in\mathcal{S}_{u}} x^{(\phi,i)}\bar{\lambda}_k^{(c,\phi)} \le \mu_{u}/(1+\epsilon).
$$

Recall Eq. (A.9). Upon the arrival of $A^{(c,\phi)}(t)$ commodity-(c,ϕ) packets, policy π^* chooses a route $T^{(c,\phi),\pi^*}$ that achieves the minimum

$$
\min_{\pi} \sum_{(u,v)\in\mathcal{E}} \tilde{Q}_{uv}(t) A_{uv}^{(c,\phi),\pi}(t) + \sum_{u\in\mathcal{V}} \tilde{Q}_u(t) A_u^{(c,\phi),\pi}(t).
$$

The randomized policy randomly chooses $T_k^{(c,\phi)} \in \mathcal{T}_k^{(c,\phi)}$ which has an equal or larger weight. Conditional on queue lengths $\tilde{Q}(t)$, taking expectation over the random variable $A^{(c,\phi)}(t)$ and the random actions in the randomized policy,

$$
\sum_{(u,v)\in\mathcal{E}} \tilde{Q}_{uv}(t)\mathbb{E}(A_{uv}^{(c,\phi),\pi^*}(t)|\tilde{Q}(t))
$$

$$
\begin{aligned}
& + \sum_{u \in \mathcal{V}} \tilde{Q}_u(t) \mathbb{E}(A_u^{(c,\phi),\pi^*}(t) | \tilde{Q}(t)) \\
\leq & \sum_{(u,v) \in \mathcal{E}} \tilde{Q}_{uv}(t) \mathbb{E}(A_{uv}^{(c,\phi),\text{rand}}(t) | \tilde{Q}(t)) \\
& + \sum_{u \in \mathcal{V}} \tilde{Q}_u(t) \mathbb{E}(A_u^{(c,\phi),\text{rand}}(t) | \tilde{Q}(t)).
\end{aligned}
$$

Summing over all commodity packets, for each link (u, v),

$$
\begin{aligned}
\mathbb{E}(A_{uv}^{\pi^*}(t) | \tilde{Q}(t)) &= \sum_{(c,\phi) \in (\mathcal{C},\Phi)} \mathbb{E}(A_{uv}^{(c,\phi),\pi^*}(t) | \tilde{Q}(t)), \\
\mathbb{E}(A_{uv}^{\text{rand}}(t) | \tilde{Q}(t)) &= \sum_{(c,\phi) \in (\mathcal{C},\Phi)} \mathbb{E}(A_{uv}^{(c,\phi),\text{rand}}(t) | \tilde{Q}(t)).
\end{aligned}
$$

Similar equalities hold for $\mathbb{E}(A_u^{\pi^*}(t) | \tilde{Q}(t))$ and $\mathbb{E}(A_u^{\text{rand}}(t) | \tilde{Q}(t))$. Therefore, we obtain

$$
\begin{aligned}
& \sum_{(u,v) \in \mathcal{E}} \tilde{Q}_{uv}(t) \mathbb{E}(A_{uv}^{\pi^*}(t) | \tilde{Q}(t)) + \sum_{u \in \mathcal{V}} \tilde{Q}_u(t) \mathbb{E}(A_u^{\pi^*}(t) | \tilde{Q}(t)) \\
\leq & \sum_{(u,v) \in \mathcal{E}} \tilde{Q}_{uv}(t) \mathbb{E}(A_{uv}^{\text{rand}}(t) | \tilde{Q}(t)) + \sum_{u \in \mathcal{V}} \tilde{Q}_u(t) \mathbb{E}(A_u^{\text{rand}}(t) | \tilde{Q}(t)).
\end{aligned}
$$

The action of the randomized policy does not depend on the queue length $\tilde{Q}(t)$. Therefore, $\mathbb{E}(A_{uv}^{\text{rand}}(t) | \tilde{Q}(t)) = \mathbb{E}A_{uv}^{\text{rand}}(t)$ and $\mathbb{E}(A_u^{\text{rand}}(t) | \tilde{Q}(t)) = \mathbb{E}A_u^{\text{rand}}(t)$. Let $\epsilon' = \frac{\epsilon}{1+\epsilon} \min(\mu_{uv}, \mu_u)$. The drift of policy π^* can be upper bounded by

$$
\begin{aligned}
\Delta^{\pi^*}(t) \leq B & + 2 \sum_{(u,v) \in \mathcal{E}} \tilde{Q}_{uv}(t) \left(\mathbb{E}(A_{uv}^{\pi^*}(t) | \tilde{Q}(t)) - \mu_{uv} \right) \\
& + 2 \sum_{u \in \mathcal{V}} \tilde{Q}_u(t) \left(\mathbb{E}(A_u^{\pi^*}(t) | \tilde{Q}(t)) - \mu_u \right) \\
\leq B & + 2 \sum_{(u,v) \in \mathcal{E}} \tilde{Q}_{uv}(t) \left(\mathbb{E}A_{uv}^{\text{rand}}(t) - \mu_{uv} \right) \\
& + 2 \sum_{u \in \mathcal{V}} \tilde{Q}_u(t) \left(\mathbb{E}A_u^{\text{rand}}(t) - \mu_u \right) \\
\leq B & - 2\epsilon' \left(\sum_{(u,v) \in \mathcal{E}} \tilde{Q}_{uv}(t) + \sum_{u \in \mathcal{V}} \tilde{Q}_u(t) \right).
\end{aligned}
$$

