Core Percolation in Coupled Networks

Jiayu Pan^1, Yuhang Yao^2, Luoyi Fu^2, Xinbing Wang2
Department of {Zhiyuan College^1, Computer Science^2,}, Shanghai Jiao Tong University, China

ABSTRACT

Core percolation, as a fundamental structural transition resulted from preserving codes nodes in the network, is crucial in network controllability and robustness. Prior works are mainly based on single, non-interacting network where core nodes are obtained by a classic Greedy Leaf Removal (GLR) procedure that takes off leaf nodes along with their neighbors iteratively. Consequently, it remains unknown how such structural transition may affect realistic complex systems that usually contain multiple interdependent networks.

We take a first look into core percolation in such multi-networks by starting from coupled networks with two fully-interdependent networks. To obtain core nodes in both networks, we propose a new algorithm, called Alternating GLR procedure, that recursively switches between networks in carrying out GLR for node removal. We prove that the proposed algorithm can guarantee the uniqueness in the sense that the final remaining nodes and edges in either of the coupled networks remain the same, and then present analytical solutions for the fraction of core nodes that will be ultimately left when the algorithm terminates. Along with further empirical observations, surprisingly, we find that opposite to the single network that can always guarantee core emergence after GLR given that the network is sufficiently dense, core can possibly fail to emerge in coupled networks due to the cascading failure caused by node removal interdependence between networks. Also, the presence of core exhibits a jump at the critical point as a first order transition in coupled networks while it undergoes a continuous second order one in a single network.

1 INTRODUCTION

During the past decade, structural transition in complex networks has received considerable attention, and was manifested its crucial role in many network properties such as robustness and resilience to breakdowns [1][2], epidemic and information spreading on social networks [3][4], social influence propagation, and the fragility caused by multilayer connections [5]. One type of structural transition belongs to core percolation. It technically refers to the process of obtaining core, i.e, a spanned subgraph remained in the network after a classic greedy leaf removal procedure that removes leaves (nodes of degree one) and its neighbor iteratively, along with the removal of all their links. As revealed by recent work, the robustness of network controllability relies heavily on the presence of a core in the network [7]. While core percolation has been intensively studied in single, non-interacting networks [6–10], it is still not entirely understood how it may affect realistic complex systems, which are usually composed of multiple interdependent networks.

Such network structure is also called network of networks (NoNs) [11][12][13]. Different from a single network, NoNs describe the connections and interdependence among multiple networks [14]. In other words, the functioning of nodes in one network is dependent upon the functioning of nodes in other networks of the system. Such characteristic, on one hand, can easily lead to the cascading failure where a failure of a very small fraction of nodes in one network may lead to the complete fragmentation of a system of several interdependent networks. A typical example belongs the dramatic real-world cascade of failures, i.e., the electrical blackout that affected much of Italy on 28 September 2003: the shutdown of power stations directly led to the failure of nodes in the Internet communication network, which in turn caused further breakdown of power stations [15]. On the other hand, when effectively utilized, multi-interdependence between networks can also bring about performance gain. Take information transmission in conjoined networks [16] for instance. It suggests that the fraction of individuals who receive an item of information is shown to be significantly larger in the conjoint social-physical network case (multiple networks case), as compared to the case where the networks are disjoint (single network case). The above mentioned phenomenons, leading to either performance degradation or improvement, are closely related to core percolation that characterizes how local change of the structure may eventually incur a global network structural transition. However, existing work regarding NoNs is mainly based on the analysis of giant component emergence [17][18], whereas core percolation, which is equally important as noted earlier, has not yet been investigated in NoNs.

We are thus motivated to take a first attempt to study core percolation in NoNs. Different from a single network, the problem is made far more complicated in NoNs in the sense that the existence of intricate multi-interdependence among networks renders it difficult to mathematically characterize the node removing process. To get a tractable result, we approach the problem by starting from a special case of NoNs, i.e., coupled networks that contain two fully interdependent networks, indicating that each node in one network can always find its counterpart in the other network. The result is extendable to more general forms of NoNs. Recall that a core can be obtained in a single network through conventional GLR procedure in an iterative fashion, as stated earlier. Here we partially borrow this idea and propose a new node removal algorithm that incorporates the mutual influence between networks. Named as Alternating GRL Procedure, the main idea of the algorithm is to firstly choose a network and executes in that specific network the GLR procedure, which can be reproduced in the following way: A node with degree one (leaf) is randomly chosen, and removed along with all its neighbors as well as the links in between. Nodes that becomes isolated are also removed. The procedure is repeated until there remain no leaf nodes. The algorithm then switches to the other network where it continues the above GLR process, from which the removal of nodes will further cause some nodes to be removed in the first network. Such procedure repeats in the prescribed alternating manner till no nodes can be removed from the whole system. And the final remaining nodes in coupled networks are thus called core nodes. Additionally, several categories of removable nodes, depending on how they are removed during the algorithm, are also introduced.

While we defer definitions of each node category in Section 2.3, we prove that, the proposed algorithm can guarantee the uniqueness in the sense that the final remaining nodes and edges remain the same in terms of node category in both of the networks, as a fundamental validation of the system. Following that, we proceed to analyze the final fraction of core nodes as well as the critical condition under which core appearance occurs, as a major characteristic of core percolation when the number of nodes goes to infinity. This, interpreted from a more technical perspective, is to establish iterative equations that captures the dynamic node removal process following an alternative fashion in coupled networks. In addition to the fraction of core nodes in each iterative, we also need to incorporate the node degree distribution change that affects the calculation at the next iteration. More precisely speaking, at each iteration we need to switch between networks for updating both the fractions of core nodes and nodal degree distributions. Due to the difficulty in derivation of the exact solution of node degree distribution, we turn to provide the corresponding approximations.

We disclose from our derivations, along with empirical results of real datasets, that due to the cascading failure, core nodes start to appear at a larger mean degree in coupled networks than that in a single network. Also, we find that the fraction of final remaining nodes undergoes a jump at the critical value as a first order phase transition, whereas in single network it undergoes a continuous second order transition. The reason behind is that in coupled networks, for both of the networks around a critical state, it seems that the nodal degree distribution is close to a power law, thus adding only a few extra nodes with degree one will cause the whole system to collapse. In addition, we even obtain an interesting experimental observation that if the two networks are partially interdependent instead of being fully interdependent, there will be equal fractions of remaining nodes in both networks as long as both of them have the same degree distributions, regardless of the strength of interdependence between two networks.

The algorithm for core nodes acquisition is efficiently implementable, having a complexity of no more than $O(N^{\frac{3}{2}})$, with N representing the total number of nodes in each network. The complexity stems mainly from the renew of nodal degree distribution in each iteration, and can be potentially reduced through only renewing the node degree distribution that has higher probabilities. Also, we provide some further understanding of previously investigated percolation issue in coupled networks.

The rest of the paper is organized as follows. Section 2 introduces network model and Section 3 proves the uniqueness of the core percolation in coupled networks. In Section 4, we establish iterative equations of the dynamic process of node removal and empirically verify our theoretical findings in Section 6. Section 7 concludes the paper.

2 MODELS AND DEFINITIONS

2.1 Coupled Networks

A general interdependent system consists of multiple networks, each of which, unlike a traditional single network, establishes connections with some of the other networks in the system. Each connection is assumed to be in a random mode and each node in a generic network i is supposed to have equal probability q_{ij} to establish connection with some node in network j. In the present work, we mainly discuss coupled networks that are composed of two interdependent networks called networks A and B with equal number of nodes N. Specifically, we assume that the dependence coefficient $q_{AB} = q_{BA} = q = 1$, meaning that the two networks are f*ully interdependent*. The mutual influence in coupled networks is revealed in the way that if a node i in network A depends on a node j in network B, then the removal of j will also lead to the removal of node i. Although coupled networks serve as a special case of interdependent networks, the derived results can provide useful guidelines for future extension to multi-interdependent networks.

This paper mainly discusses the *non-feedback condition*, whose definition includes three rules: (1) Suppose nodes j, k are both in network B. If node i depends on both j and k, then $j = k$; (2) Suppose i and l are both in network A. If both i and l depend on j, then $i = l$. (3) Suppose i depends on k, k depends on l, then $i = l$.

2.2 A Node Removal Procedure in Coupled Networks

Based on the model stated above, we now introduce the node removal procedure in such networks. Recall that in a single network, nodes can be removed through the well-known Greedy Leaf Removal (GLR) algorithm where leaf node, i.e., node with degree one is randomly chosen and removed along with its neighbors, followed by the removal of resulted isolated nodes. The procedure is repeated until no leaf nodes are left. However, coupled networks differ from a single network in that the effect of connection between networks cannot be overlooked. Based on that, we introduce below a new node removal procedure in coupled networks. Specifically, we name it **Alternating GLR Procedure**, which contains three major steps:

(1) Node removal in network A: Randomly choose a leaf node in A, and remove this node along with its neighbors. The operation continues until no leaves exist in network A. Isolated nodes in A will then also be removed. This procedure resembles the traditional GLR in a single network [10].

