Improving the Survivability of Clustered Interdependent Networks by Restructuring Dependencies

Genya Ishigaki, *Student Member, IEEE,* Riti Gour, *Student Member, IEEE,*
and Jason P. Jue, *Senior Member, IEEE*

Abstract—The interdependency between different network layers is commonly observed in Cyber Physical Systems and communication networks adopting the dissociation of logic and hardware implementation, such as Software Defined Networking and Network Function Virtualization. This paper formulates an optimization problem to improve the survivability of interdependent networks by restructuring the provisioning relations. A characteristic of the proposed algorithm is that the continuous availability of the entire system is guaranteed during the restructuring of dependencies by the preservation of certain structures in the original networks. Our simulation results demonstrate that the proposed restructuring algorithm can substantially enhance the survivability of interdependent networks, and provide insights into the ideal allocation of dependencies.

Keywords—interdependent networks; network survivability; cascading failure; network function virtualization; cyber physical systems.

I. INTRODUCTION

MANY network systems encompass layering and integration of the layers in both explicit and implicit manners. For example, Software Defined Networking (SDN) decouples the control logic from forwarding functions to realize the flexibility and agility of communication networks. Also, Network Function Virtualization (NFV) involves separation of network function logic from hardware. The concept of separating logic from hardware implementations is also commonly adopted in Cyber Physical Systems (CPS), such as smart grids, in which computing capability manages physical entities.

The dissociation of logic and functions, which is effective for system flexibility, has accelerated the amount of layering and obscure dependencies in network systems. The work [1] on software defined optical networks points out the dependency of logical nodes on physical nodes that provide physical paths for connections among logical nodes, as well as the dependency of physical nodes on the logical nodes through SDN control messages, which define the operations of the physical nodes. Similarly, it is revealed that NFV embraces the interdependency between Virtual Network Functions (VNF) and physical servers hosting the VNFs, when a virtualization

(a) An interdependent network with two constituent graphs representing physical and logical network.

(b) Initial failure at a physical server v_1.

(c) Cascading failure affecting a logical node v_2.

(d) Cascading failure affecting a physical server v'_1. The entire network becomes nonfunctional.
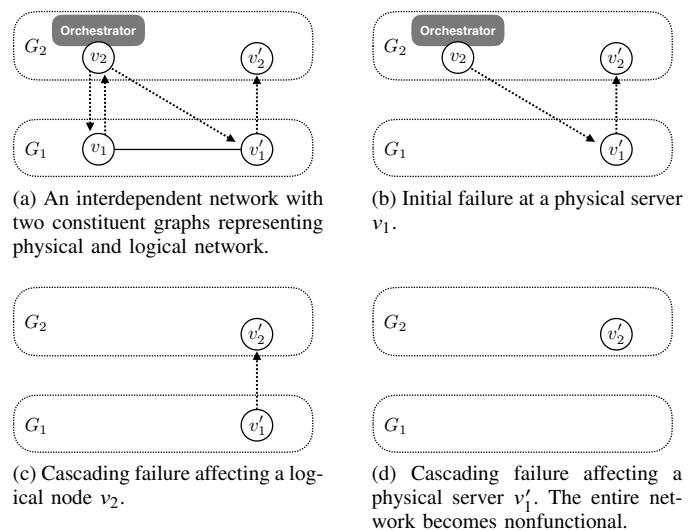
Fig. 1.   An example of cascading failure in an interdependent network representing the dependency between physical servers and NFVs.

orchestrator is recognized as one of the VNFs [2]. Furthermore, the integration of a control information network and an electricity network seen in smart grids is a typical example of the interdependency of two different layers in CPSs [3]. This tendency of layering and collaborative functionality of layered networks is likely to be more evident for next-generation network systems.

However, it has been revealed that certain types of dependencies between different layers of networks can deteriorate the robustness of the entire interdependent system [4]. Consecutive multiple failure phenomena called *cascading failures* exemplify the unique fragility of such network systems. In networks without interdependencies, a failure would influence a certain part of a network. Nonetheless, in networks with interdependencies, some nodes that are not directly connected to the failed portion can become nonfunctional due to the loss of service provisioning from nodes in other layers, which are directly influenced by the initial failure.

Fig. 1 shows an example of such a cascading failure, which starts as a single node failure of v_1 and results in the entire network failure. Suppose that a network G_1 consists of physical servers v_1 and v'_1, and G_2 represents logical computing nodes v_2 and v'_2 hosting VNFs. The orchestrator, which coordinates the mapping between physical and logical

layer, is realized as one of the VNFs on v_2. The arcs from G_1 to G_2 $((v_1, v_2), (v_1', v_2'))$ illustrate the dependency of NFVs or computing nodes on the physical servers, while the arcs from G_2 to G_1 $((v_2, v_1), (v_2, v_1'))$ indicate the dependency of physical servers on a logical node in terms of the flow of coordination messages from the orchestrator to the physical servers. When the physical server v_1 fails, the logical node hosting the orchestrator v_2 loses its dependent physical node v_1, and becomes nonfunctional. This induces another loss of the dependent node of v_1', and eventually the single node failure causes a failure of the whole network.

Cascading failures can also lead to the malfunctioning of CPSs. In fact, it has been reported that some major electricity outages in smart grids, such as the 2003 nation-wide blackout in Italy [5], and the 2004 blackout over 8 states in US and 2 provinces in Canada [6], were due to cascading failures induced from poorly designed dependencies between the electricity network and control information network.

Many contributions have been made since the first theoretical proposal on the cascading failure model by Buldyrev et al. in 2010 [7]. The pioneering works [7], [8] focus on analyzing the behavior of cascading failures rather than proposing design strategies. In contrast, some following works identify vulnerable topologies in interdependent networks to avoid such fragile structures in the design phase by investigating the relation between node degree and failure impacts [9], or evaluating the importance of nodes exploiting the algebraic expression of dependencies [10]. Furthermore, other works propose design strategies in more realistic models to consider the impact of failures caused by a single component [11], integrated factors within and between layers [12], or the heterogeneity of nodes in each layer [13].

This paper discusses a design problem for interdependent networks to improve their survivability, which is a measure of the robustness against a whole network failure, by modifying an existing network topology. The contribution that contrasts our work with other related works is the consideration of existing network facilities. Our method is aimed at redesigning a relatively small part of the existing network to enhance the survivability so that the entire network remains operational even during the restructuring process. In order to realize this continuous availability, a special type of dependency, whose removal does not influence the functionality of the entire system, is identified in the first step of our restructuring method. Our heuristic algorithm increases the survivability of entire systems by the relocations of these dependencies. While our previous work [14] allows a node to have dependencies with any nodes in the other layer, this paper extends the model by considering geographical, economic, or logical accessibility of provisioning by nodes. These constraints are represented as clusters of nodes, and an interdependent network is modeled as a directed graph consisting of multiple clusters. The membership of a node in a specific cluster imposes restrictions on the nodes to which the node can provide support, and the nodes from which the node can receive support. Hence, possible modifications to the dependencies between nodes would vary, depending on the cluster to which a node belongs. Finally, our method is evaluated by simulations in different pseudo

interdependent networks.

II. RELATED WORKS

Most of the preceding works on interdependent networks attempt to analyze the behavior of cascading failures in well-known random graphs, which have certain characteristics in degree distributions and underlying topology [7], [8]. Those works analyze the propagation of failures based on percolation theory developed in the field of random networks. Following the directions shown by a seminal work by Buldyrev et al. in [7], more general models are discussed in [8].

The works [9]–[13] focus on the design aspect of interdependent networks. The relation between the impact of failures and interdependencies is empirically demonstrated to decide appropriate dependency allocations in [9]. A method to evaluate the importance of nodes in terms of network robustness is proposed in [10] by introducing a novel representation of interdependencies based on boolean algebra. This evaluation enables network operators to prioritize the protection of the nodes that contribute more to the robustness of the network. In [12], the authors consider dependency relations not only between layers but also within a single-layer. Combining multiple factors that make a node nonfunctional, their method adjusts the dependency of a node on the other nodes. The work in [13] also considers the influence within a single-layer, supposing the heterogeneity of nodes. In this model, a network can have different types of nodes such as generating and relay nodes. Zhao et al. [11] formulate an optimization problem enhancing the system robustness, defining Shared Failure Group (SFG), a group of nodes that can simultaneously fail due to a cascading failure initiated by the same component.