Taking expectation over the virtual queue lengths $\tilde{Q}(t)$,

$$\begin{aligned} \mathbb{E}L(\tilde{Q}(t+1)) &- \mathbb{E}L(\tilde{Q}(t)) \leq B \\ &- 2\epsilon' \left(\sum_{(u,v)\in\mathcal{E}} \mathbb{E}\tilde{Q}_{uv}(t) + \sum_{u\in\mathcal{V}} \mathbb{E}\tilde{Q}_u(t) \right). \end{aligned} \tag{A.11}$$

Summing Eq. (A.11) from $t = 0, \ldots, T-1$, and noting that $L(\tilde{Q}(T)) \geq 0$, $L(\tilde{Q}(0)) = 0$, we obtain

$$\frac{1}{T} \sum_{t=0}^{T-1} \left(\sum_{(u,v)\in\mathcal{E}} \mathbb{E}\tilde{Q}_{uv}(t) + \sum_{u\in\mathcal{V}} \mathbb{E}\tilde{Q}_u(t) \right) \leq \frac{B}{2\epsilon'}.$$

By taking limsup on both sides, we have proved that all the queues are strongly stable.

A.8.3 Stability of the physical queues

Before the proof, we first discuss the the intuitions on what makes a queue unstable and why the extended nearest-to-origin (ENTO) scheduling policy stabilizes the queue. Consider external packets arriving at a network, each of which has a specified path to travel. If the rate of external arrivals that will use link e is no more than the service rate of e, the only cause of instability of the queue at e is the variation of packet delays before reaching e. The packets may take different paths and experience different queueing delays. Within some time period, the actual arrival rate to e can be higher than the service rate of e. The rate increase can be viewed as the contribution from the old packets in other queues (in contrast with the fresh packets that just arrived). The ENTO policy gives a higher priority to a packet that has traveled a smaller number of hops. Thus, few packets that have traveled a small number of hops are queued. These packets do not contribute much to the actual arrival rate to a subsequent queue. Thus, few packets that have traveled a slightly more number of hops are queued, because the only old packets that have higher priorities are those packets that have traveled a small number of hops. By induction, not many packets

are in each queue, regardless of the number of hops that they have traveled, and thus the queues are stable.

In the following, we first show that, within any time interval, the packets that arrive at the network do not contribute to a physical link e much more traffic than what can be transmitted through e. Then we prove that ENTO stabilizes the queue. The first proof is identical to [103]. The second proof is similar, but takes care of flow scaling and cyclic routes.

Average arrival rate is no more than the service rate for every physical queue

For simplicity, we augment each computation node in the original graph G_{IntDep} by a self-loop that represents the computation queue. We denote the set of all links and self-loops by $\bar{\mathcal{E}}$.

Since the virtual queues are strongly stable under policy π^* (Theorem A.2), all the virtual queues are rate stable (Lemma 1 in [103]).

$$\lim_{t\to\infty} \frac{\tilde{Q}_e(t)}{t} = 0, \quad \forall (u,v) \in \bar{\mathcal{E}}, \quad \text{w.p. 1.}$$

Almost surely for any sample path $\omega \in \Omega$ (*i.e.*a realization of random arrivals),

$$A_e(\omega; t_0, t) \leq S_e(\omega; t_0, t) + F_e(\omega; t), \quad e \in \bar{\mathcal{E}}, \tag{A.12}$$

where $A_e(\omega; t_0, t) = \sum_{\tau=t_0}^{t-1} A_e^{\pi^*}(\omega; \tau)$ is the total number of packets that arrive at virtual queue $\tilde{Q}_e$ during time $[t_0, t)$ under policy π^* and sample path ω; $S_e(\omega; t_0, t) = \sum_{\tau=t_0}^{t-1} \mu_e = (t - t_0)\mu_e$ is the total number of packets that can be served by e; $F_e(\omega; t) = o(t)$ (*i.e.*$\lim_{t\to\infty} F_e(\omega; t)/t = 0$). Eq. (A.12) implies that the average arrival rate to the virtual queue $\tilde{Q}_e$ is no more than the service rate of e.