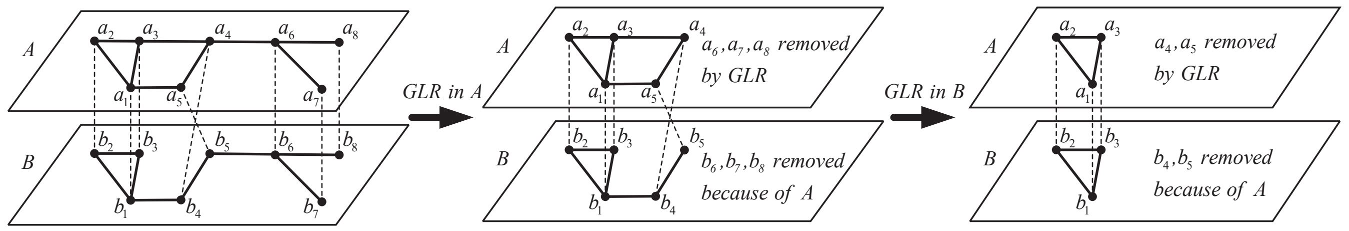

A
B
GLR in A
GLR in B
a_6, a_7, a_8 removed by GLR
b_6, b_7, b_8 removed because of A
a_4, a_5 removed by GLR
b_4, b_5 removed because of A

Figure 1: Overview of Alternating GLR Procedure in coupled networks. Each node is dependent on a different node in the other network. Specifically, $\forall i \in [1 \dots 8]$, node a_i depends on node b_i and in return b_i depends on a_i. Finally nodes $a_1 - a_3$ and $b_1 - b_3$ become core nodes in networks A and B, respectively. All of the nodes, i.e., removed nodes and core nodes in A, B are well-defined. Moreover, nodes a_7, a_8 and b_7, b_8 are α removable nodes, a_4, nodes a_6, b_4, b_6 are β nodes and nodes a_5, b_5 are γ nodes.

The fraction of remaining nodes in A is denoted as μ_1. The difference here is that the removed nodes in A will cause the removal of the corresponding (dependent) nodes in network B. As illustrated in Fig. 1, we choose network A and operate GLR that causes nodes $a_6 - a_8$ to be removed. Consequently, nodes $b_6 - b_8$ in network B are also removed.

(2) Further node removal in network B: Upon completion of the above GLR in A, the procedure turns to Network B where traditional GLR is further adopted for node removal. Recall that during GLR in A, some nodes in B have already been removed due to the interdependency between A and B. Similarly, the fraction of remaining nodes in B is denoted as μ_2, and the removed nodes in B also cause the further removal of the corresponding nodes in A. Take Fig. 1 as example. We can see that the removal of nodes b_4, b_5 further leads to the removal of nodes a_4, a_5.

(3) Alternative Repetition of steps 1 and 2: The above two steps are iteratively repeated in the prescribed alternative manner until no nodes can be removed in both networks. During the iteration, the fraction of remaining nodes μ_3, μ_4...can be obtained. Specifically, till the $n+1^{th}$ iteration, in network A, we have obtained μ_1, μ_3, μ_5,. . .,μ_{2n+1} with odd subscripts while for network B, we have obtained μ_2, μ_4, μ_6,. . .,μ_{2n+2} with even subscripts. We denote by $\mu_{\infty,1}$ and $\mu_{\infty,2}$ respectively the fraction of final remaining nodes in networks A and B. Since the system is fully interdependent, it is apparent that $\mu_{\infty,1} = \mu_{\infty,2} = \mu_\infty$.

The overview of the whole Alternating GLR Procedure is also illustrated in Fig. 1. In consistency with previous works, here we also introduce the conception of **core percolation**, defined as the phenomenon of structural transition resulted from the algorithm when the number of nodes goes to infinity, which is our major concern of this work.

2.3 Categorization of Node Types

To systematically study core percolation on coupled networks, we classify the nodes into two major categories according to how they can be removed during the above Alternating GLR procedure:

1. **Core nodes**: Analogous to a single network where the remaining nodes are called *core nodes* [10], in coupled networks, the remaining nodes after Alternating GLR are also called core nodes. In Fig. 1, nodes $a_1 - a_3$ and $b_1 - b_3$ are core nodes in networks A and B, respectively.

2. **Removable nodes**: similar to a single network [10], the removable nodes can be further classified into three types: α *removable nodes* which can be isolated without directly removing themselves, β *removable nodes* which can be the neighbor of a leaf node and γ *nodes* which are removable but meanwhile neither α nor β removable.

Furthermore, as we will also demonstrate in Section 3, as long as the node removal follows the proposed algorithm, both the remaining nodes and edges, belonging to either core nodes or three types of removable nodes in networks A and B, are *well-defined*, a phenomenon that is alternatively called *uniqueness* indicating that the remaining nodes and edges remain the same in terms of node type under different removing sequences which all follow the common framework of the proposed algorithm.

The interplay among nodes of different types can be interpreted in a more detailed way. According to the behaviors of Alternating GLR algorithm, when it is executed in network A, the node removal process resembles the traditional GLR in a single network. According to the algorithm, removed nodes will cause corresponding nodes in network B to be removed. To remain the aforementioned uniqueness, the corresponding removed nodes in network B are classified as the same with the dependent nodes in network A. As exemplified in Fig. 1, the definitions of nodes $b_6 - b_8$ are the same with their dependent nodes. As a result, nodes b_7, b_8 are α nodes and node b_6 is β node, regardless of their roles in network B. Similarly, when GLR is operated in network B, the corresponding nodes in network A are classified as the same with dependent nodes in B. For instance, if node a_1 in A is removed and classified as α removable, and node b_2 in network B depends on node a_1, then b_2 is also α removable.

Table 1 lists the main notations that will be used throughput the paper.

Notation	Definition
$\mathcal{G}$	an undirected graph
$P(k)$	probability of a node with degree k
$G(z)$	generating function of a graph
μ	fraction of core nodes in one iteration
q_{ij}	the number of nodes in network j that depend on network i divided by total number of nodes in network j
c	mean degree of nodes
$g(G(z))$	fraction of core nodes of a graph with $G(z)$
$f(G(z))$	the generating function after operating GLR in a graph with $G(z)$

Table 1: Notions and Definitions

3 UNIQUENESS OF CORE PERCOLATION

In this section, we prove that when the algorithm terminates, each type of nodes (See Section 2.3) are well-defined in coupled networks.

3.1 Reasonability of the Algorithm

A major characteristic of our algorithm is that before it switches to the other network (without loss of generality, here we can say network B), it will continue the execution of GLR in the original network (say network A) until there are no more nodes that can be removed from that network. As a seemingly feasible alternative, this constraint can be loosened by letting the algorithm switch to network B for a new GLR execution before it finishes the original GLR procedure in network A. However, as we will illustrate in the example below, such change will not guarantee the uniqueness.

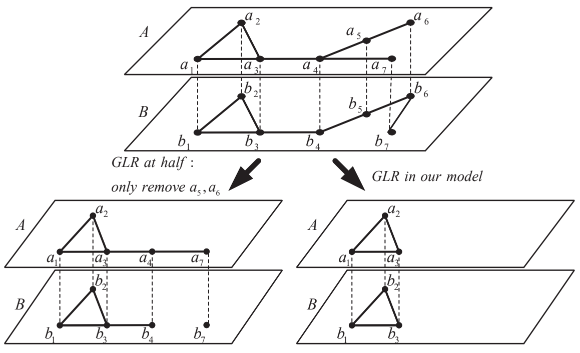


Figure 2: Difference between using GLR process to the end and using GLR process at half before applying it to another network. According to Alternating GLR procedure, the core nodes are shown on the right lower side of the figure. If we loosen our restriction, the condition will be different, and the uniqueness will not be guaranteed.

We take Fig. 2 for example to explain why the uniqueness cannot be guaranteed in such case. In the figure, $\forall i \in [1 \ldots 7]$, node a_i depends on node b_i and in return b_i depends on a_i. The nodes on the right lower figure are remaining nodes according to our algorithm. However, if we remove only a part of the removable nodes in network A and then switch to network B for the new GLR, the final remaining nodes in the two networks will be different from those obtained previously. The left lower part of the figure exemplifies such possibility. Here the algorithm only removes nodes a_5, a_6 from network A, and then turns to network B to start a new GLR procedure. As a result, node b_3 is removed since it is a neighbor of the leaf node b_4. In the end, all of the nodes in the system will be removed, leading to a total cascade failure. This outcome is quite different from the former one where there are same remaining nodes and edges left in coupled network. This implies that it is reasonable to stick to one network to finish the whole GLR process before it switches to the other one for a new GLR. In addition, it is worthwhile noting that the choice of A or B for GLR operation at first will lead to different remaining nodes in two networks, even if μ_∞ for the two networks will be the same.

3.2 Uniqueness of Core Percolation

With the clarification of the reasonability of the algorithm, we are now ready to prove the uniqueness property. Since we are considering core percolation in coupled networks, proving the uniqueness in coupled networks is equivalent to proving that it is unique in either of the networks. Here the proof in either network of the coupled networks also includes two parts, i.e., to prove that both the core nodes and removable nodes obtained under our algorithm are unique. The following theorem states a uniqueness guarantee in either of the network in the coupled ones.