Another branch of interdependent network research is recovery after failures [15]–[20]. The works in [15]–[17] analyze the behaviors of failure propagations when each node performs local healing, where a functioning node substitutes for the failed node by establishing new connections with its neighbors. The speed of further cascades and resulting network states are revealed by percolation theory [15], [16] or steady state analysis in the belief propagation algorithm [17]. Also, resource allocation problems, which consider the different roles of network nodes are discussed in [18]–[20]. The order of assigning repairing resources is a critical problem during the recovery phase when the amount of available resources is limited. The works in [18], [19] propose node evaluation measurements to decide the allocation, while an equivalent problem in the phase diagram is discussed in [20].

Our work proposes a method to improve the survivability of interdependent networks, following the survivability definition in [21]. Our work would be classified into the category of protection design methods before failures. Specifically, the proposed method is exploited in a redesign process of an existing network to enhance the survivability, while the existing works [9]–[13] discuss the initial design of an entire network. Our protection method, considering the functionality during the redesign, would reduce the cost of survivability improvement in contrast to the entire reconstruction of the systems.

III. Modeling and Motivating Example

In this section, we present a mathematical model for describing interdependent networks, and we present a motivating example of our method. Section III-B summarizes related work [21] defining the survivability for interdependent networks, which we adopt to evaluate the networks.

A. Network Model

An interdependent network consists of k constituent graphs $G_i = (V_i, E_{ii})$ $(1 \leq i \leq k)$ and their interdependency relationships, which are defined by sets of (directed) arcs A_{ij} $(1 \leq i, j \leq k,\ i \neq j)$ representing the provisioning between a pair of nodes in different graphs. Edges in $E_{ii} \subseteq V_i \times V_i$ are called *intra*-edges because they connect pairs of nodes in the same network. In contrast, arcs in $A_{ij} \subseteq V_i \times V_j$ $(i \neq j)$ are called *inter-* or *dependency* arcs. If there exists an arc $(v_i, v_j) \in A_{ij}$ $(v_i \in V_i, v_j \in V_j)$, it means that a node v_j has dependency on a node v_i. The node v_i is called the *supporting* node, and v_j is a supported node. A node v is said to be *functional* if and only if it has at least one functional supporting node.

When an interdependent network is logically partitioned, each constituent graph G_i has a clustering function $\kappa_i : V_i \longrightarrow \{1, 2, ..., \gamma_i\}$, where $\gamma_i \in \mathbb{N}$ is the number of clusters in $G_i = (V_i, E_{ii})$. Then, a graph $I_i^x = (W_i^x \subseteq V_i, E_{ii}(W_i^x))$ induced by a node set $W_i^x = \{v \mid \kappa_i(v) = x\ (1 \leq x \leq \gamma_i)\}$ is called a *cluster*. Note that this definition insists that a node is in exactly one cluster.

In order to emphasize the dependency between constituent graphs, an interdependent network can be represented as a single-layer directed graph $G = (V, A)$, where $V := \bigcup_i V_i$, and $A := \bigcup_{\{(i,j)|i\neq j\}} A_{ij}$ by abbreviating intra-edges. With this notation, a node v is said to be *functional* if and only if $\deg_{in}(v) \geq 1$. Note that all the discussions in the rest of this paper follow this single-layer graph representation.

Additionally, we introduce a different notation of arcs with respect to their source nodes. Let $A(v) \subseteq A$ represent a set of arcs whose source node is $v \in V$. To identify each arc during the restructuring process, where some arc temporarily loses its destination, each arc is denoted as $(v, \cdot)_m$ $(m = 1, ..., \deg_{out}(v))$. The index m is a given fixed identification number for each arc in $A(v)$. Hence, every arc in A can be specified by providing source node v and its identification number m.

A set of constituent graphs is totally ordered by the number of nodes that are the source of at least one intra-arc: $|V_i^{out}|$, where $V_i^{out} := \{v \in V_i \mid A(v) > 0\}$. A constituent graph that has the least number of nodes with outgoing arcs is named the minimum supporting constituent graph G_i: $|V_i^{out}| \leq \min_j |V_j^{out}|$.

B. Survivability of Interdependent Networks

Parandehgheibi et al. [21] propose an index that quantifies the survivability of interdependent networks against cascading failures exploiting the *cycle hitting set*, and they prove that the computation of the survivability is NP-complete. They show that a graph needs to have at least one directed cycle in order to maintain some functional nodes; in other words, the existence
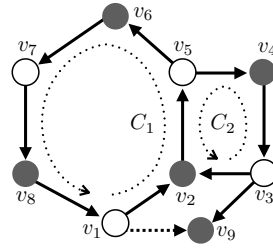
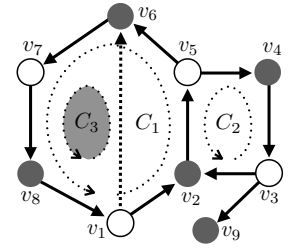

Fig. 2. Graph G with (v_1, v_9).

Fig. 3. Graph G' with (v_1, v_6).

of one cycle prevents an interdependent network from its entire failure. Thus, the survivability of interdependent networks is defined as the cardinality of the minimum cycle hitting set whose removal brings non-functionality for the entire network. Note that a cycle hitting set S is a set of nodes such that any cycle $C = (V(C), E(C))$ in a given graph $G = (V, A)$ has at least one node in the hitting set: $S \cap V(C) \neq \emptyset,\ \forall C \in C(G)$, where $C(G)$ is the set of all cycles in the given graph. This definition implies that the entire failure of an interdependent network occurs when the corresponding graph becomes acyclic. Let $H(G)$ denote a cycle hitting set with the minimum cardinality: $|H(G)| := \min_{S \in \mathcal{S}} |S|$, where $\mathcal{S}$ is the set of all the cycle hitting sets in G. Formally, the survivability of an interdependent network G is the cardinality of the minimum cycle hitting set, $|H(G)|$.

C. Motivating Example

Adopting the survivability definition shown above, improving survivability would be equivalent to increasing the number of disjoint cycles in a graph. Figs. 2 and 3 show an example comparing two similar interdependent networks.

In graph G in Fig. 2, there exists two cycles: C_1 and C_2. If v_2, which is in both $V(C_1)$ and $V(C_2)$, becomes nonfunctional because of a failure, all the nodes in G eventually lose their supporting nodes and become nonfunctional: $H(G) = \{v_2\}$. On the other hand, no single node failure can destroy all the three cycles in G' in Fig. 3, while a two-node failure can make it acyclic (e.g. $H(G') = \{v_2, v_7\}$). Therefore, the graph G' is more survivable than G, since $1 = |H(G)| < |H(G')| = 2$, although they differ only in the destination node of one dependency arc $((v_1, v_9)$ in G or (v_1, v_6) in G'). Supposing that G is an existing topology of a network, a method that relocates (v_1, v_9) to (v_1, v_6) can achieve an enhancement of the survivability.

IV. Problem Formulation

A. Assumptions

This paper deals with the case in which interdependent networks have two types of homogeneous constituent networks with identical dependencies $(k = 2)$. However, our discussion with the restriction on k can be easily extended to more general cases. In more advanced network models, each constituent network can have different types of nodes, such as independently functional generating nodes and relay nodes, which need provisioning from a generating node via paths of intra-edges [13]. Nevertheless, for simplicity, this work follows the

assumption in [21] that each node in a constituent network is directly connected to a reliable conceptual generating node by a reliable edge (homogeneous constituent graphs). Moreover, it is assumed that each supporting node provides a unit amount of support that is enough for a supported node to be operational (identical dependencies), following the same model in [21].

Additionally, this paper presumes that each cluster x receives some support from at least one of the clusters that are supported by cluster x. In other words, this presumption excludes the case that a cluster does not receive provisions from any of the clusters that the cluster is supporting.

B. Requirement Specification

One aspect contrasting our scheme to other works is the consideration to improve the survivability of existing interdependent networks by changing some topological structures. Because all the nodes need to remain functional even during the relocations of dependency relations, it is necessary to avoid the loss of all supporting nodes for any node at any stage of the restructuring. In other words, each node needs to be survivable from a cascading failure, which requires the direct or indirect support by the nodes in directed cycles. This constraint is formally represented as the following rule for the live restructuring.

1) Every node remains reachable from a node in a directed cycle via at least one directed path at any stage of the restructuring.