Next, we relate the arrival rate at a virtual queue to the arrival rate at a physical queue. Since the routing policy for the physical system is identical to the routing policy π^* for the virtual system, the exogenous packets that arrive at the network at

time t contribute a total of $A_e(t_0, t)$ packets to e during the course of their service in the physical system. (Recall that a packet with a scaled size enters a virtual queue of a link immediately if the link is part of its route.)

ENTO stabilizes the physical queues

We aim to prove that ENTO stabilizes the physical queues for any sample path ω that satisfies Eq. (A.12). In particular, we aim to prove

$$\lim_{t\to\infty} \frac{Q_e(\omega; t)}{t} = 0, \quad \forall e \in \bar{\mathcal{E}}. \tag{A.13}$$

Then, ENTO stabilizes the physical queues almost surely because Eq. (A.12) holds for almost all sample paths.

$$\lim_{t\to\infty} \frac{Q_e(t)}{t} = 0, \quad \text{w.p. } 1, \ \forall e \in \bar{\mathcal{E}}.$$

For simplicity of presentation, we drop the ω in the notations and focus on one sample path. It has been shown in [103] that there exists a *non-decreasing non-negative* function $M(t) = o(t)$ such that

$$A_e(t_0, t) \leq S_e(t_0, t) + M(t), \quad \forall e \in \bar{\mathcal{E}}, t_0 \leq t. \tag{A.14}$$

We introduce a few new notations. A *hop-k* packet is a packet that has traveled k hops from the origin. The processing at a computation node is also considered as one hop. A duplication of a packet inherits the hop of the original packet. The packets entering the network during $[t_0, t)$ contribute to e a total of $A_e(t_0, t)$ packets. Among these packets, $A_e^k(t_0, t)$ packets use e as their $(k+1)$-th hop, and they are hop-k packets whiling waiting to cross e. Let $M^{\max} = \max_\phi M_\phi + 1$ denote the maximum number of functions in any service chain plus one. The maximum number of hops that a packet travels under the routing policy π^* is $nM^{\max}$, where n is the number of

nodes in G_{IntDep}. By definition,

$$A_e(t_0, t) = \sum_{k=0}^{nM^{\max}-1} A_e^k(t_0, t).$$

Let

$$\gamma = \max_{\phi \in \Phi, 0 \le i \le j \le M_\phi} \frac{\max(w^{(\phi,j)}, x^{(\phi,j)})}{\min(w^{(\phi,i)}, x^{(\phi,i)})}$$

denote the maximum aggregated scaling factor. *I.e.*each packet that departs one link contributes to at most γ packets to any subsequent link in $G_{\text{IntDep}}^{(\phi)}$. Note that the value $A_e(t_0, t)$ has taken the scalings into consideration, *i.e.*Eqs. (A.7) and (A.8). Let $Q_e(t)$ denote the physical queue length at e at time t. Let $Q_e^k(t)$ denote the number of hop-k packets in the queue at e at time t. Let $Q^k(t) = \sum_{e \in \bar{\mathcal{E}}} Q_e^k$ denote the total number of hop-k packets in the network at time t. We prove by induction that $Q^k(t) = o(t)$ for all $k \in \{0, \ldots, nM_{\max} - 1\}$.

Base step $k = 0$: Let $t_0 < t$ be the largest time at which no hop-0 packet were waiting to cross a specified link e. If no such time exists, $t_0 = 0$. During $[t_0, t)$, at most $A_e^0(t_0, t) \le A_e(t_0, t) \le S_e(t_0, t) + M(t)$ hop-0 packets arrived at e, by Eq. (A.14). Moreover, e is constantly transmitting hop-0 packets, for a total of $S_e(t_0, t)$ packets, because hop-0 packets have the highest priority and there are always hop-0 packets waiting to cross e by the choice of t_0. Therefore,

$$Q_e^0(t) \le S_e(t_0, t) + M(t) - S_e(t_0, t) = M(t).$$

There are at most $\bar{m} = |\mathcal{E}| + |\mathcal{V}|$ physical queues. Therefore, $Q^0(t) \le \bar{m} M(t)$. Let $B^0(t) = \bar{m} M(t) = o(t)$. Note that $B^0(t)$ is non-decreasing in t.

Induction step: Suppose that $Q^j(t) \le B^j(t)$ for all $0 \le j < k$, where $B^j(t) = o(t)$ is non-decreasing. We aim to prove that $Q^k(t) \le B^k(t)$, for a non-decreasing $B^k(t) = o(t)$. Let t_0 be the largest time at which no hop-k packets were waiting to cross a specified link e. Let $t_0 = 0$ if no such time exists.

