Cascading failures in networks with proximate dependent nodes

Yosef Kornbluth,* Steven Lowinger, Gabriel Cwilich, and Sergey V. Buldyrev
Department of Physics, Yeshiva University, 500 West 185th Street, New York, New York 10033, USA
(Received 22 October 2013; revised manuscript received 21 February 2014; published 20 March 2014)

We study the mutual percolation of a system composed of two interdependent random regular networks. We introduce a notion of distance to explore the effects of the proximity of interdependent nodes on the cascade of failures after an initial attack. We find a nontrivial relation between the nature of the transition through which the networks disintegrate and the parameters of the system, which are the degree of the nodes and the maximum distance between interdependent nodes. We explain this relation by solving the problem analytically for the relevant set of cases. In the process, we solve a variant of Rényi's parking problem on treelike graphs.

DOI: 10.1103/PhysRevE.89.032808 PACS number(s): 89.75.Hc

I. INTRODUCTION

Previous studies of the robustness of interdependent networks have focused on networks in which there is no constraint on the distance between the interdependent nodes [1–13]. However, many dependency links in the real world connect nearby nodes. For example, the international network of seaports and the network of national highways form a complex system. As seen recently from the effects of Hurricane Sandy in New York City, if a seaport is damaged, the city that depends on it will become isolated from the highway network due to the lack of fuel. Similarly, a city without roads cannot supply a seaport properly. However, a city will depend on a nearby seaport, not on one across the world. Li *et al*. [14] investigated distance-limited interdependent *lattice* networks by computer simulations and found that allowing only local interdependency links changed the resilience properties of the system. Here, we study the analogous problem in the case of analytically tractable random networks.

We study the mutual percolation of two interdependent random regular (RR) graphs. We build two identical networks, A and B, each of whose nodes are labeled $1, \ldots, N$. Each node is randomly connected by edges to exactly k other nodes in such a way that the two networks have identical topologies. We then create one-to-one bidirectional dependency links, requiring that the shortest path between the interdependent nodes does not exceed an integer constant ℓ.

Formally, we establish two isomorphisms between networks A and B, a topological isomorphism and a dependency isomorphism. The topological isomorphism is defined for each node A_i as $T(A_i) = B_i$ and $T(B_i) = A_i$. If A_i and A_j are immediate neighbors in network A, then B_i and B_j are immediate neighbors in network B and vice versa. The dependency isomorphism has the property that if $D(A_i) = B_k$, then $D(B_k) = A_i$. We create a restriction that $B_k = D(A_i)$ only if there are a maximum of ℓ connectivity links on the shortest path connecting A_i and $A_k = T(B_k)$.

There are many methods to create an isomorphism D that satisfies the shortest path length restriction. The method employed by Li *et al*. for lattices was based on creation of a random permutation of indices of the nodes that satisfies the maximal distance restriction. This random permutation was created iteratively by attempting a sufficiently large number of elementary transpositions, each of which satisfied the distance restriction. This method does not work in random networks, because these transpositions switch the dependencies of a pair of neighboring nodes and therefore cannot create permutations which form the long loops that exist in random networks. The problem of creating a random permutation that satisfies the length restriction in random graphs is a formidable problem. Various approaches such as identifying loops in the network by the Hopcroft-Tarjan algorithm [15] cannot guarantee the selection of all random permutations that satisfy the length restriction with equal probability.

Therefore, for simplicity, we introduce two more restrictions in the dependency isomorphism. We set $D(A_i) = B_i$ only if there are no other possibilities for $D(A_i)$. Additionally, we require that if $D(A_i) = B_k$, then $D(B_i) = A_k$. This restriction decreases the computation time needed to generate a permutation and, more importantly, precisely defines the algorithm creating the dependency isomorphism. This makes the model analytically tractable. We will call this method of establishing dependency isomorphism the "parking algorithm" due its similarity to Rényi's parking problem (see Appendix A). We will address the changes in the results obtained using other algorithms in Appendix C.