THEOREM 3.1. *Regardless of the different sequences of GLR node removal procedures, the remaining nodes and edges remain the same.*

PROOF. Without loss of generality, we focus on network A in coupled networks. The proof is divided into two steps: (1) proving the uniqueness of the core nodes, and (2) proving the uniqueness of removable nodes, as stated above. Now we focus on the first step as follows:

1. The uniqueness of core nodes: According to the behavior of Alternating GLR procedure, it adopts traditional GLR when it is in the progress of node removal in one specific network. Now suppose in network A, there exist two different GLR sequences called GLR_1 and GLR_2, where a node v becomes a core node after the execution of GLR_1, but meanwhile not a core node resulted from GLR_2. Then, proving Theorem 1 is equivalent to proving that such kind of node v does not exist.

First, we take a look at GLR_2, where we know that node v is removed after some steps due to one of the three following conditions:

(i) v is a leaf, thus it will be removed at the next step;
(ii) v is a neighbor of a leaf;
(iii) v becomes isolated.

Now we prove that v cannot be a core node in GLR_1 in any of the three conditions.

(i) If v is a leaf, the neighbor nodes of v must be removed since they are also the neighbors of other leaves. We denote one of v's neighbors as n_1, and one of n_1's another neighbors as v_1, which is also a leaf that can link to other nodes similar to itself. Following the same manner, we can find a series of nodes denoted as n_2, v_2, ... located on a path. Notice that v and n are staggered. If the path, after some steps, finds that its point belongs to this path, then this path will step back one point and find another link to go along. In this way, every bifurcation can stretch and end up to a leaf without incurring a cycle. If one of v's links cannot find a non-cycled path, then this cycled path will not be removed under any circumstances. Hence we can find such kind of non-cycled path, the end of which is a leaf.

Now we take a look back into GLR_1, where v is a core node and may undergo many bifurcations before. Since v

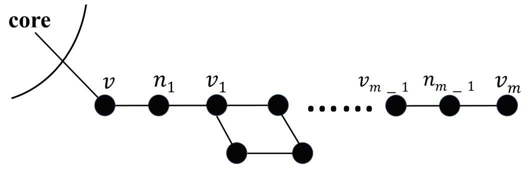


Figure 3: Every bifurcation could stretch and end up to a leaf without a cycle. If we want to protect v, we must protect one v_1, leaf neighbor of n_1. For bounded number m, v, n_1, v_1, n_2, v_2 .... n_m and v_m should be all protected. But v_m is an original leaf, it cannot be protected, contradictory.

is removed in GLR_2, at least one link should be protected. Fig. 3 provides an illustration of such a link. If v was protected, because of n_1 it is impossible that n_1 is a leaf in any GLR methods[1]. Thus v_1 would be protected. And so would an arbitrary node v_m (m is a bounded number), which is a leaf in this graph. In other words, there could be a path (not cycled) that consists of bounded number of nodes $\{v, n_1, v_1, n_2, v_2, \ldots, n_m\}$ and ends up to a leaf v_m. However, this is impossible since no GLR method can protect an original leaf from being removed.

(ii) If v is a neighbor of a leaf, we find a leaf which is neighbor of v. This condition is therefore equivalent to the first condition.

(iii) If v is isolated, it must be a leaf during the last iteration since a point used to be neighbor of v is removed. Again, this is equivalent to the first condition.

2. The uniqueness of removable nodes: Having proved the uniqueness of the core nodes, we proceed to prove the uniqueness of the previously defined three types of removable nodes as follows:

(i) α removal: Given a node, as long as we can find an approach to isolate it, it will be α removable. Apparently α nodes are independent of the removing sequence.

(ii) β removal: Given a node, as long as we can find an approach to discover that it is a neighbor of a leaf, it will be β node. Apparently β nodes are independent of removing sequence.

(iii) γ removal: Since we have reasoned that α, β nodes and core nodes are independent of GLR removing sequence, the remaining nodes, γ nodes are apparently unique.

(iv) there does not exist a node that is both β and α removable. We prove this by considering in the reverse way: Suppose that a node v is both α and β removable. Then α node can be isolated. Thus all neighbors of v are β nodes in G. In addition, each of them must have another leaf, and must be removed after a leaf is removed. However, it is impossible to have a path that starts from an α node, contains a series of intermediate leaf nodes and ends with a leaf. Hence none of v's neighbor nodes are leaves or leaves to be. In consequence, node v cannot be a β node.

Now, we complete the whole proof by showing the uniqueness from both aspects above. □

[1]We did not prove this theorem, but we have not found any counterexamples

Upon completion of proving the uniqueness of either network, the following theorem further states the uniqueness of the coupled network.

THEOREM 3.2. *In coupled networks, if the Alternating GLR algorithm guarantees the uniqueness of either network, then it can also guarantee the uniqueness of the whole coupled networks.*

PROOF. Recall again that in the Alternating GLR algorithm, originally we choose network A for GLR operation. According to Theorem 1, we have that the core percolation in network A is unique. the uniqueness of the removed nodes in A will also lead to the uniqueness of the removed nodes in B. According to the behaviors of our algorithm, the formation of the other removed nodes in B are unique, so the whole remaining nodes in B are also unique in that iteration. Similarly, the formation of the removable nodes which will influence A is unique. Such inter-network influence continues in the prescribed alternating way during iterations, where the uniqueness can always be guaranteed in both networks. Ultimately, when the algorithm terminates, the core nodes, and α, β, γ nodes left in coupled networks are unique. □

4 DERIVATION OF THE DYNAMIC PROCESS OF ALGORITHM

Upon the proof of uniqueness of the proposed algorithm, in this section we start to derive the iteration equations of the Alternating GLR algorithm to disclose the interplay of node removal between networks during different iterations. Recall that in step 3 of the algorithm (see Section 2.2), we define $\mu_1, \mu_3 \ldots \mu_{2n+1}$ as the fractions of survived nodes in network A in the n^{th} iteration, and $\mu_2, \mu_4 \ldots \mu_{2n+2}$ as the fraction of survived nodes in network B in the n^{th} iteration.

Suppose that in either network A or B, all nodes are randomly assigned a degree k from a probability distribution denoted as $P(k)$. Then the generating function of the degree distribution is expressed as $G(z) = \sum_{k=0}^{\infty} P(k)z^k$, where z is an arbitrary complex variable. Hence, after random removal of $1-p$ fraction of nodes ($0 \leqslant p \leqslant 1$), the generating function of the remaining nodes will be $G(1-p(1-z))$ [20]. Equivalently, for a certain degree m, if the original probability is $P(m)$, the probability that the remaining nodes are of degree m will be

$$P'(m) = \sum_{k=m}^{\infty} P(k) \binom{k}{m} p^m (1-p)^{k-m}. \qquad (1)$$

In Alternating GLR procedure, since networks A and B initially possess the same degree distribution, they exhibit the same generating function $G(z)$ of degree distributions.

4.1 Iterative Equations

According to the algorithm, apparently $\mu_1 = g(G(z))$. Thus, the generating function of network A will be $G_1(z) = f(G(z))$.

Then in network B, since $1-\mu_1$ fraction of the nodes are randomly removed, the generating function will be $G(1 - \mu_1(1-z))$. After further executing GLR algorithm in network B, we have $\mu_2 = \mu_1 g(G(1-\mu_1(1-z)))$. The generating function will then yield to $G_2 = f(G(1-\mu_1(1-z)))$.

With such iterations repeating in the above prescribed way, up to the $(n-1)^{th}$ iterations we have obtained μ_{2n-1}, μ_{2n} as well as the generating functions of both networks A and B, i.e., G_{2n-1} and G_{2n}. In the n^{th} iterations, for network A, it should firstly randomly remove a $(\mu_{2n-1} - \mu_{2n})$ fraction of nodes. Then the relative remaining fraction becomes $\frac{\mu_{2n}}{\mu_{2n-1}}$, and the generating function thus becomes $G_{2n-1}\left(1 - \frac{\mu_{2n}}{\mu_{2n-1}}(1-z)\right)$. After the GLR process, the equation can be expressed as

$$\mu_{2n+1} = \mu_{2n} g\left[G_{2n-1}\left(1 - \frac{\mu_{2n}}{\mu_{2n-1}}(1-z)\right)\right]. \qquad (2)$$

Similarly, in the iteration, a fraction $(1 - \mu_{2n+1}/\mu_{2n})$ of the remaining nodes in network B should be removed. The generating function will be $G_{2n}\left(1 - \frac{\mu_{2n+1}}{\mu_{2n}}(1-z)\right)$. After the execution of GLR in network B, we have

$$\mu_{2n+2} = \mu_{2n+1} g\left[G_{2n}\left(1 - \frac{\mu_{2n+1}}{\mu_{2n}}(1-z)\right)\right]. \qquad (3)$$

In order to maintain the iteration, the generating functions should be updated from G_{2n-1}, G_{2n} to G_{2n+1}, G_{2n+2}, respectively.

