Connectivity in Interdependent Networks

Jianan Zhang and Eytan Modiano, *Fellow, IEEE*

***Abstract*— We propose and analyze a graph model to study the connectivity of interdependent networks. Two interdependent networks of arbitrary topologies are modeled as two graphs, where every node in one graph is supported by supply nodes in the other graph, and a node fails if all of its supply nodes fail. Such interdependence arises in cyber-physical systems and layered network architectures. We study the *supply node connectivity* of a network: namely, the minimum number of supply node removals that would disconnect the network. We develop algorithms to evaluate the supply node connectivity given arbitrary network topologies and interdependence between two networks. Moreover, we develop interdependence assignment algorithms that maximize the supply node connectivity. We prove that a random assignment algorithm yields a supply node connectivity within a constant factor from the optimal for most networks.**

***Index Terms*— Robustness, cyber-physical systems, communication networks, analytical models.**

I. 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 achieve its full functionality. 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 total number of nodes approaches infinity, the interdependent random graphs percolate. The condition for percolation measures 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 percolates.

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.

We develop an analytically tractable model for interdependent networks which aims to capture key robustness metrics for infrastructure networks. In contrast to the random graph models where some assumptions are difficult to justify in infrastructure networks (*e.g.*, very large network size and randomly placed links), we use a deterministic graph model to represent each network, where nodes and edges are specified by the topology of an infrastructure network. We develop metrics that measure the robustness of interdependent networks, by generalizing canonical metrics for the robustness of a single network. Moreover, our 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.

A. Related Work

A closely related model is the shared risk group model [6], [10]–[12], where 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. In interdependent networks, multiple demand nodes in one network may depend on the same node in another network, and they share the same risk (of the supply node's failure). 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) [12]–[14]. Complexity results and approximation algorithms have been

Manuscript received September 4, 2017; revised April 28, 2018; accepted July 4, 2018; approved by IEEE/ACM Transactions on Networking Editor J. Liu. Date of publication September 10, 2018; date of current version October 15, 2018. This work was supported by DTRA under Grant HDTRA1-13-1-0021 and Grant HDTRA1-14-1-0058. *(Corresponding author: Jianan Zhang.)*

The authors are with the Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: jianan@mit.edu; modiano@mit.edu).

Digital Object Identifier 10.1109/TNET.2018.2863715

1063-6692 © 2018 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

developed to compute the minimum number of colors that appear in an *edge cut* that disconnects a colored graph [12], [15]. 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.*, st edge cuts) can be naturally extended to st 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 [16], [17]. Thus, new techniques need to be developed in this paper 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, [6] and [9] developed lightpath routing algorithms that maximize the number of physical link failures that a given logical topology can tolerate. In this paper, 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 [18], [19], 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.

B. Our Contributions

We propose an analytically tractable model for two interdependent networks, and study the impacts of node failures in one network on the other network. 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 st 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.

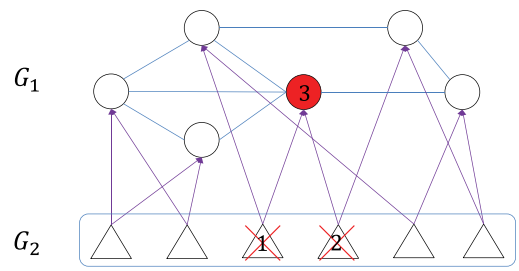


Fig. 1. Demand node 3 fails if both supply nodes 1 and 2 fail.

The rest of the paper is organized as follows. In Section II, 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 III and IV. In Section III, we evaluate the supply node connectivity of the demand network. In Section IV, we develop algorithms, which assign supply nodes to demand nodes, to maximize the supply node connectivity. In Section V, we focus on the bidirectional interdependence model and generalize the above results. Section VI provides simulation results. Finally, Section VII concludes the paper.

II. One-Way Dependent Network Model and Colored Graph Representation

A. 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. 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 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.