Following the mutual percolation model described in Buldyrev *et al*. [1], we destroy a fraction $(1 - p)$ of randomly selected nodes in A. Any nodes that, as a result, lost their connectivity links to the largest cluster (as defined in classical, single-network percolation theory [16,17]) are also destroyed. In the next stage, nodes in B that have their interdependent nodes in the other network destroyed are also destroyed. Consequently, the nodes that are isolated from the largest cluster in B as a result of the previous destruction of nodes in B are also destroyed. The iteration of this process, which alternates between the two networks, leads to a cascade of failures. The cascade ends when no more nodes fail in either network. The pair of remaining largest interdependent clusters in both networks is called a largest mutual component. If in the thermodynamic limit $N \to \infty$ the fraction of nodes μ in the largest mutual component is greater than zero, it is called the giant mutual component.

*yosef.kornbluth@mail.yu.edu

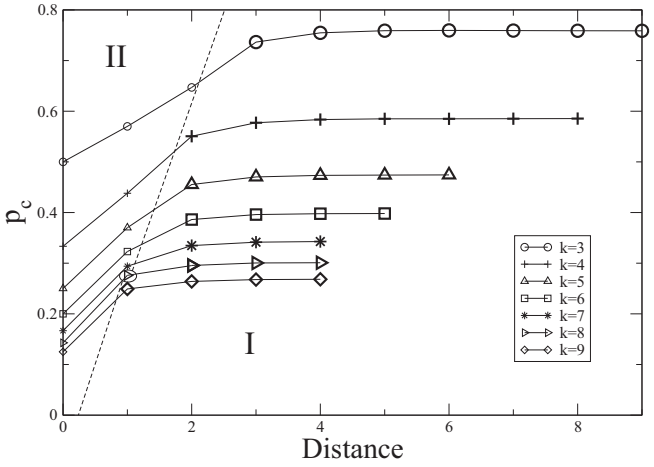

FIG. 1. Numerical results of p_c vs distance (ℓ), obtained by analyzing 100 realizations of networks consisting of $N = 10^6$ nodes for different degrees of connectivity (k). Large symbols denote first-order transitions, while second-order transitions are represented by small symbols. The regions of first-order and second-order transitions are denoted on the graph and are separated by the dashed line. The ellipse around the value $p_c(8,1)$ indicates the special case of almost coinciding first- and second-order transitions.

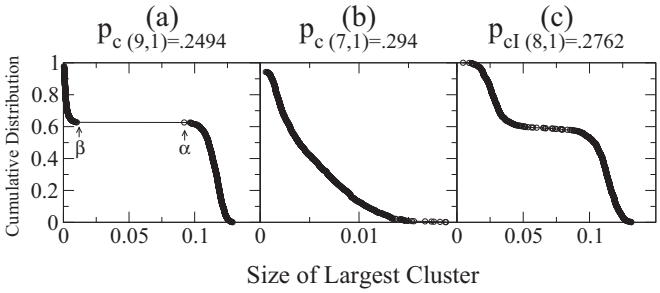

FIG. 2. The cumulative distribution of the fraction μ of nodes in the largest cluster at p_c. Each data point represents the largest cluster for 1 of 1000 realizations. These graphs represent networks of 10^6 nodes; similar patterns can be found for other sizes. In panel (a), a first-order transition, none of the simulations result in $\beta \ll \mu \ll \alpha$. In contrast, a second-order transition, shown in panel (b), displays a distribution of μ with a continuous derivative. The third figure shows the transition at $p_c(8,1)$; it is similar to a first-order transition, but the gap between β and α cannot be clearly identified.

II. SIMULATION RESULTS

As p decreases, the giant component also decreases in size and eventually disappears at the value of p that we call critical p, or p_c. We will denote p_c for a degree k and distance ℓ as $p_c(k,\ell)$. We investigate how p_c varies as a function of k and ℓ (Fig. 1).