In addition to guaranteeing the continuous availability, the amount of provisioning provided by each supporting node should remain the same after the restructuring in order to consider the capability of each node. The capability could be, for example, the limit on electricity generation, computation performance, or the number of ports available.

2) The number of supports that a node provides must remain less than or equal to its original provisioning capability.

Furthermore, depending on which cluster a node in graph G_i belongs to, the node has a constraint on clusters in G_j that it can support. The constraint is given by a supportability function $\sigma_{ij} : V_i \longrightarrow 2^{\gamma_j}$, where 2^{γ_j} is the power set of the cluster indices in a constituent network G_j. This means that a node $v (\in V_i)$ can provide its support to the nodes in the clusters of G_j given by the supportability function. This specification corresponds the geographical, economic, or logical constraints on the accessibility of supports from a node to specific groups of nodes. For example, it is impossible for information control node v to have electricity supply from node u if v and u are geographically far apart or managed by different administrative institutions. The geographical or administrative domain is shown as a cluster in each constituent graph, and dependency relations of the nodes should be closed within a set of permitted nodes, which are geographically close, or managed by the same company or allied companies, since each cluster should be independent from the outsiders. This constraint relating to network clustering is simply expressed as follows.

3) All the provisionings from a node u are directed towards the nodes in the clusters that u can support, as designated by the supportability function σ_{ij}.

C. Clustered ΔH Problem

This section formulates the clustered ΔH problem, which is aimed at enhancing the survivability of a given interdependent network with clusters by restructuring dependency relationships, considering the continuous availability, supporting capability, and clustering constraint of each node.

Considering the continuous availability of an existing network during restructuring leads to the formulation of a gradual reconstruction problem, where no relocation of two or more different arcs is conducted at a time. Each phase relocating one arc is named a *step*. Let $G^s = (V, A^s)$ denote the graph representing the interdependent network topology at step s. The improved interdependent network G^{s+1} after step s consists of a node set V, which is the same node set as in graph G^s, and an arc set A^{s+1} amended by the relocation of an arc $(u, v) \in A^s$ to (u, v'), where $v' \in V$ is a new destination for the arc (u, v).

The clustered ΔH problem is to maximize the difference in survivability between a given interdependent network, which is recognized as G^0, and the resulting network after a sequence of consecutive improvements. The resulting network is represented as G^f, where f denotes the step at which the last arc relocation is completed. Formally, the objective is to maximize the difference between $|H(G^0)|$ and $|H(G^f)|$, which is defined as ΔH.

Problem (Clustered ΔH Problem). For a given $G^0 = (V = \bigcup_i V_i, A^0)$, the number of clusters $\gamma_i \in \mathbb{N}$ in each constituent graph G_i, a clustering function $\kappa_i : V_i \longrightarrow \{1, 2, ..., \gamma_i\}$ for each constituent graph G_i, and supportability functions $\sigma_{ij} : V_i \longrightarrow 2^{\gamma_j}$, maximize $\Delta H := |H(G^f)| - |H(G^0)|$, where $G^{s+1} = (V, A^{s+1})$ $(0 \le s \le f - 1)$ is obtained by the relocation of the destination of a single arc in A^s: $A^{s+1} = A^s \setminus (u, v) \cup (u, v')$, satisfying

1) $\deg_{in}(v)_{G^s} \ge 1 \quad \forall v \in V$,
2) $\deg_{out}(v)_{G^{s+1}} = \deg_{out}(v)_{G^s} \quad \forall v \in V$,
3) $\kappa_j(v \in V_j) \in \sigma_{ij}(u \in V_i) \quad \forall (u, v) \in A^s$.

These three conditions correspond to the three rules described in Section IV-B. The second and third conditions are easily derived from the corresponding rules. Lemma 1 shows the equivalence of the condition 1 and Rule 1.

Lemma 1. When $\deg_{in}(v)_G \ge 1$ $(\forall v \in V)$ in a connected directed graph $G = (V, A)$, (a) G has at least one directed cycle, and (b) any node $v \in V$ is reachable from a node $u \in V$ that is contained in a directed cycle.

Proof. $\deg_{in}(v)_G \ge 1$ $(\forall v \in V)$ insists that any node v has at least one parent v'. The path $v \leftarrow v' \leftarrow ...$ composed by repeating the trace of parents can be acyclic until the length of the path is $|V - 1|$. However, the $|V|$th node must have at least one parent from the assumption. Thus, the pigeonhole principle indicates that it is necessary that the path forms a directed cycle. $\square$
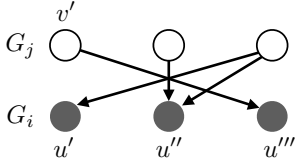
Fig. 4. Original Dependencies, where (v', u') is missing. Note that this figure only shows A_{ji}. The symmetric discussion can be done for A_{ij}.
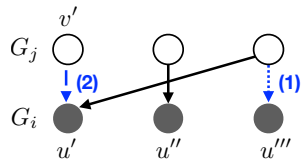
Fig. 5. Relocation Steps (1) to maintain the functionality of u''', and (2) to form a length-2 cycle with v' and u'.

D. Problem Analysis

This section provides the analysis on the trivial optimal case of the clustered ΔH problem with a special setting, where each of constituent graph consists only of one cluster. Let $\rho((u, \cdot)_m)$ denote the number of relocations that arc $(u, \cdot)_m \in A$ experienced during the restructuring process. Note that $\sum_{u \in V} \sum_{m=1}^{\deg_{out}(u)} \rho((u, \cdot)_m) = f$.

From the definition, the optimum survivability cannot exceed the number of supporting nodes, which each have at least one outgoing arc, in the minimum supporting constituent graph G_i. This is because a set of such nodes covers all the directed cycles in an interdependent network G. This observation implies that the optimum survivability is achieved when every node $v_i \in V_i$ of G_i has an injective mapping to a node in V_j ($j \neq i$). In other words, for each node v_i in G_i, there exists at least one unique disjoint cycle whose length is 2 with v_j in G_j. The following lemma gives a sufficient condition to reach the ideal state by repeated relocations while preserving the problem constraints.

Lemma 2. When the number of relocations for each arc $\rho((u, \cdot)_m)$ is not upper bounded, in order to have the optimum restructuring, it is sufficient that the minimum supporting constituent graph G_i satisfies $|V_j| < \sum_{u \in V_i} |A(u)|$ and $\sum_{v \in V_j} |A(v)| > |V_i|$ ($j \neq i$). Then, the optimum survivability becomes $|V_i^{out}|$.

Proof. The maximum survivability achievable by restructuring is equal to the number of nodes that have at least one outgoing arc $|V_i^{out}|$ in the minimum supporting constituent graph $G_i = (V_i, E_{ii})$, because the removal of such nodes from G_i must destroy all the cycles between G_i and another constituent graph. In order to achieve the maximum survivability via the restructuring process, it is necessary that each node $u \in V_i^{out}$ belongs to a cycle whose length is 2. Otherwise, the cycle contains another node $w \in V_i^{out}$, and the removals of such w's make u lose all incoming arcs. Note that a node in $V_i \setminus V_i^{out}$ is never a part of directed cycles, since it has no outgoing arc.

Suppose that we have the minimum supporting constituent graph G_i and another constituent graph G_j that satisfy the two conditions in the lemma. From the definition of the minimum supporting constituent graph, we can make $|V_i^{out}|$ pairs of nodes $\langle u \in V_i^{out}, v \in V_j^{out} \rangle$, which are expected to form a length-2 cycle together after restructuring, so that no two nodes in V_i are paired with the same node in V_j^{out}.

Figs. 4 and 5 illustrate a general example of a restructuring process to form such a length-2 cycle by dependency arc relocations. Note that the figures only show A_{ji}, but the symmetric

argument can be done for A_{ij}. Let $\langle u' \in V_i^{out}, v' \in V_j^{out} \rangle$ be a pair such that $(v', u') \notin A_{ji}$. In order to make a length-2 cycle between v' and u', the arc (v', u''') should be relocated to (v', u'). However, the relocation makes u''' lose all of its incoming arc. The loss of incoming arc of u''' is always avoided by relocating one of the arcs incoming to u'' to u''' (See Figs. 4 and 5 (1)). The supposition in the lemma and the pigeonhole principle suggest the existence of at least one node $u'' \in V_i$ that has two incoming arcs. After the adjustment of the provisioning for u''' by this relocation, the arc (v', u''') can be relocated to (v', u') (See 4 and 5 (2)).