In each iteration, both the generating functions in networks A and B will undergo two changes. The first change results from random node removal while the second one is incurred by GLR process. Based on the definition, the new iteration equations are

$$G_{2n+1}(z) = f\left[G_{2n-1}\left(1 - \frac{\mu_{2n}}{\mu_{2n-1}}(1-z)\right)\right] \qquad (4)$$

and

$$G_{2n+2}(z) = f\left[G_{2n}\left(1 - \frac{\mu_{2n+1}}{\mu_{2n}}(1-z)\right)\right]. \qquad (5)$$

Here the four parameters $\mu_{2n+1}, \mu_{2n+2}, G_{2n+1}, G_{2n+2}$ will keep being updated until the algorithm converges. And when n is sufficiently large, we have that $\mu_{2n+1} = \mu_{2n+2} = \mu_\infty$. Since a monotonous and bounded sequence must converge to a unique number, there exists only one μ_∞. Therefore, obviously we have

$$\lim_{n\to\infty} G_{2n-1}\left(1 - \frac{\mu_{2n}}{\mu_{2n-1}}(1-z)\right) = \lim_{n\to\infty} G_{2n-1}(z), \qquad (6)$$

and

$$\lim_{n\to\infty} G_{2n}\left(1 - \frac{\mu_{2n+1}}{\mu_{2n}}(1-z)\right) = \lim_{n\to\infty} G_{2n}(z). \qquad (7)$$

For the ease of understanding, Fig. 4 shows the overall diagram of our derivation. Initially, both networks have $\mu_0 = 1$ fraction of nodes and μ_0 becomes μ_1 due to GLR. Next, removing a fraction $\mu_0 - \mu_1$ of nodes in network A will cause some nodes in network B to be removed. This, interpreted technically, is that the generating function (probability distribution) G_1 influences G_0 in network B and changes it into G_0'. After GLR operation in network B, G_0' further becomes G_2 and fraction of core nodes yields to μ_2. Then, G_2 changes G_1 to G_1' which further becomes G_3 after GLR. Following this route, in the n^{th} iteration, G_{2n-1} is influenced by G_{2n} to G_{2n-1}', after GLR, it becomes G_{2n+1}, based on which μ_{2n+1} can be calculated. In analogy, in network B,

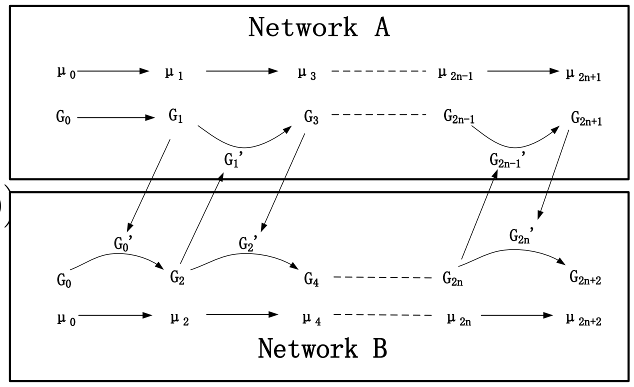


Figure 4: Diagram of the derivation of our model.

G_{2n} becomes G_{2n+2} and μ_{2n+2} can also be calculated. In summary, in each iteration, we should update μ_{2n+1}, μ_{2n+2} as well as the degree distributions of networks A and B.

Remark 1. *The proof in previous section indicates that both networks A and B possess the same remaining core nodes. Actually, it is worthwhile noting that in non-feedback condition considered in the present work, even if we relax the assumption of full interdependence between networks (remove the constraint of $q = 1$), the number of core nodes for both networks will always remain the same as long as the initial degree distributions are the same in networks A and B. This conclusion, on the first sight, does not appear to be apparent since it is supposed in the algorithm that network A is chosen first for GLR operation, and either choice of A or B will destroy the symmetry of the system, possibly for the non-connecting part, i.e., $1 - q$ of both networks, leading to the different remaining core nodes. Notice that if $q = 1$, which is exactly the case we analyze in the paper, the result is demonstrated to be true; if $q = 0$, the network is reduced to the traditional single network, where the result is obviously true again; For other cases in between, suppose we introduce a slight increase of q. Then it seems that whether network A or B loses more nodes is irrelevant to q. Therefore, the difference of the number of core nodes will increase with as q decreases. However, the difference is still zero when $q = 0$, verifying the conclusion that the core nodes always remain the same in networks A and B.*

4.2 Solutions of Iterative Equations

After establishing the iterative equations, now we need to seek for the solutions of the two functions, i.e., $g(G(z))$ and $f(G(z))$ in the equations.

4.2.1 Solution of Fraction of Core Nodes $g(G(z))$. As defined earlier, $g(G(z))$ represents the faction of remaining core nodes in each of the coupled networks when the algorithm terminates. Inspired from prior work of traditional single network [21], we obtain the solution of $g(G(z))$ with the main idea presented below:

Let α' (or β') denote the probability that a random neighbor of a random node i in network $\mathcal{G}$ is α removable (or β

removable) in network $\mathcal{G}/i$ where node i and all of its links are removed. In particular, node i is α removable in $\mathcal{G}$ if all its neighbor nodes are β removable in $\mathcal{G}/i$ and β removable in $\mathcal{G}$ if at least one neighbor is α removable in $\mathcal{G}/i$. Based on that, we have

$$\alpha' = H(1-\beta'), \qquad 1-\beta' = H(\alpha'), \tag{8}$$

where $H(x) = \frac{1}{c}\sum_{k=0}^{\infty}(k+1)P(k+1)(1-x)^k$. Defining $f(x) = H(H(x)) - x$, we have that α' is the root of $f(x) = 0$. In the iterative mode, the initial point is $x = 0$, and according to the shape of $f(x)$, the iteration will converge to its smallest fixed point. We want the smallest root of $f(x) = 0$. After solving α', $\beta' = 1 - A(\alpha')$, the solution of the fraction of core nodes n_{core} in each of the coupled network is

$$n_{\text{core}} = \sum_{k=0}^{\infty} P(k) \sum_{s=2}^{k} \binom{k}{s} (\beta')^{k-s} (\alpha')^{s}. \tag{9}$$

According to binomial theorem,

$$n_{\text{core}} = G(1-\alpha') - G(\beta') - c(1-\alpha'-\beta')\alpha'. \tag{10}$$

We need to be aware that here α', β' are not the same as the fractions of α, β removable nodes in a traditional single network. The purpose of introducing α', β' is to solve n_{core} conveniently. And the the number of α and β nodes, i.e., n_α and n_β, can also be solved by α' and β'.

4.2.2 Solution of Updated Generating Function $f(G(z))$. With the solution of $G(g(z))$, we turn to seek for the solution of updated generating function $f(G(z))$ of the number of core nodes. A typical previous methodology to get the solution [22] is using iteration function, where, however, iterations are required for every step of leaf node removal, and consequently makes it far more complicated to establish the equations for every step and relatively difficult to compute the process. In addition, since it is impossible to get derivations for many different kinds of graphs, we get around the difficulty by approximating a generating function rather than solving it exactly. In other words, we aim to find an approximate degree distribution of $f(G(z))$.

According to the algorithm, after GLR operation, nodes with degrees 1 and 0 are removed, leading to the removal of those with higher degrees in the next iteration. Obviously, the number of nodes with degree 2 will incur the most changes. Therefore, our approximation method is to delete $P(0), P(1)$ and add some part of $P(2)$. Moreover, we know that if $P(1)$ is very small, meaning that very few number of nodes are removed, then it will not incur too much change to the degree distribution. Under such circumstance, it can be seen that the change of degree distribution relies heavily on $P(2)$. Taking into consideration of those conditions, now we introduce an index qq satisfying $qq \geqslant 1$. And for a network with generating function $G(z)$, after GLR, the updated degree distribution, denoted as $G'(z)$, should be

$$G'(z) = \frac{1}{a}\left(\sum_{k=3}^{\infty} P(k)z^k + P(2)(1+qq\times P(1))z^2\right), \tag{11}$$

where the normalization number a satisfies

$$a = \sum_{k=3}^{\infty} P(k) + P(2)(1+qq\times P(1)). \tag{12}$$

In short, the only change occurs to $P(2)$, which becomes $P(2)(1+qq\times P(1))$. To validate this phenomenon, we graphically present it through simulations from two real Arxiv data sets, i.e., Arxiv HEP-TH (High Energy Physics - Theory) collaboration network (with # of nodes and edges being 5000 and 12000), and Arxiv GR-QC (General Relativity and Quantum Cosmology) (with # of nodes and edges being 25000 and 100000), where the proposed algorithm is operated. Specifically, we set $qq = 5$ and plot the observation of degree distribution change in Fig. 5, from which it can be seen that $P(2)$ exhibits an apparent change before and after GLR, whereas the changes of $P(i)$, $i = 3, 4, \ldots$ are negligible.

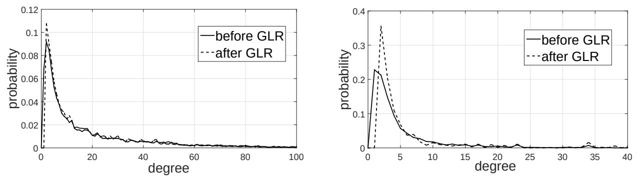


Figure 5: Validation for derivation of generating function change. (a) is the degree distribution of Arxiv HEP-TH (High Energy Physics - Theory) collaboration network, with # of nodes and edges being 5000 and 12000. (b) is the Arxiv GR-QC (General Relativity and Quantum Cosmology) collaboration network, with # of nodes and edges being 25000 and 100000. The active line is the original degree distribution and the dotted line is the distribution after GLR. After GLR, we find that $P(2)$ increases the most.