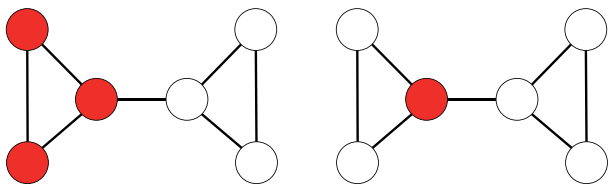


Fig. 2. Let the three red nodes in the left figure be supported by the same supply node. Removing the supply node leads to the failure of the three red nodes, which do not form a proper cut but form a superset of a proper cut (*i.e.*, the red node in the right figure). The supply node is viewed as a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

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.* [20]), 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 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). Under this definition, graphs with larger supply node connectivity are more robust under supply node failures.

Remark 1: In Fig. 2, suppose that every node has a single supply node, and that 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 graph include all the failed nodes in the right graph. 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 st node cut is the *st node connectivity*. Analogously, we define *st supply node cut* and *st supply node connectivity* as follows.

Definition 2: An st supply node cut *is a set of supply nodes whose removals induce an st node cut. (Mathematically, an st 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 st node cut.)*

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

An st 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 st node cut. However, removing $V_d^{st} \setminus \{s, t\}$ must disconnect s from t.

We consider non-adjacent s and t throughout the paper. 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.

B. Transformation to a Colored Graph

Our model is closely related to the shared risk node group (SRNG) model [12], [21]. 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 *st color node cut* as follows.

Definition 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 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 st node cut *is a set of colors C_c^{st} such that the nodes covered by colors C_c^{st} contain an st node cut.*

A minimum color st node cut *is a color st 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 st node cut.*

Colored graph provides an intuitive representation of the correlated node failures using 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 1 that transforms the demand network

[1]Previous studies 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 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 $\tilde{G}_1$.

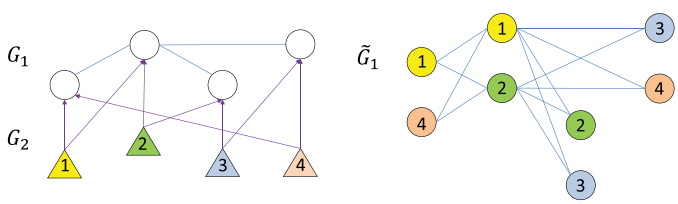


Fig. 3. Illustration of the transformation algorithm.

to a colored graph where every node has a single color, and use Fig. 3 to illustrate the algorithm.

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

Theorem 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 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 st 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 1.

Corollary 1: There is a one-to-one mapping between a supply node st 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: 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 st 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 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 st node connectivity in G_1 and $\tilde{s}\tilde{t}$ node connectivity in $\tilde{G}_1$. □

C. Notations

We define notations to be used throughout the rest of the paper. For a finite set X, the cardinality of X is denoted by $|X|$. For a colored graph $G(V, E, \mathcal{C})$, the number of nodes, edges, and colors are denoted by n, m, n_c, respectively. The graph connectivity is denoted by k, and the st 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 paper. 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 $g(x) = o(f(x))$.

III. 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 $G(V, E, \mathcal{C})$ throughout this section.

A. Complexity

We prove that computing both the global minimum color node cut of a graph and the minimum color st node cut are NP-hard. The proof for the complexity of the minimum color st node cut follows a similar approach to that of the minimum color st edge cut in [12]. In contrast, the complexity of the global minimum color edge cut is unknown. The detailed proofs of Theorems 2 and 3 can be found in the appendix in the supplementary material.

Theorem 2: Given a colored graph, computing the value of the global minimum color node cut is NP-hard.

Theorem 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 values of the minimum color cuts, and then develop polynomial time approximation algorithms.

B. Exact Computation for Arbitrary Colored Graphs

We compute the minimum color st 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 [22].