For a pair $\langle u' \in V_i^{out}, v' \in V_j^{out} \rangle$ such that $(u', v') \in A_{ij}$, similar relocations are always possible, because $|V_j| < \sum_{u \in V_i} |A(u)|$. Thus, these relocations eventually achieve the maximum survivability by forming $|V_i^{out}|$ length-2 cycles that each consist of a pair $\langle u \in V_i^{out}, v \in V_j^{out} \rangle$.   □

Some propositions similar to Lemma 2 appear in related literature [11], [22]. The sufficient condition provided in Lemma 2 allows the entire restructuring of inter-arcs by repeated relocations of each arc. Therefore, the ΔH problem is recognized as a design problem of an entire interdependent network discussed in [11] under these assumptions. Also, the work [22] claims that such a one-to-one provisioning relation realizes the robustness, while assuming certain structural characteristics of random graphs.

However, it is unrealistic to relocate a dependency arc many times, when considering the overhead of the changes of provisioning relations in network systems. Therefore, the following part of our paper discusses the case where the number of relocations are strictly restricted: $\rho((u, \cdot)_m) \leq 1$ ($1 \leq m \leq \deg_{out}(u), \forall u \in V$). Under this condition, it cannot be guaranteed to obtain the optimum survivability even when the sufficient condition above holds.

V. HEURISTIC ALGORITHM FOR ΔH PROBLEM

This section proposes a heuristic algorithm for the clustered ΔH problem. Before providing the details of our heuristic algorithm, we first define special types of arcs named Marginal Arcs (MAs), which are candidates for the relocations in Section V-A. Then, the heuristic algorithm, which consists of two algorithms: Find-MAs and ΔH, is described. The Find-MAs algorithm enumerates all the arcs that match the definition of MAs. With the set of MAs found by the Find-MAs algorithm, the ΔH algorithm decides appropriate relocations of the dependency arcs in the set, considering disjointness of newly formed cycles, so that it can improve the survivability of a given network.

After the discussion for a simple case with only one cluster in each constituent graph in Sections V-B to V-C, Section V-D explains how the other cases with multiple clusters are broken down into the simple case.

A. Restructuring of Dependencies

In order to guarantee continuous availability, it is necessary to classify the dependency arcs into either changeable or fixed arcs. However, it is computationally difficult to know the
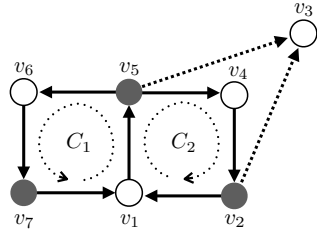
Fig. 6. Original graph G with Marginal Arcs (v_2, v_3) and (v_5, v_3).
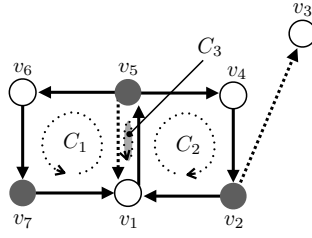
Fig. 7. Modified graph G' with a new arc (v_5, v_1).
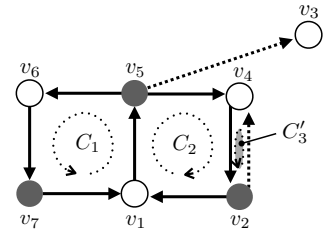
Fig. 8. Modified graph G'' with a new arc (v_2, v_4).

classification beforehand under the condition of $\rho((u, \cdot)_m) \leq 1$ ($\forall u \in V$), because this process involves enumeration of all the permutations of arc relocations and their combinations of destinations. Thus, in this paper, the classification is simplified by using a sufficient condition, while this enumeration is likely to become another optimization problem for a further investigation.

As observed in Section III-C, increasing disjoint cycles in a given network could be an important factor to enhance overall survivability. Hence, our method maintains all existing cycles, which is sufficient to avoid cascading failures, and tries to reallocate the destinations of the arcs that do not belong to directed cycles and that do not make their descendant nodes nonfunctional. Let the arcs that are not in any cycles in a given directed graph $G = (V, A)$ be called *Marginal Arcs* (MAs). Formally, the set $M \subsetneq A$ of MAs is defined as

$$M := \{(u, v) \mid (u, v) \notin A(C) \ \forall C \in C(G)\}. \tag{1}$$

Lemma 3. A removal of any marginal arc never decreases the survivability of an interdependent network: $|H(G)| \leq |H(\overline{G})|$, where G is a given graph, and $\overline{G}$ is the graph obtained by the removal.

Proof. Let M be a set of marginal arcs. From the definition of MAs (Eq. (1)), the removal of MAs does not destroy or connect any existing cycles in $G = (V, A)$. Therefore, $|H(G)| = |H(\overline{G})|$, where $\overline{G} = (V, A \setminus M)$. $\square$

Moreover, appropriate relocations of the removed MAs could improve the survivability of interdependent networks, assuring operability during the relocation process and maintaining the provisioning capability of each node. Let us analyze the effect of dependency relocations using simple examples in Figs. 6-8. The given graph G in Fig. 6 has two marginal arcs: $M = \{(v_2, v_3), (v_5, v_3)\}$. In order to maintain at least one supporting node for v_3, one of the MAs has to remain the same, and the other can be relocated. Fig. 7 shows the case of relocating (v_5, v_3) to (v_5, v_1); on the other hand, Fig. 8 indicates the case of relocation of (v_2, v_3) to (v_2, v_4). Even though one new cycle (C_3 and C'_3 respectively) is formed by each relocation, the modified graphs G' and G'' have different survivability: $|H(G')| = 1$ ($= H(G)$), and $|H(G'')| = 2$. This is because the cycles in G' are not disjoint with each other: $V(C_1) \cap V(C_2) \cap V(C'_3) \neq \emptyset$; in contrast, $V(C_1) \cap V(C_2) \cap V(C''_3) = \emptyset$ in G''. Therefore, it could be said that the appropriate relocation for improving survivability is to form disjoint cycles.

Algorithm 1 ΔH-algorithm(G, l)

Input: subgraph (directed graph) $G = (V, A)$, maximum hop $l \in \mathbb{N}$ (odd)

1: $M \leftarrow$ find-MAs(G) # $M \subset A$
2: **for** each $(v, w) \in M$ **do**
3:      **if** $\deg_{in}(w) \geq 1$ after $A \setminus \{(v, w)\}$ **then**
4:          **while** True **do**
5:              pick $C \in C(v)$ (randomly)
6:              **for** $i \leftarrow l; i > 0; i \leftarrow i - 2$ **do**
7:                  pick $u \in V(C) : \overline{d_C}(v, u) = i$
8:                  **if** $u \notin U$ **then**
9:                      $A \leftarrow A \setminus (v, w) \cup (v, u)$
10:                     $U \leftarrow U \cup \{n \mid \overline{d_C}(v, n) \leq i\}$
11:                     break to next arc in M (line 2)
12:                 **end if**
13:             **end for**
14:         **end while**
15:         pick $(u, v) \in A_{in}(v)$ (randomly) # Minimal-add process (line 15,16)
16:         $A \leftarrow A \setminus (v, w) \cup (v, u)$
17:     **end if**
18: **end for**

B. Find-MAs Algorithm

The Find-MAs algorithm first distinguishes MAs M, which are candidate arcs for relocations, from the arcs in directed cycles in a given graph $G = (V, A)$, by employing Johnson's algorithm [23]. Johnson's algorithm enumerates all elementary cycles in a directed graph within $O((|V|+|E|)(|C(G)|+1))$. It is enough for distinguishing MAs to obtain elementary directed cycles because any non-elementary cycle can be divided into multiple elementary cycles within which dependency relationship are closed. After the enumeration of cycles in G by Johnson's algorithm, the set of MAs is obtained by $M \leftarrow A \setminus \bigcup_{C \in C(G)} A(C)$.