As the distance and the degree increase, the transition at p_c shifts from a second-order transition in which μ continuously approaches zero to a first-order one, in which μ decreases discontinuously in the thermodynamic limit. The line separating these two regimes is shown in Fig. 1. Of particular note is the shift for $p_c(k,1)$. If $3 \leqslant k \leqslant 7$, the transition is second order. For $k \geqslant 9$, the transition is first order. For $k = 8$, we observe two transitions: the first-order transition at $p = p_{cI} = 0.276$ followed by the second-order transition at $p = p_{cII} = 0.269$. As we will explain, the existence of the second-order transition in this case affects the behavior of the first-order transition.

For each value of k, p_c increases monotonically as ℓ increases. This monotonic increase is in contrast with the results found by Li *et al.* [14], who found that in distance-limited *lattices*, there is a maximum in p_c as a function of ℓ. In Li *et al.*, this maximum coincided with the distance at which the transition shifts its nature from second-order for smaller distances to first-order for larger distances. In our model, a similar shift occurs at a very low value of $\ell \leqslant 2$ and is not associated with a maximum of p_c.

We describe the differences that we find between first-order transitions and second-order ones with the goal of properly characterizing the transition for $p_{cI}(8,1)$. The distribution of the largest mutual cluster differs for first- and second-order transitions. A system that undergoes a first-order transition has a single mutual giant component of size μ, as well as many much smaller clusters. When the system collapses, the giant component breaks into many smaller mutual clusters; even the largest of these clusters is much smaller than the giant component.

If the transition is of the first order, the size of the largest mutual component of a finite network at a fixed value of $p \approx p_c$ may fluctuate dramatically. Due to fluctuations always present in a finite system, μ can be close to the value α, which is the fraction of nodes in the mutual giant component for $N \to \infty$ at $p = p_c + \epsilon$, where ϵ is a small positive number. In other cases, it can be close to β, the value of μ at $p = p_c - \epsilon$ for $N \to \infty$. However, no values of μ are expected in the interval between α and β. This phenomenon tells us that, due to the cascade of failures, if a cluster becomes smaller than α, then it will not reach stability until it falls below β. Figure 2(a) illustrates the first-order transition observed at $p_c(9,1) = 0.2494$; we see that there is no intermediate value of μ when the simulation is repeated for many such systems. As the size of the network increases, the distribution concentrates around α and β, with the probability density approaching a δ function at these two points. We define p_c as the point where half of the realizations lead to each result. As the size of the network increases, β goes to 0 in most cases. The sole exception is $p_{cI}(8,1)$ [Fig. 2(c)].

In the case of second-order transitions, the drop in $\mu(p)$ is less dramatic. There is no gap in the size distribution of the largest cluster [Fig. 2(b)]. Rather, there is a steady decline of the cumulative distribution of μ. Thus, α and β are not defined.

These two categories describe every transition with the exception of $p_c(8,1)$. There, we see a plateau in the cumulative distribution of the largest component size, reminiscent of a first-order transition. However, there is no clear gap in the distribution of the mutual giant component in a finite network and we cannot clearly identify α and β (Fig. 2). A second noticeable difference between the two types of transitions is the progress of the cascade of failures in the network. These cascades are the cause of the network's collapse, as described in Buldyrev *et al.* [1]. The order of the transition determines the behavior of the cascades. In a first-order transition, all of the realizations begin with an abrupt initial decay, as the first stages of the cascade take their toll [Fig. 3(a)]. The realizations that

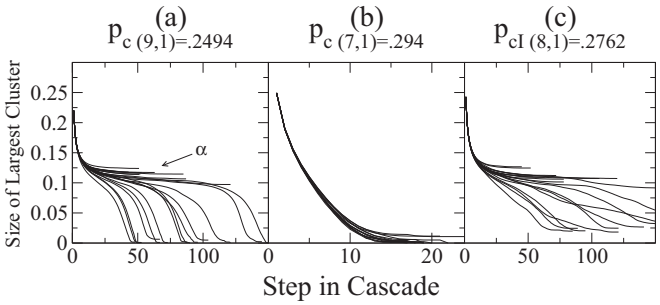

FIG. 3. The size of the giant cluster (μ), as a function of the stage in the cascade. Each line represents a different simulation. In the first-order transition (a), the simulations that quickly stabilize form asymptotic lines near α, the smallest possible giant component. The simulations that collapse decay quickly once they decrease to below α. In the second-order transition (b), the largest cluster decays steadily over the cascade's progression. At $p_c(8,1)$ (c), we see both behaviors.