The *new* packets that arrive at the network during $[t_0, t)$ contributes at most $A_e^k(t_0, t)$ hop-k packets to e by time t. The *old* packets that were already in the

network by time t_0 contributes to e at most $\gamma \sum_{0 \le j < k} B^j(t_0)$ hop-k packets, because each of the $\sum_{0 \le j < k} B^j(t_0)$ old packets of hop fewer than k contributes at most γ hop-k packets to e. Note that the old packets of hop more than k never become hop-k packets again.

Next we bound the number of packets of hop fewer than k that are transmitted through e during $[t_0, t)$. The new packets that arrive at the network during $[t_0, t)$ contribute to e at most $\sum_{0 \le j < k} A_e^j(t_0, t)$ packets of hop fewer than k. Each old packet contributes at most γ hop-j packets ($0 \le j < k$). Thus, the total number of packets of hop fewer than k contributed by one old packet is at most γk. For a total of $\sum_{0 \le j < k} B^j(t_0)$ old packets, at most $\gamma k \sum_{0 \le j < k} B^j(t_0)$ packets of hop fewer than k travel through e during $[t_0, t)$.

The link is consistently processing packets of hop no more than k during $[t_0, t)$, by the choice of t_0. The packets that have hop fewer than k have a higher priority than the hop-k packets. Thus, the number of hop-k packets that are processed by e is at least $\max(0, S_e(t_0, t) - \sum_{0 \le j < k} A_e^j(t_0, t) - \gamma k \sum_{0 \le j < k} B^j(t_0))$.

The number of hop-k packets at queue e at time t is at most

$$\begin{aligned} Q_e^k(t) &\le A_e^k(t_0, t) + \gamma \sum_{0 \le j < k} B^j(t_0) \\ &- (S_e(t_0, t) - \sum_{0 \le j < k} A_e^j(t_0, t) - \gamma k \sum_{0 \le j < k} B^j(t_0)) \\ &\le \gamma(k+1) \sum_{0 \le j < k} B^j(t_0) + M(t). \end{aligned}$$

Let $B_e^k(t) = \gamma(k+1) \sum_{0 \le j < k} B^j(t) + M(t)$. Since $M(t)$ and $B^j(t)$ are non-decreasing in t for $0 \le j < k$, $B_e^k(t)$ is a non-decreasing function and $B_e^k(t) \ge \gamma(k+1) \sum_{0 \le j < k} B^j(t_0) + M(t)$. We have $Q_e^k(t) \le B_e^k(t)$. Since $B^j(t) = o(t)$ for $0 \le j < k$ and $M(t) = o(t)$, we have $B_e^k(t) = o(t)$. Let $B^k(t) = \sum_{e \in \bar{\mathcal{E}}} B_e^k(t) = \bar{m} B_e^k(t)$. It is easy to check that $B^k(t) = o(t)$ is a non-decreasing function.

We have proved that $Q^k(t) = o(t)$ for all k. Then, the sum of all queue lengths $\sum_{e \in \bar{\mathcal{E}}} Q_e(t) = \sum_k Q^k(t) = o(t)$. Therefore, all the physical queues are stable, and Eq. (A.13) holds.

Bibliography

[1] Vittorio Rosato, L Issacharoff, F Tiriticco, Sandro Meloni, S Porcellinis, and Roberto Setola. Modelling interdependent infrastructures using interacting dynamical models. *Int. J. Critical Infrastructures*, 4(1-2):63–79, 2008.

[2] Marzieh Parandehgheibi and Eytan Modiano. Robustness of interdependent networks: The case of communication networks and the power grid. In *Proc. IEEE GLOBECOM*, pages 2164–2169, Atlanta, 2013.

[3] Chang-Gui Gu, Sheng-Rong Zou, Xiu-Lian Xu, Yan-Qing Qu, Yu-Mei Jiang, Hong-Kun Liu, Tao Zhou, et al. Onset of cooperation between layered networks. *Physical Review E*, 84(2):026101, 2011.

[4] Osman Yagan, Dajun Qian, Junshan Zhang, and Douglas Cochran. Optimal allocation of interconnecting links in cyber-physical systems: Interdependence, cascading failures, and robustness. *IEEE Transactions on Parallel and Distributed Systems*, 23(9):1708–1720, 2012.

[5] Nasir Ghani, Sudhir Dixit, and Ti-Shiang Wang. On IP-over-WDM integration. *IEEE Communications Magazine*, 38(3):72–84, 2000.

[6] Kayi Lee, Eytan Modiano, and Hyang-Won Lee. Cross-layer survivability in WDM-based networks. *IEEE/ACM Transactions on Networking*, 19(4):1000–1013, 2011.

[7] Sergey V Buldyrev, Roni Parshani, Gerald Paul, H Eugene Stanley, and Shlomo Havlin. Catastrophic cascade of failures in interdependent networks. *Nature*, 464(7291):1025–1028, 2010.