C. ΔH Algorithm

With the set of MAs obtained by Johnson's algorithm, the ΔH algorithm (shown as pseudo code in Algorithm 1) relocates the destinations of MAs, considering disjointness of newly created cycles. (See the discussion in Section V-A.) For each MA (v, w), our algorithm first checks whether or not the relocation of this MA causes the loss of supports for the current destination w: $\deg_{in}(w)_{\overline{G}=(V, A \setminus \{(v, w)\})} \geq 1$ (line 3).
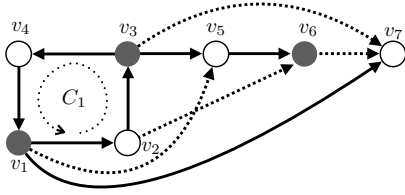
Fig. 9.   A given graph G with $M = \{(v_1, v_5), (v_2, v_6), (v_3, v_7), (v_6, v_7), (v_1, v_7), (v_3, v_5), (v_5, v_6)\}$.
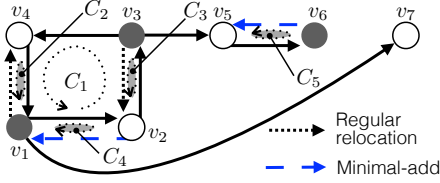

Fig. 10.   A modified graph G' with new arcs: $(v_1, v_4), (v_3, v_2), (v_2, v_1), (v_6, v_5)$.

If w still has some supporting node after the removal of (v, w), the next step is determining a new destination for $(v, \cdot)$. Our algorithm randomly selects one of the cycles that contains the source v denoted by $C \in C(v)$ (line 5). There may be multiple possible candidate nodes for a new destination in the cycle C. Thus, the new destination is decided by the size of the newly formed cycle, which is a result of the relocation (line 6, 7). To represent the size of the newly formed cycle, the distance from a node v to a node u in an (existing) cycle C in the counter direction is denoted as $\overline{d_C}(v, u)$ in our pseudo code. When the maximum hop is designated by l, the algorithm tries to make a new cycle with size $l + 1$ using a node u, such that $\overline{d_C}(v, u) = l$, as the destination of the MA. If it fails to form the cycle, it attempts to compose a smaller cycle using a node u' such that $\overline{d_C}(v, u') = l - 2$. Because of the definition of the dependency, an arc must span between two different layers or constituent networks. Since the node at $\overline{d_C}(v, u) = l - 1$ in C is in the same constituent network as the source node v, it cannot be a new destination.

Consider an example using a given graph G shown in Fig. 2 and the restructured graph in Fig. 3. Since the removal of (v_1, v_9) does not make v_9 lose all its incoming dependency arcs, our algorithm tries to relocate the destination of this arc to one of the nodes in the cycle C_1, which are v_2, v_6, v_8. For instance, in the case $l = 3$, a new cycle C_3 is formed as depicted in Fig. 3 by choosing v_6, that satisfies $\overline{d_{C_1}}(v_1, v_6) = l (= 3)$. Similarly, if l is initialized to 1, a new cycle C_3 is formed using $\{v_1, v_8\}$.

After selecting a destination candidate u in line 7, our algorithm checks if u is already used to create a new cycle (line 8). This is confirmed by a set of nodes U storing all the nodes that are in newly formed cycles: $\{n \mid \overline{d_C}(v, n) \le i\}$ (line 10). For instance in Fig. 3, $U \leftarrow U \cup \{v_1, v_6, v_7, v_8\}$. As will be understood, when another MA tries to form a new cycle using one of these nodes in U, the new cycle and C_3 share some nodes, which means that those cycles are not disjoint. Also, the arc set A is updated when the new destination is finally fixed (line 9).

If there exists no possible destination for an MA (v, w) that satisfies all the conditions, the relocation of the MA is

conducted by randomly selecting an incoming arc of v, (u, v) and relocating (v, w) to (v, u), so that it composes a cycle of length 2 (line 15, 16). This random selection is named *Minimal-add* process.

The MAs relocated by the Minimal-add process satisfy either of the following cases: 1) The node v does not belong to any cycles: $C(v) = \emptyset$, or 2) all the nodes in the cycles of $C(v)$ are already used to compose new cycles by other MAs. Figs. 9 and 10 show examples of these two conditions (dashed arcs). A given graph G has the MA set $M = \{(v_1, v_5), (v_2, v_6), (v_3, v_7), (v_6, v_7), (v_1, v_7), (v_3, v_5), (v_5, v_6)\}$. Eventually, the ΔH algorithm respectively relocates (v_1, v_5) and (v_3, v_7) to (v_1, v_4) and (v_3, v_2). Because v_6 is not in any cycles in G (reason 1), the Minimal-add process picks the source of one of the current incoming arcs in $A_{\text{in}}(v)$, v_5 as the new destination. Also, (v_2, v_6) does not have any possible destinations that are not in the set U (reason 2), and it is relocated to (v_2, v_1) by the Minimal-add process.

D. Application to Clustered Networks

Our heuristic algorithm employs another algorithm named *Decompose-cluster* to form subgraphs, which indicate candidate destinations for the MAs in each cluster, from a given interdependent network. When interdependent networks are clustered, the modification of the destinations of MAs needs to be conducted under more constraints given by supportability functions σ_{ij}: $\kappa_j(v \in V_j) \in \sigma_{ij}(u \in V_i) \ \forall (u, v) \in A$. The Decompose-cluster algorithm selects each cluster (node set W_i^x ($1 \le i \le k$, $1 \le x \le \gamma_i$)) and collects MAs (u, v) whose sources are in the cluster ($u \in W_i^x$), or whose destinations and sources are respectively in the cluster W_i^x and in a cluster in $\sigma_{ij}(v)$ ($v \in W_i^x$ & $\kappa_j(u \in V_j) \in \sigma_{ij}(v)$)). Using the collected MAs and their endpoints, a subgraph Y for reallocations of MAs in W_i^x is composed. Each subgraph for each cluster is given to the ΔH-algorithm so that it can improve the survivability by restructuring dependencies in the subgraph.

As will be understood, no directed cycles exist if no MA matches the condition of $v \in W_i^x$ & $\kappa_j(u \in V_j) \in \sigma_{ij}(v)$. However, this is not going to happen in our work due to the assumption mentioned in Section IV-A. Note that the absence of such MAs means that nodes in a cluster x are not provided any support by the nodes that receive some supports from the nodes in the cluster x.

E. Complexity Analysis

The Decompose-cluster algorithm extracts $\sum_{i=1}^{k} \gamma_i$ subgraphs from a given graph $G = (V, A)$. The number of clusters γ_i in each constituent graph tends to be much smaller than the number of nodes; thus, $\sum_{i=1}^{k} \gamma_i$ can be considered as a constant. In order to compose each subgraph, the algorithm requires to check the source and destination of each arc in A. However, each edge appears in exactly one subgraph because of the used edge set D. Therefore, the total complexity of the Decompose-cluster algorithm is $O(|V| + |A|)$.

The complexity of the ΔH-algorithm is sensitive to the number of cycles in the interdependent network. It is known that Johnson's algorithm finds all elementary cycles within

---

Algorithm 2 Decompose-cluster(G)

Input: interdependent network (directed graph) $G = (V = \bigcup_{i=1}^{k} V_i, A)$, clustering functions κ_i

1: $D \leftarrow \emptyset$
2: **for** a node set W_i^x ($1 \leq i \leq k$, $1 \leq x \leq \gamma_i$) **do**
3:    $P \leftarrow \emptyset$, $R \leftarrow \emptyset$
4:    **for** each $(u, v) \in A \setminus D$ **do**
5:       **if** $u \in W_i^x$ or ($v \in W_i^x$ & $\kappa_j(u \in V_j) \in \sigma_{ij}(v)$) **then**
6:          $P \leftarrow P \cup \{u, v\}$
7:          $R \leftarrow R \cup (u, v)$
8:          $D \leftarrow D \cup (u, v)$
9:       **end if**
10:    **end for**
11:    compose graph $Y = (P, R)$
12:    ΔH-algorithm(Y, l)
13: **end for**

$O((|V| + |E|)(|C(G)| + 1))$. The ΔH-algorithm determines a new destination after $\frac{l}{2} \times C(G)$ searches for each MA, in the worst case. When only one cycle whose size is 2 exists in the input, and the other nodes are supported by the cycle, the size of the set M becomes $|E| - 2$. It is obvious that the complexity of the Minimal-add process is $O(1)$, so the worst case analysis takes the case where all MAs are reallocated by the ΔH-algorithm. Thus, its complexity is $O((|V|+|E|)(|C(G)|+1))+O((|E|-2)(\lceil \frac{l}{2} \rceil \times |C(G)|))$. Assuming the maximum hop l is small enough to be considered as a constant, the overall complexity of our heuristic algorithm becomes $O((|V| + |E|)|C(G)|)$. Note that the assumption on l is valid with our strategy, which tries to increase disjoint directed cycles in a given graph.