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 st 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 (4)). This allows the cut nodes to be a *subset* of nodes with colors $\{r|c_r = 1\}$ (recall Definition 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 (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). Moreover, constraint (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 st node cut. The objective is to minimize the number of colors of the cut nodes.

$$\min \sum_{r\in C} c_r \quad \text{(MILP)}$$

$$\text{s.t.} \quad -y_i - y_j \leq p_i - p_j \leq y_i + y_j, \quad \forall (i,j) \in E, \tag{1}$$

$$p_s = 0, p_t = 1, \tag{2}$$

$$y_s = y_t = 0, \tag{3}$$

$$y_v \leq c_r, \forall r \in C, v \in \{v|r \text{ is the color of } v\}, \tag{4}$$

$$p_v, y_v \geq 0, \quad \forall v \in V,$$

$$c_r \in \{0,1\}, \quad \forall r \in C.$$

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 (6) guarantees that there is at least one node that has potential 1, in addition to all the cut nodes. Constraints (5) guarantee that all the cut nodes have potential 1. Constraint (7) guarantees that there is at least one node with 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 (8). Given that M is sufficiently large (*e.g.*, $M = 2|V|$), if $z = 0$, constraint (8) is satisfied; if $z = 1$, constraints (6) and (7) are satisfied. Thus, a node cut that satisfies either condition is a feasible solution of the following IP.

$$\min \sum_{r\in C} c_r \quad \text{(IP)}$$

$$\text{s.t.} \quad -y_i - y_j \leq p_i - p_j \leq y_i + y_j, \quad \forall (i,j) \in E,$$

$$p_v \geq y_v, \quad \forall v \in V, \tag{5}$$

$$\sum_{v\in V} p_v - \sum_{v\in V} y_v - 1 \geq -Mz, \tag{6}$$

$$\sum_{v\in V} p_v - |V| + 1 \leq Mz, \tag{7}$$

$$\sum_{v\in V} y_v - |V| + 1 \geq -M(1-z), \tag{8}$$

$$y_v \leq c_r, \quad \forall r \in C, \ v \in \{v|r \text{ is the color of } v\},$$

$$c_r, p_v, y_v, z \in \{0,1\}, \quad \forall v \in V, \ \forall r \in C.$$

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

Algorithm 2 Computation of the Minimum Color st 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 st 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.

Lemma 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 st 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 st 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 st 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 st cut in G. In Step 1 of Algorithm 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 st cut in G. Similarly, if V^{st} contains both s and t, then $V^{st} \setminus s, t$ is an st cut in G. This concludes the proof that V^{st} is a superset of an st node cut in G. □

Remark 2: A similar result exists in the computation of the minimum color st edge *cut under the condition that all the edges that have the same color are connected [12]. 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 3 computes the exact value of the global minimum color node cut of G where $G[V_i]$ is connected $\forall i$.

Algorithm 3 Computation of the Global Minimum Color Node Cut of G Where $G[V_i]$ Is Connected $\forall i$

1) Compute minimum color st node cut C_c^{st} for all non-adjacent st pairs in G by Algorithm 2. Let C_c^1 denote the minimum size C_c^{st} over all st 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.

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 st node cut in G corresponds to the $s't'$ node cut in G' obtained by Algorithm 2, and that the global minimum color node cut of G can be computed by Algorithm 3. Note that the topology of G' depends on the choice of s and t (see Step 1 of Algorithm 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 $G[V_i]$ is not necessarily connected. To approximate the value of the minimum color st node cut, the algorithm is a slight modification of Algorithm 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.

Lemma 2: The $s''t''$ node connectivity in G'' is at most q times the value of the minimum color st 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 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 st cuts for all non-adjacent st pairs and taking the minimum size cut, and continuing Steps 2 and 3 of Algorithm 3. We conclude this section by summarizing the performance of the approximation algorithms.

Theorem 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 st 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 st node cut can be approximated to within factor q follows from Lemma 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 [20]. The total time of approximating the minimum color st 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 st 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 st node cut can be approximated to within factor q, the minimum over all non-adjacent st 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 st 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.*, [23]), not all the accelerations can be applied to our problem. For example, computing $(k+1)|V|$ pairs of minimum st 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 st node connectivity for $\Theta(|V|^2)$ st pairs should be evaluated. □

IV. 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 V, 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 st node cut for $s, t \in V$) is maximized. Graph G is the transformed graph of the demand graph G_1, obtained by Algorithm 1, where each node is replicated into n_s nodes.