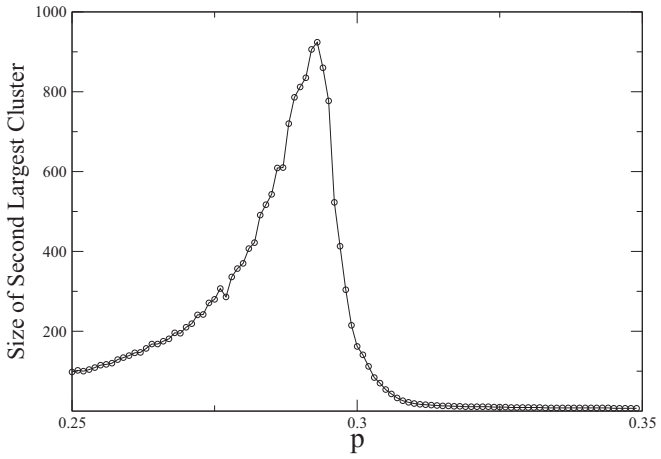

FIG. 4. The size of the second largest cluster as a function of p, for $k = 7$ and $\ell = 1$. Note the maximum at $p_c(7,1) = 0.294$.

will retain a mutual giant component then decline more slowly and eventually stabilize, generating giant components greater or equal to α. The realizations that do eventually collapse also begin to stabilize. However, instead of stabilizing, they enter a rapid decline once the largest cluster becomes smaller than α, eventually forming clusters of sizes less than or equal to β. In a second-order transition, in contrast, the largest cluster steadily declines over time [Fig. 3(b)]. In $p_c(8,1)$, the largest cluster approaches an asymptote, as it would in a first-order transition. However, the clusters that eventually collapse, rather than entering a rapid decline, decrease slowly, as if they were undergoing a second-order transition [Fig. 3(c)].

In order to determine p_c for second-order transitions, we examine the size of the second largest cluster. This quantity peaks at $p = p_c$. For $p > p_c$, the giant component spans the entire cluster, preventing other large clusters from forming. For $p < p_c$, the networks are too fragmented to allow large clusters to form. These phenomena are common to both first- and second-order transitions, but are more useful in second-order transitions because they serve as the best indication of p_c. This measure of p_c coincides with the traditional calculation of the percolation threshold, the point where μN, the average size of the largest cluster, begins to increase rapidly as a function of p. However, the average size of the second cluster as a function of p yields a more precise measure of p_c (Fig. 4) because it has a sharp peak at $p = p_c$ [3].

III. ANALYTIC SOLUTION

We offer an analytic solution for the case where $\ell = 1$, which is the simplest case, in which we see the transition shift from second order to first order.

A. Parking problem

In our simulation, we establish dependency links in the following way: At each step, we select at random a node A_i that currently does not have a dependency link and set two dependency links $D(A_i) = B_j$ and $D(A_j) = B_i$, where B_j is randomly chosen among B_i's immediate neighbors without a dependency link, if at least one such neighbor exists. We call (i,j) a dimer. If all of the neighbors of B_i already have dependency links, we set $D(A_i) = B_i$ and i is called a monomer. This system of assigning dependency links, introduced for computational efficiency, allows us to find an exact solution. In order to calculate p_c, we find the fraction of indices that are monomers,

$$m = \int_0^1 \frac{2k \exp\left(2\int_0^A \frac{(B^k-1)dB}{[B^k(k-2)-2(k-1)](B-1)}\right)}{2(k-1)-A^k(k-2)} dA - 1. \quad (1)$$

Finding the value of $1 - m$ is a version of Rényi's parking problem [18] that applies to a discrete graph; another version on a discrete graph was solved by Dehling *et al.* [19]. A full derivation of Eq. (1) appears in Appendix A.