F. Optimality in Special Graphs

In order to analyze the performance of our heuristic algorithm, we consider the survivability improvement in special graphs where either an exhaustive search gives us the optimum survivability, or some special properties allow us to compute the optimum.

In the analysis, the upper bound of the survivability improvement, which is used as a benchmark for the rest of this paper, is calculated based on the number of the MAs that satisfy the following two conditions. First, let V_s be a set of nodes that hold more than one MA, and M_s be a set of MAs whose source nodes are in V_s. Even when the MAs from $v \in V_s$ form more than one new cycles, the removal of such a source node v can destroy all the newly formed cycles. This indicates that restructuring increases the survivability by at most $|V_s|$, when relocating MAs in M_s. Second, let V_d be a set of nodes whose incoming arcs are all MAs, and M_d be a set of MAs whose destination nodes are in V_d. If all the MAs incident to $v \in V_d$ are relocated, v loses its functionality during this restructuring. Therefore, at least one MA should remain as an incoming arc to v. This implies that the number of cycles newly formed by the MAs in M_d is at most $|M_d| - |V_d|$. Thus, the upper bound U is obtained by $|M| - |M_s| + |V_s| - |V_d|$.

Fig. 11 illustrates a comparison of our algorithm with the optimum solution in a small interdependent network such that
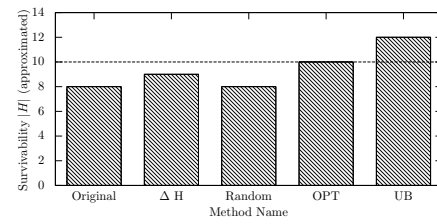

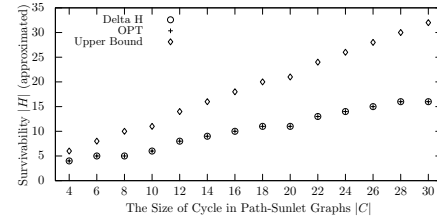Fig. 11.   Numerical comparison with the optimum solution in a small interdependent network.

Fig. 12.   Survivability of MA-saturated Path-Sunlet graphs $\zeta_2(G \in \mathcal{L})$ with two length-3 paths: $|\mathcal{P}| = 2$, $k_i = 3$ ($\forall P_i \in \mathcal{P}$).

each constituent graph has 15 nodes, and the number of dependency arcs is 84, including 5 MAs. The optimum solution is obtained by an exhaustive search of 759,375 combinations of reallocations. This numerical example shows that the solution given by the ΔH algorithm would not provide solutions that are exceptionally divergent from the optimum solution. It also infers that the upper bound is not tight in general.

Fig. 12 indicates that the survivability obtained by our restructuring heuristic algorithm matches the optimum in a special class of graphs, which are named MA-saturated Path-Sunlet Graphs $\zeta_2(G)$, $G \in \mathcal{S}$. The optimum value of survivability for these graphs is always computable based on the following discussion.

Definition 1. Path-Sunlet Graphs $\mathcal{L}$: A set of graphs satisfying the following conditions are named Path-Sunlet graphs. Let $\mathcal{L}$ denote the set of Path-Sunlet graphs.

- $G \in \mathcal{L}$ only has one cycle C.
- The arcs that are not in the cycle C form a set of disjoint paths whose initial nodes are in C: $\mathcal{P} = \{P_i = (v_1^i, v_2^i, ..., v_{k_i}^i) \mid v_1^i \in C$ and $P_i \cap P_j = \emptyset$ ($\forall P_{j \neq i} \in \mathcal{P}$)$\}$.

Definition 2. MA-saturation $\zeta_\delta(G)$ of a graph G: The MA-saturation is an operation of adding additional arcs to a given graph until any addition of an arc makes the graph non-simple, maintaining the out-degree constraint that the out-degree of any node does not exceed a given constant $\delta \in \mathbb{N}$.

Remark. The optimal restructuring of MAs in MA-saturated Path-Sunlet graphs $\zeta_2(G)$, $G \in \mathcal{L}$ consists of forming length-2 cycles using an MA and an edge in either $P_i \in \mathcal{P}$ or C.

We consider the cases where $|\mathcal{P}| \geq 1$, because the survivability in the case of $|\mathcal{P}| = 0$ is obviously $\left\lceil \frac{|V(C)|}{2} \right\rceil$.

Lemma 4. By removing arcs that are not in any cycle, the optimally restructured MA-saturated Path-Sunlet graph $\zeta_2(G)$ is decomposed into some sequence of cycles.

Proof. Three or more cycles do not meet at the same node, since $\delta = 2$. Therefore, the only possible topology with multiple length-2 cycles is a chain of cycles, in which two cycles share exactly one node. □

Lemma 5. The survivability of the optimally restructured MA-saturated Path-Sunlet graphs $\zeta_2(G)$, $G \in \mathcal{L}$ is $\sum_{q \in Q} \left\lceil \frac{q}{2} \right\rceil$, where Q is the set of all the sequences of cycles obtained by removing the arcs that are not in any cycles.

Proof. A removal of one node that is shared by two cycles breaks the two cycles. When q is even, the process gives us the survivability of $\frac{q}{2}$. If q is odd, one additional removal is needed to destroy the remaining cycle. Thus, the survivability of a sequence of q cycles is $\left\lceil \frac{q}{2} \right\rceil$.

Since each sequence in Q is disjoint with the other, the survivability of the entire graph is obtained by summing up the survivability of each sequence. □

VI. SIMULATION

In order to understand the performance of the proposed algorithm, our simulations are conducted in both non-clustered and clustered interdependent network models of different sizes. The results from the simplest cases where each constituent network only consists of one cluster (non-clustered) are first described, and the clustered cases follow.

A. Network Topology

The performance of the proposed algorithm is analyzed in random directed bipartite graphs that contain at least one directed cycle. Assuming the situation in which a current interdependent network is working normally, each node is either a member of some cycle or reachable from a node in a cycle through some directed path in the input graph. Because our algorithm only concerns the dependency arcs between 2 constituent graphs ($k = 2$), any interdependent network is represented as a directed bipartite graph whose arcs connect a pair of different types of nodes.

Each random bipartite graph is generated by specifying the following parameter: V_i, $\max_{v \in V} \deg_{in}(v)$ and $\min_{v \in V} \deg_{in}(v)$. In order to observe the performance in different conditions, experiments are conducted in symmetric and asymmetric interdependent networks. A symmetric interdependent network has constituent networks which each have identical number of nodes: $|V_1| = |V_2|$, while constituent networks of an asymmetric interdependent network have different number of nodes: $|V_1| = \frac{|V_2|}{q}$ ($q \in \mathbb{N}$). The degree of each node is determined based on the uniform distribution between the given maximum and minimum incoming degree.

B. Clustering Settings

As the non-clustered cases have symmetric and asymmetric constituent graphs, clustered interdependent networks are also examined in three patterns of topology configurations. In our simulations, each constituent graph has three clusters: W_i^1, W_i^2 and W_i^3 ($i = 1, 2$) (See Fig. 13). In symmetric cases, a pair of corresponding clusters in different constituent graphs have the
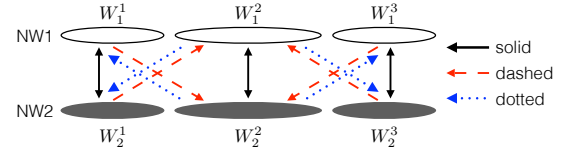

Fig. 13. Dependency models of clustered interdependent networks. Arrows show the dependency relationships between clusters. Model 1: solid. Model 2: solid and dashed. Model 3: solid, dashed, and dotted.

same number of nodes: $W_1^x = W_2^x$, while a cluster is half-sized to the corresponding cluster in the other constituent graph in asymmetric models: $W_1^x = \frac{W_2^x}{2}$.

Also, Fig. 13 illustrates the three models that have different dependency relationships indicated as arrows. Note that when an arrow is drawn from W_i^x to $W_j^{x'}$, it means that the nodes in cluster $W_j^{x'}$ can have supports from the nodes in W_i^x. Model 1 consists only of the solid arrows, which means that each pair of corresponding clusters has dependency relationships. Model 2 has the dependencies illustrated by the solid and dashed arrows, while Model 3 has all the arrows (solid, dashed and dotted). A major difference between these models is the possibility for a network to have some directed cycles over three or more clusters. In Model 1 and 2, directed cycles are able to exist only in a subgraph consisting of W_1^1 and W_2^1, W_1^2 and W_2^2, or W_1^3 and W_2^3, while a directed cycle can lie over the entire graph containing all the clusters in Model 3.