[8] Marzieh Parandehgheibi, Konstantin Turitsyn, and Eytan Modiano. Modeling the impact of communication loss on the power grid under emergency control. In *Proc. IEEE SmartGridComm*, pages 356–361, Miami, 2015.

[9] Hyang-Won Lee, Kayi Lee, and Eytan Modiano. Maximizing reliability in WDM networks through lightpath routing. *IEEE/ACM Transactions on Networking*, 22(4):1052–1066, 2014.

[10] Gaogao Dong, Jianxi Gao, Ruijin Du, Lixin Tian, H Eugene Stanley, and Shlomo Havlin. Robustness of network of networks under targeted attack. *Physical Review E*, 87(5):052804, 2013.

[11] Jianxi Gao, Sergey V Buldyrev, Shlomo Havlin, and H Eugene Stanley. Robustness of a network of networks. *Physical Review Letters*, 107(19):195701, 2011.

[12] Amir Bashan, Yehiel Berezin, Sergey V Buldyrev, and Shlomo Havlin. The extreme vulnerability of interdependent spatially embedded networks. *Nature Physics*, 9(10):667–672, 2013.

[13] Yehiel Berezin, Amir Bashan, Michael M Danziger, Daqing Li, and Shlomo Havlin. Localized attacks on spatially embedded networks with dependencies. *Scientific reports*, 5, 2015.

[14] Louis M Shekhtman, Yehiel Berezin, Michael M Danziger, and Shlomo Havlin. Robustness of a network formed of spatially embedded networks. *Physical Review E*, 90(1):012809, 2014.

[15] Jia Shao, Sergey V. Buldyrev, Shlomo Havlin, and H. Eugene Stanley. Cascade of failures in coupled network systems with multiple support-dependence relations. *Phys. Rev. E*, 83:036116, Mar 2011.

[16] Roni Parshani, Sergey V Buldyrev, and Shlomo Havlin. Interdependent networks: Reducing the coupling strength leads to a change from a first to second order percolation transition. *Physical review letters*, 105(4):048701, 2010.

[17] Di Zhou, Jianxi Gao, H Eugene Stanley, and Shlomo Havlin. Percolation of partially interdependent scale-free networks. *Physical Review E*, 87(5):052812, 2013.

[18] Shunsuke Watanabe and Yoshiyuki Kabashima. Cavity-based robustness analysis of interdependent networks: Influences of intranetwork and internetwork degree-degree correlations. *Physical Review E*, 89(1):012808, 2014.

[19] Sergey V Buldyrev, Nathaniel W Shere, and Gabriel A Cwilich. Interdependent networks with identical degrees of mutually dependent nodes. *Physical Review E*, 83(1):016112, 2011.

[20] Yanqing Hu, Dong Zhou, Rui Zhang, Zhangang Han, Céline Rozenblat, and Shlomo Havlin. Percolation of interdependent networks with intersimilarity. *Physical Review E*, 88(5):052805, 2013.

[21] Xueming Liu, H Eugene Stanley, and Jianxi Gao. Breakdown of interdependent directed networks. *Proceedings of the National Academy of Sciences*, 113(5):1138–1143, 2016.

[22] Charles D Brummitt, Raissa M D'Souza, and Elizabeth A Leicht. Suppressing cascades of load in interdependent networks. *Proceedings of the National Academy of Sciences*, 109(12):680–689, 2012.

[23] MA Di Muro, SV Buldyrev, HE Stanley, and LA Braunstein. Cascading failures in interdependent networks with finite functional components. *Physical Review E*, 94(4):042304, 2016.

[24] MA Di Muro, LD Valdez, HH Aragão Rêgo, SV Buldyrev, HE Stanley, and LA Braunstein. Cascading failures in interdependent networks with multiple supply-demand links and functionality thresholds. *Scientific reports*, 7(1):15059, 2017.

[25] Marcell Stippinger and János Kertész. Enhancing resilience of interdependent networks by healing. *Physica A: Statistical Mechanics and its Applications*, 416:481–487, 2014.

[26] MA Di Muro, CE La Rocca, HE Stanley, S Havlin, and LA Braunstein. Recovery of interdependent networks. *Scientific reports*, 6:22834, 2016.

[27] Louis M Shekhtman, Michael M Danziger, and Shlomo Havlin. Recent advances on failure and recovery in networks of networks. *Chaos, Solitons & Fractals*, 90:28–36, 2016.

[28] Junjian Qi, Wenyun Ju, and Kai Sun. Estimating the propagation of interdependent cascading outages with multi-type branching processes. *IEEE Transactions on Power Systems*, 32(2):1212–1223, 2017.

[29] Junjian Qi, Jianhui Wang, and Kai Sun. Efficient estimation of component interactions for cascading failure analysis by em algorithm. *IEEE Transactions on Power Systems*, 33(3):3153–3161, 2018.