In our analytic calculation of $\mu(p)$, we will use the probability q that a link leaving a dimer reaches a monomer. Because, in our algorithm, two monomers cannot be adjoining, any connectivity link that has a monomer at one end has a dimer at the other end. Thus, q is simply the number of links with monomers at one end divided by the number of links that leave dimers, which is $\frac{km}{(k-1)(1-m)}$. The factor $k-1$ in the denominator comes from the exclusion of the link between the two members of the dimer.

B. Analytic determination of p_c

When $\ell = 1$, it is possible to express the problem of finding the mutual giant component of two networks as the problem of finding of a giant component in a single network, while taking into account the disruption caused by the other network. To determine p_c for a single RR graph, percolation theory introduces a probability b that a randomly selected link does not lead to the giant component. This probability satisfies the equation [16]

$$b = (1-p) + pb^{k-1}. \quad (2)$$

The first term on the right side of the equation refers to the probability that the link leads to a node that was destroyed in the initial attack, while the second term represents the probability that the node survived, but does not have any

outgoing links that lead to the giant component. Previous studies of interdependent networks have used this equation [10–12], often modifying the value of p and the degree distribution [1–8]. They use these modifications to describe the diminished network at each stage of the cascade. However, this method is only useful when the nodes destroyed in each stage of the cascade can be seen as a random subset of nodes in the other network. The distance restriction in our model precludes us from doing so. We modify the equation itself to account for the local interdependency between our two networks. This allows us to study both networks at once and to account for more complex relationships between them.

When studying a link in network A, we assume that the link starts from a node A_i, such that its interdependent counterpart in network B, $B_k = D(A_i)$, *a priori* belongs to the mutual giant component. This assumption allows us to isolate the effect that the destruction of nodes in network A will have. Formally, this is done by assuming that B_k is artificially connected to the mutual giant component in a way that does not affect the topology in network A. We thus define a as the probability that a randomly chosen link leaving node A_i would not lead to the mutual giant component, if $B_k = D(A_i)$ were to be artificially attached to the mutual giant component. If the link leaving A_i leads to a node A_j that survived the initial attack, x represents the event that A_j is not connected to the mutual giant component via any of its outgoing links, even with our *a priori* assumption. However, even if x does not occur, A_j may still be dead because its interdependent node $B_l = D(A_j)$ does not belong to the mutual giant component. Accordingly, we define the event y that the node B_l is not connected to the mutual giant component via links of network B. Thus, we can write

$$a = (1-p) + p[P(x \cup y)], \tag{3}$$

where $P(x \cup y)$ plays the role that b^{k-1} did in Eq. (2). We need to study separately monomers, "matched" dimers (in which both nodes in the dimer survive the initial attack), and "unmatched" dimers (in which only one of the two nodes survives). Equation (3) is a general framework which must be adapted to account for the three different types of nodes in our system. We thus obtain three equations from Eq. (3). The equation for a_m studies links that leave monomers, while the equation for a_d studies links that leave matched dimers and the equation for a_u studies links that leave unmatched dimers.