Under the first assumption, according to Corollary 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.

A. Maximizing the st Supply Node Connectivity by Path-Based Assignment

We first propose Algorithm 4 that maximizes the value of the minimum color st node cut, which is simple but provides insight towards maximizing the value of the global minimum color node cut of a graph.

For the k^{st} node-disjoint st paths, any pair of paths do not share the same color if there are sufficient colors ($n_c \geq k^{st}$), by the assignment in Algorithm 4. 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 4 is given by the following theorem.

Algorithm 4 Path-Based Color Assignment

1) Compute the st node connectivity k^{st}. Identify k^{st} node-disjoint st 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.

Theorem 5: The value of the minimum color st 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 st 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 st 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.

B. 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 st connectivity. Indeed, nodes in a *connected dominating set* (CDS) form such a group. A CDS is a set of nodes S such that the induced graph $G[S]$ is connected and that every node in V 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 st 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 [17]–[19]. 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 [17], [19].

We propose Algorithm 5 that assigns colors based on CDS partitions.

The performance of Algorithm 5 can be analyzed in a similar approach to that of Algorithm 4. If $n_c \geq k^{\text{CDS}}$, each CDS has a distinct color, and the graph stays 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 5 is summarized by the following theorem.

Algorithm 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 [19].
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.

Theorem 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.*

C. 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 [19]. We first restate a sampling theorem in [19] as follows.

Lemma 3 [19, Th. 6]: 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 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 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 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 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 [19].

Theorem 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 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 $\eta = \log\frac{en_c}{k'} = \log\frac{2\eta en_c}{\alpha\min(k,n_c)}$ 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 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

$$p_{\text{union-2}} = n(k'/n_c)^k \leq n(1-\epsilon)^k = o(1). \tag{9}$$

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 [19]. 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 $\binom{2\ k'}{k'} \leq\ (2e)^{k'}$. For $k' = \alpha k/(2\log(2e))$,

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

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 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 $O(\log n)$.

V. 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. 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 st supply node connectivity, and is omitted.

A. 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.

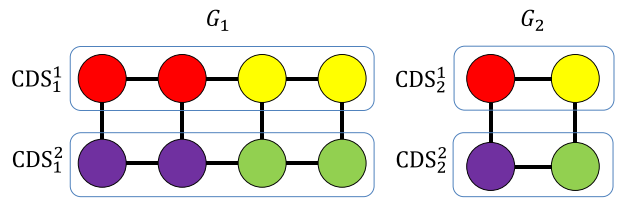


Fig. 4. An example of the partition of CDS nodes into groups. Every node in G_1 and G_2 has $n_{s1} = 1$ and $n_{s2} = 2$ supply nodes, respectively. Each CDS in G_i is partitioned into two groups of size n_{sj} ($i, j \in \{1, 2\}$, $i \neq j$). In each graph, nodes that have the same color are in the 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.

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 6. The remaining nodes (if any) form an occupied group V_i^* if $|V_i|/n_{sj}$ is not an integer.

Algorithm 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.

We denote by h the number of disjoint CDS in G_i. Using the algorithm in [19], $h = \Omega(k_i/\log^2 n_i)$ 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.

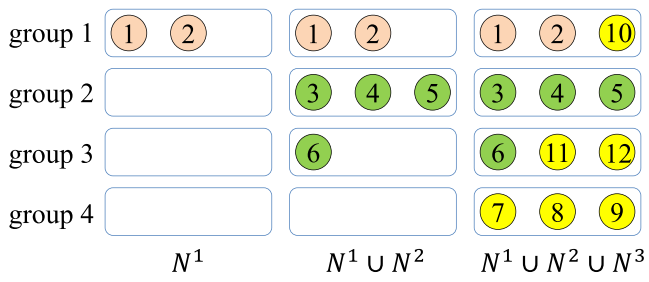


Fig. 5. Partition the CDS nodes $\{N^1, N^2, N^3\}$ into four groups of size three. The left, middle, right figures represent the snapshots after assigning nodes in N^1, N^2, N^3 in Step 2 of Algorithm 6, respectively.