[30] Marzieh Parandehgheibi, Eytan Modiano, and David Hay. Mitigating cascading failures in interdependent power grids and communication networks. In *Smart Grid Communications (SmartGridComm), 2014 IEEE International Conference on*, pages 242–247. IEEE, 2014.

[31] Baichao Wu, Aiping Tang, and Jie Wu. Modeling cascading failures in interdependent infrastructures under terrorist attacks. *Reliability Engineering & System Safety*, 147:1–8, 2016.

[32] Yatin Wadhawan and Clifford Neuman. Evaluating resilience of gas pipeline systems under cyber-physical attacks: A function-based methodology. In *Proc. 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy*, pages 71–80, Vienna, Austria, 2016.

[33] Min Ouyang. Review on modeling and simulation of interdependent critical infrastructure systems. *Reliability engineering & System safety*, 121:43–60, 2014.

[34] D. Papadimitriou et al. Inference of shared risk link groups. Internet-Draft draft-many-inference-srlg-02.txt, IETF Secretariat, November 2001.

[35] Jian Qiang Hu. Diverse routing in optical mesh networks. *IEEE Transactions on Communications*, 51(3):489–494, 2003.

[36] David Coudert, P Datta, Stéphane Pérennes, Hervé Rivano, and M-E Voge. Shared risk resource group complexity and approximability issues. *Parallel Processing Letters*, 17(02):169–184, 2007.

[37] Shengli Yuan, Sumir Varma, and Jason P Jue. Minimum-color path problems for reliability in mesh networks. In *Proc. INFOCOM 2005*, pages 2658–2669, Miami, 2005.

[38] Sulamita Klein, Luerbio Faria, Ignasi Sau, Rubens Sucupira, and U Souza. On colored edge cuts in graphs. *Sociedade Brasileira de Computaçao, editor, Primeiro Encontro de Teoria da Computaçao (ETC)*, 2016.

[39] Peng Zhang, Jin-Yi Cai, Lin-Qing Tang, and Wen-Bo Zhao. Approximation and hardness results for label cut and related problems. *Journal of Combinatorial Optimization*, 21(2):192–208, 2011.

[40] David R Karger. A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. *SIAM review*, 43(3):499–522, 2001.

[41] Keren Censor-Hillel, Mohsen Ghaffari, and Fabian Kuhn. Distributed connectivity decomposition. In *Proc. ACM symposium on Principles of distributed computing*, pages 156–165, Paris, France, 2014.

[42] Keren Censor-Hillel, Mohsen Ghaffari, and Fabian Kuhn. A new perspective on vertex connectivity. In *Proc. Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 546–561, Portland, Oregon, 2014.

[43] Keren Censor-Hillel, Mohsen Ghaffari, George Giakkoupis, Bernhard Haeupler, and Fabian Kuhn. Tight bounds on vertex connectivity under vertex sampling. In *Proc. Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2006–2018, San Diego, California, 2015.

[44] Song Yang, Stojan Trajanovski, and Fernando A Kuipers. Shortest paths in networks with correlated link weights. In *Proc. IEEE Computer Communications Workshops (INFOCOM WKSHPS)*, pages 660–665, Hong Kong, 2015.

[45] Hyang-Won Lee, Eytan Modiano, and Kayi Lee. Diverse routing in networks with probabilistic failures. *IEEE/ACM Transactions on Networking*, 18(6):1895–1907, 2010.

[46] M Farhan Habib, Massimo Tornatore, Ferhat Dikbiyik, and Biswanath Mukherjee. Disaster survivability in optical communication networks. *Computer Communications*, 36(6):630–644, 2013.

[47] Weidong Cui, Ion Stoica, and Randy H Katz. Backup path allocation based on a correlated link failure probability model in overlay networks. In *Proc. IEEE Int. Conf. Network Protocols*, pages 236–245, Paris, 2002.

[48] Qiang Zheng, Guohong Cao, Thomas F La Porta, and Ananthram Swami. Cross-layer approach for minimizing routing disruption in ip networks. *IEEE Transactions on Parallel and Distributed Systems*, 25(7):1659–1669, 2014.

[49] Chiping Tang and Philip K McKinley. Improving multipath reliability in topology-aware overlay networks. In *Proc. IEEE International Conference on Distributed Computing Systems Workshops*, pages 82–88, Columbus, Ohio, 2005.

[50] T. Fei, S. Tao, L. Gao, and R. Guerin. How to select a good alternate path in large peer-to-peer systems? In *Proc. INFOCOM 2006*, pages 1–13, Barcelona, April 2006.

[51] Shimon Even and R Endre Tarjan. Network flow and testing graph connectivity. *SIAM journal on computing*, 4(4):507–518, 1975.