C. Metrics

The survivability of the given graphs, restructured graphs, randomly reassigned graphs, and the upper bound of the improvement are illustrated in our results. The random reassignments of MAs are conducted with a uniform distribution over all the nodes in the other constituent graph from the constituent graph that includes the source of an MA.

Computing the size of the cycle hitting set is known to be NP-complete even in bipartite graphs, so the exact value cannot be obtained in larger graphs. Our evaluation is conducted using a well-known approximation algorithm whose approximation factor is $\ln |V| + 1$ [24].

Furthermore, the density of a given graph $G = (V, A)$ defined by $\frac{|A|}{\prod_i |V_i|}$ is used to examine the relationship between the survivability improvement, and the maximum and minimum degrees.

D. Results

1) Non-clustered Cases: Figs. 14 and 15 illustrate the survivability of the given and restructured graphs with identical and halved size constituent graphs, respectively. In both cases, our method demonstrates more improvement of the survivability compared to the random reassignment. The survivability of the original graphs $|H(G)|$ maintains a similar value regardless of the size of graphs, though the survivability of the graphs restructured by our method $|H(G')|$ steeply increases along with the size of the graph. Since, in the original graph G, arcs are randomly added, it could be difficult to form larger directed cycles. Therefore, it is reasonable that the number of disjoint
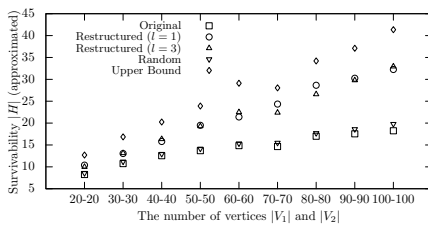
Fig. 14. Survivability of interdependent networks before and after the improvement under $|V_1| = |V_2|$, $\max_{v \in V} \deg_{in}(v) = 4$, and $\min_{v \in V} \deg_{in}(v) = 2$, and $l = 1, 3$.
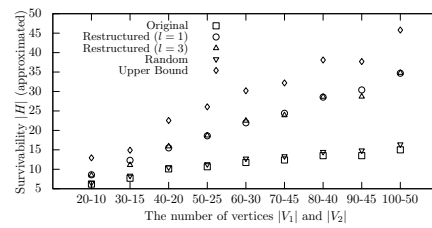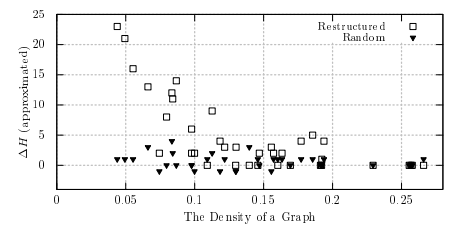

Fig. 15. Survivability of interdependent networks before and after the improvement under $|V_1| = \frac{|V_2|}{2}$, $\max_{v \in V} \deg_{in}(v) = 4$, $\min_{v \in V} \deg_{in}(v) = 2$, and $l = 1, 3$.

Fig. 16. The relationship between graph density and ΔH.
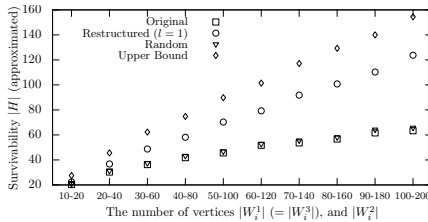

Fig. 17. Survivability of clustered interdependent networks (Model 2) before/after the improvement under $|W_1^1| = |W_1^3| = |W_2^1| = |W_2^3|$, $|W_1^2| = |W_2^2|$, $\max_{v \in V} \deg_{in}(v) = 4$, $\min_{v \in V} \deg_{in}(v) = 2$, and $l = 1$.
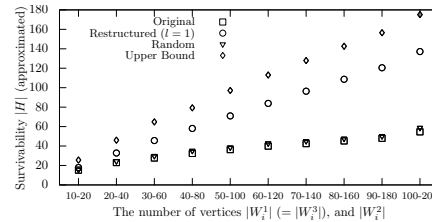

Fig. 18. Survivability of clustered interdependent networks (Model 2) before/after the improvement under $|W_1^1| = |W_1^3| = \frac{|W_2^1|}{2} = \frac{|W_2^3|}{2}$, $|W_1^2| = \frac{|W_2^2|}{2}$, $\max_{v \in V} \deg_{in}(v) = 4$, $\min_{v \in V} \deg_{in}(v) = 2$, and $l = 1$.
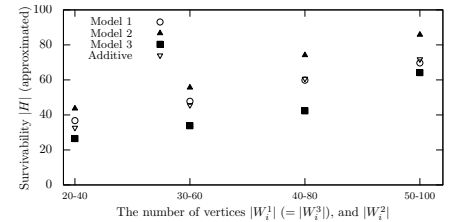

Fig. 19. Comparison of survivability among different dependency models.

cycles indicates the tendency to stay within a similar range of values. On the other hand, there would exist more MAs in larger graphs, because these graphs have more arcs that are not in directed cycles. This results in dramatic enhancement of the survivability in larger graphs. The difference caused by the given maximum hop l for our algorithm remains small over all sizes of a graph.

Fig. 16 indicates the relationship between the density of graphs and ΔH, the amount of survivability improvement. We compare our method to the random reassignment. The result shows that, in graphs with lower density, our method has greater success in increasing the survivability. An observed general trend of our method is the gradual decrease in ΔH in accordance with the density. This trend seems to be induced by the fact that the graphs with more arcs have a higher possibility of composing cycles even in the original topology. This implies that graphs with higher density have fewer MAs that can form new disjoint cycles. On the other hand, the random reassignment does not demonstrate its effectiveness for the improvement in graphs with any density, which is the same result from Figs. 14 and 15. Moreover, the random reassignment sometimes decreases the survivability ($\Delta H < 0$). It is conceivable that the reassignment connects two (or more) cycles and make it possible to decompose all these cycles by the removal of a node. This result implies that imprudent restructuring of the dependency may cause more fragility of the interdependent networks.

2) Clustered Cases: The results in clustered interdependent networks whose dependency relationships follow Model 2 are shown in Figs. 17 and 18. Similar trends to non-clustered cases

are observed for both symmetric and asymmetric cases. The proposed method succeeds in increasing the survivability for different sizes of interdependent networks.

Fig. 19 illustrates the difference in survivability after restructuring among the three types of dependency models of symmetric networks. The value of "Additive" is obtained by the simple addition of non-clustered cases that jointly compose a clustered case. For instance, the case of clustered networks consisting of 20, 40, and 20 nodes clusters is compared with the sum of the survivability of the cases of non-clustered networks of 20, 40, and 20 nodes shown in Fig. 14. The dependency relations among clusters increase from Model 1 to Model 3 (See Fig 13).

Model 1 gives similar survivability to the simple addition of non-clustered cases, since a pair of corresponding clusters in two constituent graphs is independent from the other pairs in this model. In Model 2, the survivability of the entire network increases, because the nodes in cluster W_i^2 can have more supports from the clusters whose cycles are disjoint from the cycles in W_i^2. Although more supports exist among the clusters in Model 3, its survivability is less than the other models. In Model 3, a cycle can lie on more clusters because of the bidirectional dependencies among all the clusters. This topological characteristic is likely to increase the overlapping of multiple cycles and results in the decline of survivability in this model. These results cast a doubt on a naive statement claiming that the increase of dependencies induces more fragility in general interdependent networks.
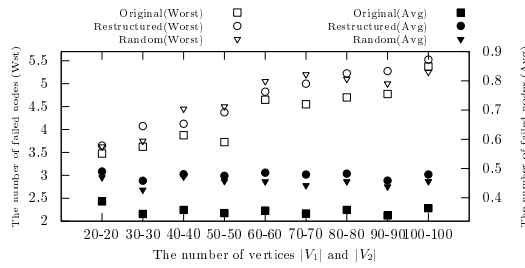
Fig. 20. The number of failed nodes (Worst case and Average) after a single node failure under $|V_1| = |V_2|$, $\max_{v \in V} \deg_{in}(v) = 4$, $\min_{v \in V} \deg_{in}(v) = 2$, and $l = 1$.