Note that adding extra nodes to a CDS still yields a CDS, since these nodes are adjacent to the nodes in the original CDS.

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. 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. 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. 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. 5).

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 4: Let V_i be assigned to groups according to Algorithm 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 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 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

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

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 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 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 [19], $h = \Omega\ (k_i/\log^2\ n_i)$ disjoint CDS can be found in G_i. By Lemma 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, $\forall i \in \{1,2\}$.

B. 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 1) are removed. In contrast, in the one-way dependence model (Section IV-C), 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 3.

Lemma 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:

$$\begin{aligned}
&\Pr(A(N_{\text{rm}})) \\
&\quad = \sum_{n_{\text{rm}}=0}^{n} \Pr(A(n_{\text{rm}}))\Pr(N_{\text{rm}} = n_{\text{rm}}) \qquad (10)\\
&\quad \leq \Pr(A((1-\epsilon)(1-p)n))\Pr(N_{\text{rm}} \geq (1-\epsilon)(1-p)n) \\
&\qquad + 1\Pr(N_{\text{rm}} < (1-\epsilon)(1-p)n) \qquad (11)\\
&\quad \leq \Pr(A((1-\epsilon)(1-p)n)) + e^{-(1-p)n\epsilon^2/2}, \qquad (12)
\end{aligned}$$

where Eq. (10) follows from the law of total probability, Eq. (11) follows from that $\Pr(A(n_{\text{rm}}))$ is non-increasing in n_{rm}, and Eq. (12) follows from the Chernoff bound. Thus,

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

From the proof of Corollary 3, we know that by removing N_{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. (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 $1-\Pr(A(N_{\text{rm}})) \leq ne^{-\alpha k} + n(1-p)^k$.

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.

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

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 7, and noting Corollary 2, we obtain the following result.

Theorem 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, the transformed graph (by Algorithm 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 5, the supply node connectivity of G_i can be computed in the same approach as the proof for Theorem 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, $\forall i \in \{1,2\}$.

VI. 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

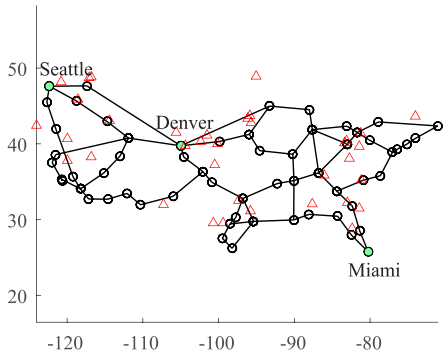


Fig. 6. XO network as a demand network, with randomly generated supply nodes. The x-axis represents longitude degrees (west), and the y-axis represents latitude degrees (north).

use JuMP [24] 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.

A. st Supply Node Connectivity

We use the XO communication network [25] of 60 nodes as an example of the demand network, and randomly generate 36 supply nodes (marked as triangles in Fig. 6) within the continental US.

Let each node in the XO network be supported by three nearest supply nodes. After transforming the network into a colored graph by Algorithm 1 and solving the MILP, we obtain that the supply node connectivity of the st pair Seattle-Denver is 5. In contrast, the maximum st 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 4). As another example, the supply node connectivity of the st pair Seattle-Miami is only 3, because one node in an st path has the same set of three supply nodes as another node in a disjoint st path. By assigning distinct supply nodes to two disjoint paths (Algorithm 4), the supply node connectivity of Seattle-Miami can be increased to 6.

B. 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 5) only guarantees that the supply node connectivity is at least 3, since there do not exist two disjoint CDS in the XO network.