[52] Pallab Datta and Arun K Somani. Diverse routing for shared risk resource groups (SRRG) failures in WDM optical networks. In *Proc. International Conference on Broadband Networks (BroadNets)*, pages 120–129, San Jose, 2004.

[53] D. Bertsimas and J.N. Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific series in optimization and neural computation. Athena Scientific, 1997.

[54] Harold N Gabow. Using expander graphs to find vertex connectivity. *Journal of the ACM (JACM)*, 53(5):800–844, 2006.

[55] Miles Lubin and Iain Dunning. Computing in operations research using julia. *INFORMS Journal on Computing*, 27(2):238–248, 2015.

[56] XO Communications. Network map.

[57] Richard M Karp, Michael Luby, and Neal Madras. Monte-carlo approximation algorithms for enumeration problems. *Journal of algorithms*, 10(3):429–448, 1989.

[58] Wolfgang Gatterbauer and Dan Suciu. Oblivious bounds on the probability of boolean functions. *ACM Transactions on Database Systems (TODS)*, 39(1):5, 2014.

[59] Abraham P Punnen. A linear time algorithm for the maximum capacity path problem. *European J. Oper. Res.*, 53(3):402–404, 1991.

[60] Jack Edmonds and Richard M Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM (JACM)*, 19(2):248–264, 1972.

[61] Leslie G Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.

[62] Dan Roth. On the hardness of approximate reasoning. *Artificial Intelligence*, 82(1):273–302, 1996.

[63] Catherine Greenhill. The complexity of counting colourings and independent sets in sparse graphs and hypergraphs. *Computational Complexity*, 9(1):52–72, 2000.

[64] Mathew Penrose. *Random geometric graphs.* Oxford Univ. Press, 2003.

[65] Ronald Meester and Rahul Roy. *Continuum percolation.* Cambridge Univ. Press, 1996.

[66] Paul Balister, Amites Sarkar, and Béla Bollobás. Percolation, connectivity, coverage and colouring of random geometric graphs. In *Handbook of Large-Scale Random Networks*, pages 117–142. Springer, 2008.

[67] Wei Li, Amir Bashan, Sergey V Buldyrev, H Eugene Stanley, and Shlomo Havlin. Cascading failures in interdependent lattice networks: The critical role of the length of dependency links. *Physical review letters*, 108(22):228702, 2012.

[68] Peter Hall. On continuum percolation. *The Annals of Probability*, pages 1250–1266, 1985.

[69] Paul Balister, Béla Bollobás, and Mark Walters. Continuum percolation with steps in the square or the disc. *Random Structures and Algorithms*, 26(4):392–403, 2005.

[70] Geoffrey R. Grimmett. *Percolation.* Springer-Verlag Berlin Heidelberg, 1999.

[71] Béla Bollobás and Oliver Riordan. *Percolation.* Cambridge Univ. Press, 2006.

[72] Zhenning Kong and Edmund M Yeh. Resilience to degree-dependent and cascading node failures in random geometric networks. *IEEE Trans. Inform. Theory*, 56(11):5533–5546, 2010.

[73] M. Franceschetti, O. Dousse, D.N.C. Tse, and P. Thiran. Closing the gap in the capacity of wireless networks via percolation theory. *IEEE Trans. Inform. Theory*, 53(3):1009–1018, March 2007.

[74] Rahul Roy. The Russo-Seymour-Welsh theorem and the equality of critical densities and the "dual" critical densities for continuum percolation on R^2. *The Annals of Probability*, pages 1563–1575, 1990.

[75] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[76] Bartłomiej Błaszczyszyn and Dhandapani Yogeshwaran. On comparison of clustering properties of point processes. *Advances in Applied Probability*, 46(1):1–20, 2014.

[77] Arman Kiani Bejestani, Anuradha Annaswamy, and Tariq Samad. A hierarchical transactive control architecture for renewables integration in smart grids: Analytical modeling and stability. *IEEE Trans. Smart Grid*, 5(4):2054–2065, 2014.

[78] Florian Dörfler, John W Simpson-Porco, and Francesco Bullo. Breaking the hierarchy: Distributed control and economic optimality in microgrids. *IEEE Trans. Control of Network Systems*, 3(3):241–253, 2016.

[79] Martin Andreasson, Dimos V Dimarogonas, Karl H Johansson, and Henrik Sandberg. Distributed vs. centralized power systems frequency control. In *2013 European Control Conference (ECC)*, pages 3524–3529. IEEE, 2013.

[80] Changhong Zhao, Ufuk Topcu, Na Li, and Steven Low. Design and stability of load-side primary frequency control in power systems. *IEEE Trans. Automatic Control*, 59(5):1177–1189, 2014.

[81] Enrique Mallada, Changhong Zhao, and Steven Low. Optimal load-side control for frequency regulation in smart grids. *IEEE Trans. Automatic Control*, 2017.