REMARK 2. *There could be some deviations between theoretical derivations and simulation values. The deviations stem mainly from two aspects: (i) The first one is that in our derivations we assume that whether a graph removes node i or removes i along with i's neighbor only incurs slight influence on the fraction of core nodes. Although this holds in most of the realistic networks, there can still be some exceptions, e.g., a realistic web network that contains many users with very low and very high degrees. Consequently, it is possible to choose a node with very high degree and the fraction will greatly change; (ii) the second aspect is the change of degree distributions after GLR operation. Although in Equations (11) and (12) only* $P(2)$ *is updated due to introduction of qq, it is not convincing to conclude that* $P(3), P(4), \ldots$ *change the same proportion. This minor probability error also causes some deviations.*

The proposed algorithm is efficiently implementable, with a complexity of no more than $O(N^{3/2})$. The complexity is mainly originated from calculating the change of node degree distributions, as technically characterized by Equations (1) to (10). Due to space limitations, we defer a more detailed analysis in **Appendix A in our technical report** [32].

Also, recall that in our analysis of core percolation of coupled networks, we need to update both the fractions of core nodes and nodal degree distributions. As a counterpart of core percolation, percolation theory has also been adopted to study the vulnerability of coupled networks in previous literature [11], where, however, only the fraction of nodes in giant component are updated at each iteration. As a result, it is interesting to have an extra exploration of why the degree distribution does not need to be renewed in that case. We defer the details to **Appendix B in our technical report** [32] again due to space limitations.

5 EXPERIMENTS

In this section, we verify our theoretical findings through experiments on different real data sets. Specifically, our experiments include two parts: (i) observing n_{core} in both single network and coupled networks, and (ii) observing the fractions of remaining nodes μ_∞ in coupled networks.

5.1 Observation of Core Percolation

5.1.1 Core Percolation in Single Network. To observe the phenomenon of core percolation in a single network, we conduct experiments on two real data sets listed as follows. Specifically, we get empirical results by adopting classic GLR, as noted in Section 2.2, in single networks. In contrast, the theoretical results are derived by Equations (8), (9) and (10).

Citation Networks [23]: We extract 10 networks of different subfields in Citation Networks ranging from 31599 nodes (75215 edges) to 1,062,076 nodes (2,951,981 edges), where each node represents a paper and an edge between two nodes means that there is a citation in between (here we do not distinguish the cases of mutual citation and unilateral citation). Fig. 6(a) plots different results of core percolation after GLR operation on those citation networks. It can be seen that in each network, there are more than 40% of papers (nodes) that are cited only once (the theoretical curve). According to behaviors of GLR, more than 80% of the papers are eventually removed (the experimental curve). And for those papers that have multiple citations, we delete all of the leaf nodes and their links twice in the graph.

Gnutella Peer-to-Peer Networks [24] and a **Patent Citation Network** [25]: We extract 6 networks of different subfields in Gnutella Peer-to-Peer Networks ranging from 6,301 nodes (20,777 edges) to 62,586 nodes (147,892 edges) as well as a Patent Citation Network. The results are plotted in Fig. 6(b). Note that the process of GLR in traditional single network is equivalent to the first step of GLR in our Alternating GLR algorithm, we denote μ_1 to be fraction of core nodes remained in a single network.

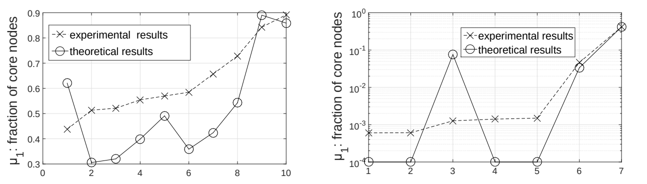
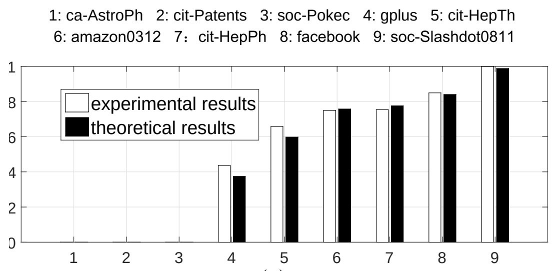
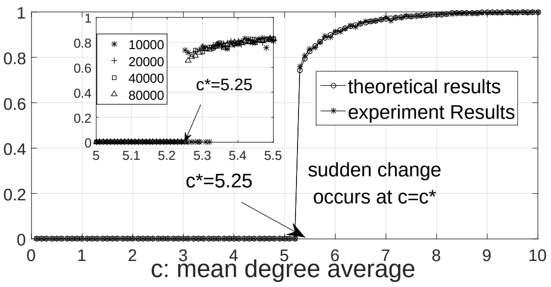


Figure 6: Numerical validation of theoretical results for core percolation in single network. μ_1 is the fraction of core nodes in a single network.

In addition, it has been shown by previous work [21] that for an undirected ER network, the fraction of core nodes will exhibit a second order phase transition at $c = e \approx 2.7$. Based on that, here we can deduce that when $c < e$, $\mu_1 = 0$; and when $c > e$, $\mu_1 > 0$ and will change continuously. Also, here μ_1 undergoes a second order transition.



(a)



(b)

Figure 7: Numerical validation of theoretical results for core percolation in coupled networks. μ_∞ is the fraction of core nodes in interdependent coupled networks. The simulations agree with our analytical results. In (b), in order to verify the discontinuity around the critical point c^*, further simulation is shown in the subgraph of (b). The subgraph is the simulation of coupled networks in shorter range. From the results in subgraph of (b), μ_∞ is either 0 or a value larger than 0.65.

5.1.2 Core Percolation in Coupled Networks. Now we turn to observe core percolation in fully interdependent coupled networks, and present below the detailed results. Again, we get empirical results according to the proposed algorithm, and the theoretical results by Equations (1)-(10), which calculate the fraction of core nodes in each iteration.

Fig. 7(a) plots the fraction of final remaining nodes from 9 different coupled networks (in number order): pokec social network, patent citation network, Astro Physics collaboration network, gplus, High-energy physics theory citation network, Amazon product co-purchasing network, High-energy physics citation network, facebook, Slashdot social network. These 9 networks are extracted respectively from: **Citation Networks** [25], **Social Networks** [26], **Amazon Product Co-purchasing Network** [27], **Collaboration Networks** [28] ranging from 4039 nodes (88234 edges) to 3,774,768 nodes (16,518,948 edges). From the figure, we can see that the theoretical results well fit the experimental ones for each of the 9 networks. Moreover, we observe that for coupled networks that are composed of pokec social network, patent citation network and Astro Physics collaboration network, respectively, there is zero fraction of remaining core nodes when the algorithm terminates.

To gain a more clear understanding of phase transition, we also plot in Fig. 7(b) the fraction of core nodes obtained under an ER network versus different values of mean degree c. Surprisingly, here μ_∞ can remain to be zero even if $c > e = 2.7$. Such occurrence is attributed to the property of cascade failure in coupled networks. That is to say, if there are no connections between two networks, it will be reduced to the single network and core nodes will appear at the critical point $e = 2.7$. However, in interdependent networks, a lot of core nodes in one of the networks will be removed because of the influence from the other one. Under such circumstance, even if $c > e$, the whole coupled networks can still possibly undergo a total cascade failure, meaning that the failure of a few nodes in a network will lead to the collapse of the whole system. Particularly, in our simulation, we observe that at the point $c = 5$, μ_1 is about 0.9, a value close to 1. However, ultimately, μ_∞ is still 0.

In the figure, there is a clear phase transition at the point $c^* = 5.25$, where the fraction of core nodes undergoes an abrupt change from 0 to around 0.7. At this point, as the total number of nodes N increases, μ_∞ also exhibits a gradual increase following the form of a ladder function. Therefore, it is persuasive to say that when N goes to infinity, μ_∞ will exhibit a sudden change. This condition concludes that at the critical point c^*, μ_∞ is discontinuous, meaning that μ_∞ undergoes a first order phase transition.

Comparison between single and coupled networks: Prior work has demonstrated that in a single network, for similar kinds of networks, the number of core nodes n_{core} will be zero if c is small but be larger than zero when c becomes large. In that case, the n_{core} will change continuously at the critical point, which is a second order phase transition. However, when it comes to coupled networks, at a critical point which is larger than that in a single network, the fraction of core nodes μ_∞ will change discontinuously, which is a first order transition. Furthermore, in interdependent networks, even if c is very large, the networks will still have a large chance of collapse. Another example is the generating network, where an initial removal of a few nodes will cause the coupled system to break down ultimately.

5.2 Explanation of Sudden Change

As observed in our experiments in different coupled networks, some systems exhibit a stable μ_∞ whereas the others collapse to zero. And among those that end up with a collapse, some may collapse to zero after only two or three iterations while some may reach the collapse after many iterations. Moreover, we find that the systems around a ‘critical point’ which collapse in the end was ‘stable’ at some intermediate iterations. To illustrate this phenomenon, here we take two real networks, i.e., Twitter network and Web-Stanford network for example, as shown in Fig. 8(a). Specifically, Twitter network tends to slow down the decreasing speed. But after some iterations, small perturbations (a random removal of only few nodes) in the next iteration causes the system to collapse very fast. If it adds a few more edges, the system will converge as soon as the decreasing speed goes down. Thus around the critical point, some networks converge to a non-zero value but others collapse and sudden change occurs. In contrast, the Web-Stanford [29] network system is not around a critical point and the number of core nodes goes down quickly.