VII. DISCUSSION: IMPACT ALLEVIATION VS SURVIVABILITY

Although it is not the primary focus of this paper, in this section, we evaluate the behavior of the proposed algorithms in terms of its effect on the size or impact of a cascading failure. Fig. 20 illustrates the influence of our dependency modifications on the size of cascading failures induced by a single node. In this experiment, the impact of a single node failure at a node v is defined as the number of nodes θ_v that become nonfunctional after a cascading failure initiated by the failure of v. The results are analyzed in terms of the following two metrics:

- *Worst* (non-filled points): the size of the largest cascading failure: $\max_{v \in V} \theta_v$,
- *Average* (filled points): the average size of all possible cascading failures: $\frac{\sum_{v \in V} \theta_v}{|V|}$.

The robustness of restructured networks against a single node failure always declines in comparison with the original topology. The decline in the size of the largest cascading failure is most remarkable in the case of $|V_1| = |V_2| = 50$ in our simulation. In this case, the size of a cascading failure increases by 1 node after the restructuring.

In general, the concentrations of provisioning on a certain portion of a network can improve the survivability, though it can make the other portions more fragile. In contrast, appropriate distributions of provisioning are necessary in order to alleviate the impact of any possible single node failure. This difference in robustness against single node failures and system survivability could be a reason for the decline.

However, when examining the average size of cascading failures, it is observed that the increase in the average number of failed nodes is suppressed within 0.1 nodes over all network sizes. Thus, it could be said that our method does not deteriorate the robustness against single node failures.

VIII. CONCLUSION

This paper addresses the design problem of survivable clustered interdependent networks under some constraints relating to the existence of legacy systems during restructuring. Based on the definition of the survivability proposed in a related work, it is claimed that the increase of disjoint cycles could enhance the survivability. The proposed heuristic algorithm tries to compose new disjoint cycles by gradual relocations of certain dependencies (Marginal Arcs) in order to guarantee

the functionality of existing systems. Our simulations indicate that the algorithm succeeds in increasing the survivability, especially in networks with fewer dependencies. Moreover, the empirical result implies that the number of dependencies, in general, is not the root cause of the vulnerability to cascading failures. Rather, the appropriate additions of dependencies can improve the overall survivability, while poorly designed dependencies make networks more fragile. When redesigning the interdependency between control and functional entities in SDN, NFV, or CPSs based on the proposed algorithm, the possibility to experience catastrophic cascading failures would decrease.

REFERENCES

[1] H. Rastegarfar, D. C. Kilper, M. Glick, and N. Peyghambarian, "Cyber-physical interdependency in dynamic software-defined optical transmission networks," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, pp. 1126–1134, Dec 2015.

[2] J. Liu, Z. Jiang, N. Kato, O. Akashi, and A. Takahara, "Reliability evaluation for NFV deployment of future mobile broadband networks," *IEEE Wireless Communications*, vol. 23, pp. 90–96, June 2016.

[3] M. Ouyang, "Review on modeling and simulation of interdependent critical infrastructure systems," *Reliability Engineering & System Safety*, vol. 121, no. Supplement C, pp. 43 – 60, 2014.

[4] D. H. Shin, D. Qian, and J. Zhang, "Cascading effects in interdependent networks," *IEEE Network*, vol. 28, pp. 82–87, July 2014.

[5] A. Berizzi, "The Italian 2003 blackout," in *IEEE Power Engineering Society General Meeting, 2004.*, pp. 1673–1679 Vol.2, June 2004.

[6] G. Andersson, P. Donalek, R. Farmer, N. Hatziargyriou, I. Kamwa, P. Kundur, N. Martins, J. Paserba, P. Pourbeik, J. Sanchez-Gasca, R. Schulz, A. Stankovic, C. Taylor, and V. Vittal, "Causes of the 2003 major grid blackouts in north america and europe, and recommended means to improve system dynamic performance," *IEEE Transactions on Power Systems*, vol. 20, pp. 1922–1928, Nov 2005.

[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] J. Gao, S. V. Buldyrev, H. E. Stanley, and S. Havlin, "Networks formed from interdependent networks," *Nature physics*, vol. 8, no. 1, pp. 40–48, 2012.

[9] S. Tauch, W. Liu, and R. Pears, "Measuring cascade effects in interdependent networks by using effective graph resistance," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 683–688, April 2015.

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

[11] Y. Zhao and C. Qiao, "Enhancing the robustness of interdependent cyber-physical systems by designing the interdependency relationship," in *2017 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2017.

[12] M. Rahnamay-Naeini, "Designing cascade-resilient interdependent networks by optimum allocation of interdependencies," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–7, Feb 2016.

[13] A. Sturaro, S. Silvestri, M. Conti, and S. K. Das, "Towards a realistic model for failure propagation in interdependent networks," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–7, Feb 2016.

[14] G. Ishigaki, R. Gour, and J. P. Jue, "Improving the survivability of interdependent networks by restructuring dependencies," in *2018 IEEE International Conference on Communications (ICC)*, pp. 1–6, May 2018.

[15] M. Stippinger and J. Kertsz, "Enhancing resilience of interdependent networks by healing," *Physica A: Statistical Mechanics and its Applications*, vol. 416, pp. 481 – 487, 2014.

[16] L. K. Gallos and N. H. Fefferman, "Simple and efficient self-healing strategy for damaged complex networks," *Phys. Rev. E*, vol. 92, p. 052806, Nov 2015.

[17] A. Behfarnia and A. Eslami, "Error correction coding meets cyber-physical systems: Message-passing analysis of self-healing interdependent networks," *IEEE Transactions on Communications*, vol. 65, pp. 2753–2768, July 2017.

[18] M. Pourvali, K. Liang, F. Gu, H. Bai, K. Shaban, S. Khan, and N. Ghani, "Progressive recovery for network virtualization after large-scale disasters," in *2016 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, Feb 2016.

[19] Y. Zhao, M. Pithapur, and C. Qiao, "On progressive recovery in interdependent cyber physical systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, Dec 2016.

[20] A. Majdandzic, L. A. Braunstein, C. Curme, I. Vodenska, S. Levy-Carciente, H. Eugene Stanley, and S. Havlin, "Multiple tipping points and optimal repairing in interacting networks," *Nature Communications*, vol. 7:10850, March 2016.

[21] M. Parandehgheibi and E. Modiano, "Robustness of interdependent networks: The case of communication networks and the power grid," in *2013 IEEE Global Communications Conference (GLOBECOM)*, pp. 2164–2169, Dec 2013.

[22] S. Chattopadhyay, H. Dai, D. Y. Eun, and S. Hosseinalipour, "Designing optimal interlink patterns to maximize robustness of interdependent networks against cascading failures," *IEEE Transactions on Communications*, vol. 65, pp. 3847–3862, Sept 2017.

[23] D. B. Johnson, "Finding all the elementary circuits of a directed graph," *SIAM Journal on Computing*, vol. 4, no. 1, pp. 77–84, 1975.

[24] V. Chvatal, "A greedy heuristic for the set-covering problem," *Mathematics of Operations Research*, vol. 4, no. 3, pp. 233–235, 1979.
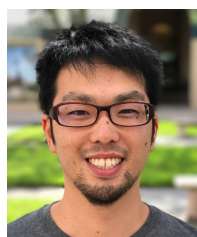
Genya Ishigaki (GS'14) received the B.S. and M.S. degrees in engineering from Soka University, Tokyo, Japan, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree in computer science at Advanced Networks Research Laboratory, The University of Texas at Dallas, Richardson, TX, USA. His current research interests include design and recovery problems of interdependent networks, and software defined networking.

Riti Gour received her BE degree in Electronics and Telecommunication Engineering from S.S.C.E.T, Bhilai, India, in 2012, and her MS degree in Telecommunications Engineering from the University of Texas at Dallas, Texas, in 2015. Since 2015, she has been working towards her Ph.D. degree at UT Dallas, majoring in telecommunications. Her research is focused towards survivability of optical networks against correlated failures and disasters using graph optimization and machine learning techniques.

Jason P. Jue (M'99-SM'04) received the B.S. degree in Electrical Engineering and Computer Science from the University of California, Berkeley in 1990, the M.S. degree in Electrical Engineering from the University of California, Los Angeles in 1991, and the Ph.D. degree in Computer Engineering from the University of California, Davis in 1999. He is currently a Professor in the Department of Computer Science at the University of Texas at Dallas. His current research interests include optical networks and network survivability.