TABLE I

SUPPLY NODE CONNECTIVITY k_i^s OF RANDOM GRAPHS UNDER CDS-BASED AND RANDOM ASSIGNMENTS

n_1	p_1	k_1	n_{s1}	$k_{1\,\max}^s$	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}	$k_{2\,\max}^s$	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

C. 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 $k_{i\,\max}^s = \min(k_i n_{si}, n_j)$ $(i, j \in \{1, 2\}, i \neq j)$. Table I depicts the supply node connectivity k_i^s 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 [19], 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.

VII. CONCLUSION

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 st 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 IV 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.

REFERENCES

[1] V. Rosato, L. Issacharoff, F. Tiriticco, S. Meloni, S. Porcellinis, and R. Setola, "Modelling interdependent infrastructures using interacting dynamical models," *Int. J. Critical Infrastruct.*, vol. 4, nos. 1–2, pp. 63–79, 2008.

[2] M. Parandehgheibi and E. Modiano, "Robustness of interdependent networks: The case of communication networks and the power grid," in *Proc. IEEE GLOBECOM*, Atlanta, GA, USA, Dec. 2013, pp. 2164–2169.

[3] C.-G. Gu *et al.*, "Onset of cooperation between layered networks," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 84, no. 2, p. 026101, 2011.

[4] O. Yagan, D. Qian, J. Zhang, and D. Cochran, "Optimal allocation of interconnecting links in cyber-physical systems: Interdependence, cascading failures, and robustness," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 9, pp. 1708–1720, Sep. 2012.

[5] N. Ghani, S. Dixit, and T.-S. Wang, "On IP-over-WDM integration," *IEEE Commun. Mag.*, vol. 38, no. 3, pp. 72–84, Mar. 2000.

[6] K. Lee, E. Modiano, and H.-W. Lee, "Cross-layer survivability in WDM-based networks," *IEEE/ACM Trans. Netw.*, vol. 19, no. 4, pp. 1000–1013, Aug. 2011.

[7] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, pp. 1025–1028, Apr. 2010.

[8] M. Parandehgheibi, K. Turitsyn, and E. Modiano, "Modeling the impact of communication loss on the power grid under emergency control," in *Proc. IEEE SmartGridComm*, Miami, FL, USA, Nov. 2015, pp. 356–361.

[9] H.-W. Lee, K. Lee, and E. Modiano, "Maximizing reliability in WDM networks through lightpath routing," *IEEE/ACM Trans. Netw.*, vol. 22, no. 4, pp. 1052–1066, Aug. 2014.

[10] D. Papadimitriou *et al.*, *Inference of Shared Risk Link Groups*, document draft-many-inference-srlg-02.txt, Working Draft, IETF Secretariat, Nov. 2001.

[11] J. Q. Hu, "Diverse routing in optical mesh networks," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 489–494, Mar. 2003.

[12] D. Coudert, P. Datta, S. Pérennes, H. Rivano, and M.-E. Voge, "Shared risk resource group complexity and approximability issues," *Parallel Process. Lett.*, vol. 17, no. 2, pp. 169–184, 2007.

[13] S. Yuan, S. Varma, and J. P. Jue, "Minimum-color path problems for reliability in mesh networks," in *Proc. INFOCOM*, Miami, FL, USA, 2005, pp. 2658–2669.

[14] S. Klein, L. Faria, I. Sau, R. Sucupira, and U. Souza, "On colored edge cuts in graphs," in *Sociedade Brasileira de Computaçao, Editor, Primeiro Encontro de Teoria da Computaçao—ETC*. Porto Alegre, Brazil: CSBC, 2016.

[15] P. Zhang, J.-Y. Cai, L.-Q. Tang, and W.-B. Zhao, "Approximation and hardness results for label cut and related problems," *J. Combinat. Optim.*, vol. 21, no. 2, pp. 192–208, 2011.

[16] D. R. Karger, "A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem," *SIAM Rev.*, vol. 43, no. 3, pp. 499–522, 2001.