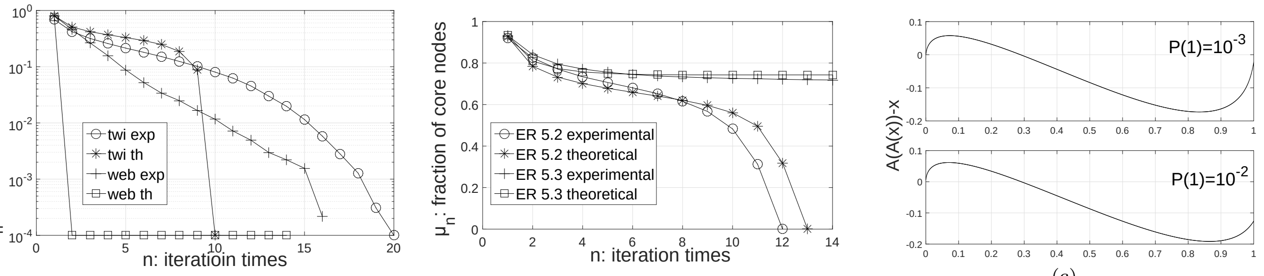
The reason of such phenomenon is that the systems should have converged, if they did not undergo a small perturbation such as random removal of a small number of nodes. Interpreted more technically, many systems can converge to coupled networks with each network having truncated a power law node degree distribution satisfying $P(0) = P(1) = 0$, but a small perturbation such as adding some $P(1)$s will lead to the collapse of the systems. We further graphically represent this in Fig. 8(c), where we provide the diagram of $A(A(x)) - x$ in two subfigures corresponding to two distributions. In particular, the upper subfigure shows a graph with power law distribution and $P(0) = P(1) = 0$, but given a small perturbation of $P(1) = 10^{-3}$. The lower subfigure represents the situation where a perturbation of $P(1) = 10^{-2}$ is introduced. From the curves we find that both functions have only one root, i.e., $1 - \alpha' - \beta' = 0$, meaning that the fraction of core nodes will eventually be zero under both cases[21]. This confirms our conclusion that the system is unstable around a critical point, where adding some leaf nodes can destroy the system, which collapses very fast.

We further disclose, technically, why the fraction μ_∞ of final remaining core nodes undergoes a first order phase transition. In single network, assume out-degree and in-degree distributions as $P^+(k)$ and $P^-(k)$, respectively. In a directed network, a bipartite graph is used to calculate the fraction of core nodes and usually $P(k)^+ \neq P^-(k)$. Thus at the critical point $A^+(A^-(x)) - x = 0$ has a smaller single root and bigger double root and the fraction of core nodes will change discontinuously according to [21].

Notice that our model is similar to the conditions above in the sense that during the iteration, our algorithm aims to find the root of function $f(g(x)) - x = 0$, where function f can be treated as a GLR operator in a single network, and function g can regarded as a randomly attacking operator. It is difficult to judge the number of roots the function has by simply looking at the graph. However, in analogy with the technique in previous research, that $f \neq g$ (apparently two operator is different) causes the first order transition. At the critical point there will exist a two-multiple root, leading to a sudden change occurrence at the smallest root x^*. This helps us explain why a sudden change occurs under our model.

6 CONCLUSION

The paper has established a model of core percolation in coupled networks that are fully interdependent under a non-feedback condition. From a chosen network, we adopt GLR algorithm for node removal that will cause some nodes to be removed in the other network. Such iteration repeats alternatively until no more nodes in the networks can be further



(a) (b) (c)

Figure 8: Explanation for first order transitions. (a) Both Twitter and Web networks collapse to 0, but with different forms of collapse. Twitter becomes 'stable' for a time and collapses, while web goes down very quickly. It is convincing that twitter is around its critical point. (b) An ER network with $c = 5.2$ undergoes collapsing just like Twitter. Around this critical point, some networks converge to a non-zero value while others collapse. This incurs to a sudden change. (c) Both of the two functions have a unique root with $A(A(x)) - x = 10^{-4}, 10^{-3}$. Hence, the power law graph system collapses by small perturbations, which explains why Twitter network does not exhibit convergence but collapses.

removed. The result of this model shows significant difference from that on traditional single network, where the core nodes appear at relatively small mean degree and undergoes a second order phase transition. In contrast, in coupled networks, we find that the core nodes will maintain relatively large mean degree and fraction of the core nodes will undergo a first-order phase transition. Some interesting future directions are provided in **Appendix C in our technical report** [32].

REFERENCES

[1] K. Bhawalakr, J. Kleinberg, K. lewi, T. ROUGHGARDEN and A. Sharma, "Preventing Unraveling in Social Networks: The Anchored k-Core Problem", in *SIAM J.DISCRETE MATH*, Vol.29, No.3, pp.1452-1475, 2015.

[2] S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Menders, "k-Core Organization of Complex Networks", in *Phys.Rev.Lett 96,040601*, 2006.

[3] Christos Giatsidis, Dimitrios M. Thilikos, Michalis Vazirgiannis, "Evaluating cooperation in communities with k-core structure", in *2011 International Conference on Advances in Social Networks Analysis and Mining.*

[4] Abhijin Adiga and Anil Kumar S. Vullikanti. "How Robust Is the Core of a Network?", in *Machine Learning and Knowledge Discovery in Databases*, pp 541-556, 2013

[5] C. F. Chiasserini, M. Garetto and E. Leonardi, "Deanonymizing Scale-free Social Networks by Percolation Graph Matching", in *2015 IEEE INFCOM*, pp.1571-1579.

[6] M. Bauer, O. Golinelli, "Core percolation in random graphs: a critical phenomena analysis", in *The European Physical Journal B-Condensed Matter and Complex Systems*, Vol. 24, No. 3, pp. 339-352, 2001.

[7] Y. Liu, J. J. Slotine and Albert-László, "Controllability of Complex Networks", in *nature10011*, Vol.473, 2011

[8] Y. Liu, E. Csóka, H. Zhou, M. Pósfai, "Core Percolation on Complex Networks", in *Physical Review Letters*, 109(20), 205703.

[9] N. Azimi-Tafreshi, S. Dorogovtsev, J. Mendes, "Core organization of directed complex networks", in *Physical Review E*, 87(3), 032815.

[10] T. Jia, M. Pósfai, "Connecting Core Percolation and Controllability of Complex Networks", in *SCIENTIC REPORTS*, DOI: 10.1038, 2014.

[11] R. Parshani, S. V. Buldyrev and S. Havlin, "Interdependent Networks: Reducing the Coupling Strength Leads to a Change from a First to Second Order Percolation Transition", in *PRL 105,048701(2010)*

[12] J. Gao, S. V. Buldyrev, H. E. Stanley and S. Havlin, "Networks Formed from Interdependent Networks", in *nature physics*, 2011.

[13] S. M. Rinaldi, J. P. Peerenboom and T. K. Kelly, "Identifying, Understanding, and Analyzing Critical Infrastructure Interdependencies", in *IEEE Control Systems*, Vol.21, 2001

[14] J. Gao, S. V.Buldyrev, H. E. Stanley, X. Xu and S. Havlin, "Percolation of a General Network of Networks", in *Phys. Rev.E88,062816*, 2013.

[15] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley and S. Havlin, "Catastrophic Cascade of Failures in Interdependent Networks", in *nature08932*, Vol.464, 2010.

[16] Osman Yağan, Dajun Qian, Junshan Zhang, and Douglas Cochran. "Conjoining Speeds up Information Diffusion in Overlaying Social-Physical Networks", in *IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS*, VOL. 31, 2013.

[17] Zhen Wang, Attila Szolnoki, and Matjaž Perc, "Interdependent Network Reciprocity in Evolutionary Games", in *Physics and Society*, arXiv:1308.1947, 2013.

[18] Enrico Zio, Senior Member, IEEE, and Giovanni Sansavini, "Modeling Interdependent Network Systems for Identifying Cascade-Safe Operating Margins", in *IEEE Transactions on Reliability*, Vol.60, pp.94-101, 2011.

[19] M. E. J. Newman, S. H. Strogatz and D. J. Watts, "Random Graphs with Arbitrary Degree Distributions and Their Applications", in *DOI: 10.1103/PhysRevE.64.026118.*

[20] M. E. J. Newman, "Spread of Epidemic Disease on Networks", in *PHYSICAL REVIEW E 66,016128(2002)*

[21] Y. Liu, E. Csó "Core Percolation on Complex Networks", in *PRL 109,205703(2012)*

[22] M. Bauer and O. Golinelli, "Core Percolation in Random Graphs: a Critical Phenomena Analysis" in *Eur.Phys.L.B 24,339-352(2001)*

[23] Microsoft Academic Graph, https://www.microsoft.com/en-us/research/ project/microsoft-academic-graph/.

[24] M. Ripeanu and I. Foster and A. Iamnitchi. "Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design", in *IEEE Internet Computing Journal*, 2002.

[25] J. Leskovec, J. Kleinberg and C. Faloutsos. "Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations", in *ACM SIGKDD International Conference*

on Knowledge Discovery and Data Mining (KDD), 2005.

[26] J. McAuley and J. Leskovec. "Learning to Discover Social Circles in Ego Networks", in *NIPS*, 2012.

[27] J. Leskovec, L. Adamic and B. Adamic, "The Dynamics of Viral Marketing". in *ACM Transactions on the Web (ACM TWEB)*, 1(1), 2007.