[82] Desmond Cai, Enrique Mallada, and Adam Wierman. Distributed optimization decomposition for joint economic dispatch and frequency regulation. *IEEE Trans. Power Systems*, 2017.

[83] Martin Andreasson, Dimos V Dimarogonas, Henrik Sandberg, and Karl H Johansson. Distributed pi-control with applications to power systems frequency control. In *2014 American Control Conference (ACC)*, pages 3183–3188. IEEE, 2014.

[84] Changhong Zhao, Enrique Mallada, and Florian Dörfler. Distributed frequency control for stability and economic dispatch in power networks. In *2015 American Control Conference (ACC)*, pages 2359–2364. IEEE, 2015.

[85] John W Simpson-Porco, Florian Dörfler, and Francesco Bullo. Synchronization and power sharing for droop-controlled inverters in islanded microgrids. *Automatica*, 49(9):2603–2611, 2013.

[86] Nathan Ainsworth and Santiago Grijalva. Design and quasi-equilibrium analysis of a distributed frequency-restoration controller for inverter-based microgrids. In *North American Power Symposium (NAPS), 2013*, pages 1–6. IEEE, 2013.

[87] Marzieh Parandehgheibi, Konstantin Turitsyn, and Eytan Modiano. Distributed frequency control in power grids under limited communication. In *55th Conf. Decision and Control (CDC)*, pages 6940–6945. IEEE, 2016.

[88] Feier Lian, Aranya Chakrabortty, and Alexandra Duel-Hallen. Game-theoretic multi-agent control and network cost allocation under communication constraints. *IEEE Journal on Selected Areas in Communications*, 35(2):330–340, 2017.

[89] Fu Lin, Makan Fardad, and Mihailo R Jovanović. Design of optimal sparse feedback gains via the alternating direction method of multipliers. *IEEE Transactions on Automatic Control*, 58(9):2426–2431, 2013.

[90] Ivan Gutman and W Xiao. Generalized inverse of the Laplacian matrix and some applications. *Bulletin: Classe des sciences mathematiques et natturalles*, 129(29):15–23, 2004.

[91] Moses Charikar, Yonatan Naamad, Jennifer Rexford, and X Kelvin Zou. Multi-commodity flow with in-network processing. *arXiv preprint arXiv:1802.09118*, 2018.

[92] Nan Zhang, Ya-Feng Liu, Hamid Farmanbar, Tsung-Hui Chang, Mingyi Hong, and Zhi-Quan Luo. Network slicing for service-oriented networks under resource constraints. *IEEE Journal on Selected Areas in Communications*, 35(11):2512–2521, 2017.

[93] Hao Feng, Jaime Llorca, Antonia M Tulino, and Andreas F Molisch. Optimal dynamic cloud network control. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2016.

[94] Jianan Zhang, Abhishek Sinha, Jaime Llorca, Antonia Tulino, and Eytan Modiano. Optimal control of distributed computing networks with mixed-cast traffic flows. In *2018 IEEE International Conference on Computer Communications (INFOCOM)*, pages 1–9. IEEE, 2018.

[95] R Kevin Wood. Deterministic network interdiction. *Mathematical and Computer Modelling*, 17(2):1–18, 1993.

[96] Stephen R Chestnut and Rico Zenklusen. Hardness and approximation for network flow interdiction. *Networks*, 69(4):378–387, 2017.

[97] Carl Burch, Robert Carr, Sven Krumke, Madhav Marathe, Cynthia Phillips, and Eric Sundberg. A decomposition-based pseudoapproximation algorithm for network flow inhibition. In *Network Interdiction and Stochastic Integer Programming*, pages 51–68. Springer, 2003.

[98] Cynthia A Phillips. The network inhibition problem. In *Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 776–785. ACM, 1993.

[99] Rico Zenklusen. Network flow interdiction on planar graphs. *Discrete Applied Mathematics*, 158(13):1441–1455, 2010.

[100] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.

[101] Mihalis Yannakakis, Paris C Kanellakis, Stavros S Cosmadakis, and Christos H Papadimitriou. Cutting and partitioning a graph after a fixed pattern. In *International Colloquium on Automata, Languages, and Programming*, pages 712–722. Springer, 1983.

[102] Zizhong Cao, Shivendra S Panwar, Murali Kodialam, and TV Lakshman. Enhancing mobile networks with software defined networking and cloud computing. *IEEE/ACM Transactions on Networking (TON)*, 25(3):1431–1444, 2017.

[103] Abhishek Sinha and Eytan Modiano. Optimal control for generalized network-flow problems. *IEEE/ACM Transactions on Networking (TON)*, 26(1):506–519, 2018.

[104] David Gamarnik. Stability of adaptive and nonadaptive packet routing policies in adversarial queueing networks. *SIAM Journal on Computing*, 32(2):371–385, 2003.