[17] K. Censor-Hillel, M. Ghaffari, and F. Kuhn, "Distributed connectivity decomposition," in *Proc. ACM Symp. Principles Distrib. Comput.*, Paris, France, 2014, pp. 156–165.

[18] K. Censor-Hillel, M. Ghaffari, and F. Kuhn, "A new perspective on vertex connectivity," in *Proc. 25th Annu. ACM-SIAM Symp. Discrete Algorithms*, Portland, OR, USA, 2014, pp. 546–561.

[19] K. Censor-Hillel, M. Ghaffari, G. Giakkoupis, B. Haeupler, and F. Kuhn, "Tight bounds on vertex connectivity under vertex sampling," in *Proc. 26th Annu. ACM-SIAM Symp. Discrete Algorithms*, San Diego, CA, USA, 2015, pp. 2006–2018.

[20] S. Even and R. E. Tarjan, "Network flow and testing graph connectivity," *SIAM J. Comput.*, vol. 4, no. 4, pp. 507–518, 1975.

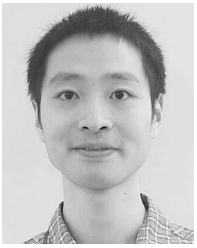
[21] P. Datta and A. K. Somani, "Diverse routing for shared risk resource groups (SRRG) failures in WDM optical networks," in *Proc. Int. Conf. Broadband Netw. (BroadNets)*, San Jose, CA, USA, 2004, pp. 120–129.

[22] D. Bertsimas and J. N. Tsitsiklis, *Introduction to Linear Optimization* (Athena Scientific Series in Optimization and Neural Computation). Belmont, MA, USA: Athena Scientific, 1997.

[23] H. N. Gabow, "Using expander graphs to find vertex connectivity," *J. ACM*, vol. 53, no. 5, pp. 800–844, 2006.

[24] M. Lubin and I. Dunning, "Computing in operations research using Julia," *INFORMS J. Comput.*, vol. 27, no. 2, pp. 238–248, 2015, doi: 10.1287/ijoc.2014.0623.

[25] XO Communications. *Network Map*. Accessed: Aug. 20, 2018. [Online]. Available: https://www.xo.com/about-xo/our-network/complete-network



Jianan Zhang received the B.E. degree in electronic engineering from Tsinghua University, Beijing, China, in 2012, and the M.S. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2014, where he is currently pursuing the Ph.D. degree with the Laboratory for Information and Decision Systems. His research interests include network robustness, optimization, and interdependent networks.

Eytan Modiano (S'90–M'93–SM'00–F'12) received the B.S. degree in electrical engineering and computer science from the University of Connecticut at Storrs in 1986 and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, MD, USA, in 1989 and 1992, respectively. He was a Naval Research Laboratory Fellow from 1987 to 1992, a National Research Council Post-Doctoral Fellow from 1992 to 1993, and a member of the Technical Staff, MIT Lincoln Laboratory, from 1993 to 1999. In 1999, he joined the faculty at the Massachusetts Institute of Technology, where he is currently a Professor with the Department of Aeronautics and Astronautics and an Associate Director of the Laboratory for Information and Decision Systems.

His research interests include the modeling, analysis, and design of communication networks and protocols. He is a Fellow of the IEEE and an Associate Fellow of AIAA. He served on the IEEE Fellows Committee. He was a co-recipient of the Infocom 2018 Best Paper Award, the MobiHoc 2018 Best Paper Award, the MobiHoc 2016 Best Paper Award, the Wiopt 2013 Best Paper Award, and the Sigmetrics 2006 Best Paper Award. He was the Technical Program Co-Chair of the IEEE Wiopt 2006, the IEEE Infocom 2007, ACM MobiHoc 2007, and DRCN 2015. He served as an Associate Editor for the IEEE TRANSACTIONS ON INFORMATION THEORY and the IEEE/ACM TRANSACTIONS ON NETWORKING. He is the Editor-in-Chief of the IEEE/ACM TRANSACTIONS ON NETWORKING.