[28] J. Leskovec, J. Kleinberg and C. Faloutsos. "Graph Evolution: Densification and Shrinking Diameters", in *ACM Transactions on Knowledge Discovery from Data (ACM TKDD)*, 1(1), 2007.

[29] J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney. "Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters", in *Internet Mathematics* 6(1) 29–123, 2009.

[30] J. Shao, S. V. Buldyrev, R. Cohen, M. Kitsak, S. Havlin, "Fractal Boundaries of Complex Networks", in *Europhysics Letters Association*, Vol.84, 2008.

[31] J. Shao, S. V. Buldyrev, L. A. Braunstein, S. Havlin, and H. E. Stanley, "Structure of Shells in Complex Networks", in *Phys. Rev. E 80, 036105*, 2009.

[32] J. Pan, Y. Yao, L. Fu, "Core Percolation in Coupled Networks", technical report, in http://rogerfu.weebly.com/uploads/8/2/8/1/82817902/core_percolation_technical_report.pdf.

APPENDIX

A. Time Complexity of the Algorithm

On one hand, the complexity of the model defined in section 2.2 is high, mainly due to the high complexity of traditional GLR in single network, where leaf node with degree one, along with their neighbors, are deleted. However, we need to search the second time the nodes with degree one and delete their neighbors. We do not stop searching the nodes until no nodes with degree one are found. Since we have no prior knowledge of the total number of times that we need to search the nodes, the worst times cost is $O(N)$.

Thus for the model the complexity is composed of two parts: (i) recording the edges and links and get degree distribution. The least cost maybe $O(N)$. In the worst case, if we use the method of adjacency matrix, the cost will be $O(N^2)$; (ii) deleting nodes with degree one, along with their neighbors, then get new degree distribution. This cost is nearly the same as that incurred in (i); (iii) repeat (i) and (ii) iteratively until no leaf nodes are left. In the worst case we delete only $O(1)$ nodes each time. So the cost is $O(N)$. In conclusion, for the model the complexity will be between $O(N^2)$ to $O(N^3)$.

On the other hand, according to our derivation, the computation occurs in three aspects: (i) Recording the degree distribution.(ii) randomly removing nodes; and (iii) operating GLR in a single network. In fact, in real networks, except the web graph which has a few nodes with very high degree, it is reasonable that we record the degree distribution up to degree $\sqrt{N}$.

(1) In part (i), we should input the nodes and their edges. According to the real networks, we suppose that the number of edges versus that of nodes in a network is less than 100. That is to say, the numbers of edges and nodes can be treated to be at the same order. Hence when the edges are recorded, it will cost less than $N \times 2$ computations. Here exponent 2 comes from the that for each edge, there are two nodes whose degrees will be increased by one. After recording the edges, we have known the degree of each node. Since we have to know the degree distribution, we need to scan the whole nodes. For a value of k, we find out the number of nodes with that degree k, and mark each of those nodes with 1. Otherwise, the nodes will be marked as 0. The scanning continues until the value reaches $N-1$ to make sure all of the information is recorded, and the cost is $O(N(N-1)) = O(N^2)$. Thus, the overall complexity in this part should be $O(N^2)$. (1) In part (i), the recording process is the same with part (i) of the model. Because we only need to record the degree up to $\sqrt{N}$, the complexity will not be larger than $O(N^{\frac{3}{2}})$.

(2) In part (ii), according to equation (1), the m is ranged from 1 to $\sqrt{N}$. The complexity should be $O(\sqrt{N}^3)$.

(3) In part (iii), calculation of GLR in a single network consists of two steps: changing the degree distribution and calculating μ_{2n-1} or μ_{2n}. Suppose the number of degree probability recorded is M. For the first step, since we just change $P(2)$ and for the other probabilities we just divide it with a normalization number, the cost is only $O(M)$ and far smaller than that of random node removal. Then we have to get the root of the equation $A(A(x))-x = 0$. $A(x)$ is an $M-1$ order polynomial. For an order k, it will cost $\frac{1}{2}k(k+1)+k$ in $A(A(x))$ when calculating $(a_M x^k + \ldots + a_1 x + a_0)^k$, where $k \in [0, 1, \ldots, M]$, and the total cost should be $O(\sum k^2) = O(M^3)$. When we start to get the root, we divide the interval $[0, 1]$ into constant parts (usually around 10^4). The total cost is $O(10^4 \times M^3)$. In fact, if $n_{core} \neq 0$, the root is small and the total cost will not excess $O(10^3 \times M^3)$. Moreover, calculation of n_{core} cost $3 \times M^3$. As a result, the total cost is $M^3 = O(N^{\frac{3}{2}})$.

In conclusion, the time complexity will be $O(N^{\frac{3}{2}})$. We find that our theoretical derivation has less complexity than that of model.

Remark 3. *The algorithm undergoes several iterations, of which the number seems to be irrelevant to N. We can stop the iteration if the difference between the last two iterations is smaller than a given number, ϵ. If the system converges, the number of iterations is only relevant to the given number. It is found out by further research that the iteration is very fast and the fraction of core nodes undergoes the change of iteration time at an exponential speed. Under such circumstance, the iteration times should be proportional to $-log\epsilon$. In fact, the iterative result can be close to the real result after only three or four iterations. If the system collapses to zero, the experiments that will be demonstrated in the sequel shows that among all the realistic networks, regardless of the network scale, the number of iterations are always below 20. Hence, the total time complexity remains to be $O(N^{\frac{3}{2}})$.*

B. Percolation in Coupled Networks

Recall that in our analysis of core percolation of coupled networks, we need to renew both the fractions of core nodes and nodal degree distributions. As a counterpart of core percolation, percolation theory has also been adopted to study the vulnerability of coupled networks in previous literature [11], where, however, only the fraction of nodes in giant component are updated at each iteration. As a result, it is interesting to explore why the degree distribution does not need to be renewed in that case since it will provide useful insight into the present work. Based on the derived results in Section 4, we are thus motivated to propose further insight about percolation theory in coupled networks.

A drawback that prior work [11] suffers is that there is lacking of sufficient explanation of the equation derivation of percolation on coupled networks. Therefore, we aim to obtain a more general equation for coupled networks ,along with a further explanation for the non-feedback condition. In doing so, we define ψ_n and ϕ_n as the fractions of survived nodes at n^{th} iteration in networks A and B, and ψ'_n and ϕ'_n as the fractions of nodes in giant components of networks A and B. Also, we let $g_A(x)$ ($g_B(x)$) represent the probability that a randomly chosen node belongs to a giant component in network A (B) after a fraction $1-x$ of nodes are randomly removed from that network. The analytical solutions of $g_A(x), g_B(x)$ are available in [30][31] and we do not reproduce it here. Be aware that $g_A(1), g_B(1)$ is not 1 in general.

We proceed by first providing analysis on the fully interdependent coupled networks, and then extending the result to a more general from of coupled networks.

1. A Special Case: Fully Interdependent Coupled Networks.

LEMMA 6.1. *For a fully interdependent coupled networks ($q=1$) under non feedback condition, we have*

$$\psi_n = pg_B(\phi_{n-1}), \qquad \phi_n = pg_A(\psi_n).$$

PROOF. We prove using the induction on the number of iterations the algorithm undergoes. Obviously, the equation holds at the first iteration. Suppose the algorithm satisfies this equation when $k \leqslant n-1$. Now we are concerned about the case when $k=n$.

Since $q=1$ in fully interdependent coupled networks, all of the survived but non-functional nodes in B will affect the area ψ_{n-1}, where, as a result, the fraction of functional nodes should be $\phi_{n-1}g_B(\phi_{n-1})$. Now we proceed to calculate the size of the giant component containing the survived nodes. Here we point out that after iteration, the degree distribution will change, meanwhile leading to the change of g_A. In order to avoid this, we assume that some removed nodes are still 'survived', with their links also reserved. Note that these nodes have no connection with the giant component, hence whether their links are reserved or deleted will not affect the calculation of the giant component. And the purpose of reserving these nodes is only to sustain the original degree distribution. Based on that, at the $(n-1)^{th}$ iteration, the 'equivalent' survived nodes is ψ_{n-1}. To maintain original degree distribution, we have to remove the same proportion of nodes at area $\phi_{n-1}(1-g_B(\phi_{n-1}))$. Thus, the fraction of the whole 'survived' nodes should be

$$\psi_n = \frac{\phi_{n-1}g_B(\phi_{n-1})}{g_A(\psi_{n-1})} = \frac{pg_A(\psi_{n-1})g_B(\phi_{n-1})}{g_A(\psi_{n-1})} = pg_B(\phi_{n-1}).$$

Because the degree distribution is the same as the original one, we get

$$\psi'_n = \psi_n g_A(\psi_n).$$

Similarly, we also have

$$\phi_n = \frac{\psi_n g_A(\psi_n)}{g_B(\phi_{n-1})} = \frac{pg_B(\phi_{n-1})g_A(\psi_n)}{g_B(\phi_{n-1})} = pg_A(\psi_n)$$

and

$$\phi'_n = \phi_n g_B(\phi_n).$$

From the equations, we conclude that the fractions of calculated survived nodes ψ_n, ϕ_n are larger than those of actual nodes in order to maintain degree distribution, but the number of giant components nodes ψ'_n, ϕ'_n are correct. This completes our proof. □

2. General Coupled Networks. Upon analysis of fully interdependent networks, now we take a further look into more general coupled networks, where networks A and B can form a partially dependent pair if a certain fraction q_{BA} ¿ 0 of nodes in network A directly depend on nodes in network B, i.e., nodes in network A cannot function if the corresponding nodes in network B do not function. Similarly, the fraction q_{AB} can be interpreted in a symmetric manner. Moreover, in such general networks, we assume that after an attack or failure only a fraction of nodes p_A (p_B) in network A (B) remains, as is commonly supposed in prior works of percolation. Then we state below our conjecture regarding the fractions of survived nodes in networks A and B under such general coupled networks. Since it is quite challenging to prove the case for all n, we take a step ahead by proving the special case where $n=2$, and will leave it a future work for the proof of an arbitrary iteration n.

CONJECTURE 1. *For general form of non-feedback condition in interdependent networks, we have*

$$\psi_n = p_A[1 - q_{BA}(1 - g_B(\phi_{n-1})p_B)]$$
$$\phi_n = p_B[1 - q_{AB}(1 - g_A(\psi_n)p_A)].$$

PROOF. As stated above, we only finish the proof of the second iteration. Suppose that $N = \frac{N_B}{N_A}$. Then according to the definition, we have $q_{AB} = \frac{1}{N}q_{AB}$ After an initial attack, it is apparent that

$$\psi_1 = p_A, \qquad \psi'_1 = p_A g_A(p_A).$$

Then the nodes that do not belong to the giant component will affect network B. We divide the network B into two parts: removed (attacked) part with a fraction $1-p_B$ and survived part with a fraction p_B. Since the connection is established in a random mode, the survived part can be simply given by

$$\begin{aligned}\phi_1 &= p_B[N - q_{BA}(1 - p_A g_A(\psi_1))]/N \\ &= P_B(1 - q_{AB}(1 - P_A g_A(\psi_1))),\end{aligned}$$

implying that the difference of the number of nodes between networks A and B does not effect the results. Then the fraction of nodes in giant component is

$$\phi_1' = \phi_1 g_B(\phi_1).$$

Since the $q_{AB}(1 - p_A g_A(p_A))$ part in network B is connected with the non functional part of network A, only $1 - q_{AB}(1 - p_A g_A(p_A))$ part will be possible to be connected with $p_A g_A(\psi_1)$. Now we need to find out the fraction that affects $p_A g_A(\psi_1)$. That is to say, in $1 - q_{AB}(1 - p_A g_A(p_A))$ part, the fraction of non-functional nodes will be

$$\begin{aligned}&(1 - p_B)(1 - q_{AB}(1 - p_A g_A(p_A))) + \phi_1(1 - g_B(\phi_1)) \\ &\quad = (1 - p_B g_B(\phi_1))[1 - q_{AB}(1 - p_A g_A(p_A))],\end{aligned}$$

and this part will be connected to the functional part in network A. Randomly, there will be a fraction $p_A g_A(p_A) q_{AB}$ of nodes that will be influenced by network B. Since the model is in a non-feedback mode, that influence must stem entirely from the part $1 - q_{AB}(1 - p_A g_A(p_A))$. Under such circumstance, the connection index has changed due to the different changes incurred by those two parts. And the equivalent connection index can be thus expressed as

$$q_{AB}' = \frac{q_{BA} p_A g_A(p_A)}{1 - q_{AB}}(1 - p_A g_A(p_A)).$$

We know the survived nodes in network A. Similarly, in order to maintain the degree distribution, we should make some survived nodes not in $p_A g_A(\psi_1)$ but in $p_A(1 - g_A(\psi_1))$, and meanwhile reserve the links of those nodes. Certainly those nodes have no connection with the nodes in the giant component and thus will not affect the size of the giant component. Divided by a fraction $g_A(\psi_1)$, the whole fraction of actual survived nodes should be be enlarged to the range of ψ_1. Hence, ψ_2 should be

$$\frac{p_A g_A(p_A) - q_{AB}'[1 - p_B g_B(\phi_1)][1 - q_{AB}(1 - p_A g_A(p_A))]}{g_A(p_A)}.$$

Rearranging the equation, we have

$$\begin{aligned}\psi_2 &= p_A[1 - q_{BA}(1 - g_B(\phi_1) p_B)] \\ \psi_2' &= \psi_2 g_A(\psi_2),\end{aligned}$$

where ψ_2 is larger than the fraction of actual survived nodes with a fraction of $g_A(\psi_1)$. Since the $1 - \psi_1 g_A(\psi_1)$ part of network A has already influenced network B, and only a fraction $1 - q_{BA}(1 - P_B g_B(\phi_1))$ of nodes survive after B, thus the part which has a chance to influence network B is $\psi_1 g_A(\psi_1)[1 - q_{BA}(1 - P_B g_B(\phi_1))]$. As a result, the probability that the nodes which have connection and survived last time will continue to survive in B this time is

$$P = \frac{\psi_2 g_A(\psi_2)}{\psi_1 g_A(\psi_1)[1 - q_{BA}(1 - P_B g_B(\phi_1))]} = \frac{g_A(\psi_2)}{g_A(\psi_1)}.$$

In analogy with the solution of ϕ_1, we have

$$\begin{aligned}\phi_2 &= \phi_1\left[1 - \frac{q_{AB} p_A g_A(p_A)}{1 - q_{AB}(1 - p_A g_A(p_A))}(1 - P)\right] \\ &= p_B[1 - q_{AB}(1 - g_A(\psi_2) p_A)],\end{aligned}$$

completing the proof. □

In summary, we further list below three key points regarding our analysis:

(1) The theoretical result of the number of 'survived' nodes in each iteration is larger than that of the actual survived nodes. However, via the equivalent equation, the theoretical result provides an effective way to get around calculating the generating function of the giant component, which may be rather difficult to analyze in core percolation of multiple networks.

(2) The connection between the two networks is fixed and it will not change with iterations. For example, when we are analyzing ψ_2, the $\psi_1 g_A(\psi_1)$ part of network A only depends on $1 - q_{AB}(1 - p_A g_A(\psi_1))$ part of network B. Furthermore, if in each iteration the connection is random and different, the model may be described as

$$\begin{aligned}\psi_n &= \psi_{n-1}[1 - q_{BA}(1 - g_B(\phi_{n-1})\phi_{n-1})] \\ \phi_n &= \phi_{n-1}[1 - q_{AB}(1 - g_A(\psi_n \psi_n))].\end{aligned}$$

(3) Both the connection (dependent) indices q_{AB} and q_{BA} change with iterations.

C. Prospect for Further Directions

There remain some interesting future directions that are worth considering. For example, it is promising to investigate how to simplify the derivation and decrease the time complexity. In our model, different from **Appendix B**, removing all of the removable nodes is not equivalent to removing a fraction of removable nodes. If we remove only part of the removable nodes, a problem appears: because many removable nodes are connected with core nodes, it is possible that the core nodes will change. Therefore in general, the core nodes after removing part of removable nodes are different from that after removing all of removable nodes. In consequence, in our model, we cannot use the technique like **Appendix B**. In summary, in percolation theory, giant component can match very well with dependent removing process: in core percolation theory, core is 'incompatible' with dependent removing.

Also, we would like to present a look into a more general coupled network where q is not limited to one. In one of the existing works, the authors build a model to search P_∞ with the change of initial attack p and interdependent index q. This model describes RR network of ER (or SF) networks. In non-feedback model, the research find that if p, q goes along the critical line $p_c(q)$, the system undergoes two phase transitions: first and second order transitions. In our model, our variable should not be p, q but c, q. To some extent we just replace initial attack index p with mean degree c.

We predict that in some cases when q is small, P_∞ have second order transition: when q is larger, P_∞ have first order transition. In fact, in core percolation theory, single network (undirected) is equivalent to $q = 0$, while model of fully interdependent networks means that $q = 1$. We believe that there also exists a critical line:

$$c = c_c(q)$$

which is also the phase transition critical line. In other words, for any $0 \leqslant q \leqslant 1$, there exists a critical point c_c, at point $(c_c(q), q)$, μ_∞ will have phase transition from $c < c_c(q)$ to $c > c_c(q)$. Because at $q = 1$ there still exist phase transition, we believe that along the line $c_c(q)$, there will have a second order phase transition but no first order transition. In short, there will be no two phase transitions. We want to find some ways to let the critical line undergo two phase transitions. Remember the parameter m: number of networks that a network depends on. In our model: coupled networks, apparently $m = 1$. If we enlarge our model and make $m = 2, 3$ or more, then the area $\mu_\infty > 0$ will apparently decrease because of extra influence. At this time ($m > 1$), when q is large, maybe no matter how large the c is, μ_∞ is 0. Removing small part of nodes still cause total collapse of the system. In this case along the critical boundary line $c = c_c(q)$, two phase transitions occurs. Devoting to discovering two phase transitions in simulation and using reasonable derivation to realize two phase transitions is the work with great theoretical value.

In addition, μ_∞ is not the only result the researchers want to get. The fractions of α, β, γ removable nodes are also very important. Knowledge about the three kinds of removable nodes can be applied to many fields. For example, in [10], the paper gives conclusions about the controllability and robustness of a single network by using the results of n_α and $n_{\beta'}$.