We then calculate the probability that A_j (the node to which the link leads) is a given type of node (for example, a monomer). We then multiply this by the probability that if A_j is a monomer (for example), A_j does not connect A_i to the giant component, given our *a priori* assumption. Once we calculate these probabilities for each of the three types of nodes, we add these probabilities, finally arriving at the total probability that a node does not lead to the giant component, given our *a priori* assumption that its support node does. We find

$$\begin{aligned} a_d =& (1-p) + qp\left(a_m^{k-1}\right) + (1-q)p(1-p) \\ &\times \left(a_u^{k-2} + \left[1-a_u^{k-2}\right]a_u^{k-1}\right) + (1-q)p^2\left(a_d^{2k-3}\right), \end{aligned} \tag{4}$$

$$\begin{aligned} a_m =& (1-p) + p(1-p)\left(a_u^{k-2} + \left[1-a_u^{k-2}\right]a_u^{k-1}\right) \\ &+ p^2\left(a_d^{2k-3}\right). \end{aligned} \tag{5}$$

These two equations are relatively simple; $P(x \cup y)$ is either just $P(x)$ or $P(x) + P(y) - P(x \cap y)$. That is, y, failure due to the second network, is either (1) impossible due to the *a priori* assumption, or (2) independent of x. Case 1 occurs when both A_i and A_j are monomers or parts of matched dimers; in these cases, nodes that survived the initial attack connect $D(A_j)$ and $D(A_i)$, allowing $D(A_j)$ to be connected to the giant component. Case 2 occurs when A_j is a part of an unmatched dimer; due to the local treelike structure of the large RR network, the path that connects A_j to its giant component will not overlap with the path that connects $D(A_j)$ to its giant component. Thus, the existence of one path has no correlation with the existence of the other. A complete derivation of Eqs. (4)–(8) follows in Appendix B. The calculation of a_u is more difficult. The events x and y are not always independent because A_j's and B_j's paths to the mutual giant component may overlap. We must find $P(x \cap y) \neq P(x)P(y)$. Therefore, we introduce another two variables, z_m and z_d, which denote the probabilities that neither the link from A_i to A_j nor the link from B_i to B_j lead to the giant component. The principle in these cases is the same as the one used in the determination of a_m and a_d. $P(x \cap y)$ can be expressed as a power of z_m or z_d. We find

$$\begin{aligned} a_u =& (1-p) + qp\left(2a_m^{k-1} - z_m^{k-1}\right) \\ &+ (1-q)p(1-p)\left(a_u^{k-2} + \left[1-a_u^{k-2}\right]a_u^{k-1}\right) \\ &+ (1-q)p^2\left(2a_d^{2k-3} - z_d^{2k-3}\right), \end{aligned} \tag{6}$$

$$\begin{aligned} z_d =& q(1-p) + (1-q)(1-p)^2 + qp\left(z_m^{k-1}\right) \\ &+ 2(1-q)p(1-p)\left(a_u^{k-2} + \left[1-a_u^{k-2}\right]a_u^{k-1}\right) \\ &+ (1-q)p^2\left(z_d^{2k-3}\right), \end{aligned} \tag{7}$$

$$\begin{aligned} z_m =& (1-p)^2 + 2p(1-p)\left(a_u^{k-2} + \left[1-a_u^{k-2}\right]a_u^{k-1}\right) \\ &+ p^2\left(z_d^{2k-3}\right). \end{aligned} \tag{8}$$

Finally, the fraction of nodes in the mutual giant component is

$$\begin{aligned} \mu =& p\left\{1 - m\left(2a_m^k - z_m^k\right) - (1-m)\left[p\left(2a_d^{2k-2} - z_d^{2k-2}\right)\right.\right. \\ &\left.\left. + (1-p)\left(2a_u^{k-1} - a_u^{2k-2}\right)\right]\right\}. \end{aligned} \tag{9}$$

To solve the equations (4)–(8), we define $\vec{\xi} \equiv (a_d, a_u, a_m, z_d, z_m)$, and a function $\vec{\varphi}(\vec{\xi})$, which represents the right-hand sides of Eqs. (4)–(8). These equations always have a trivial solution of $(1,1,1,1,1)$, for which $\mu = 0$. Through an iterative process $\vec{\xi}_0 = \vec{0}$, $\vec{\xi}_n = \vec{\varphi}(\vec{\xi}_{n-1})$, we search for a nontrivial fixed point $\vec{\xi} = \vec{\varphi}(\vec{\xi})$ for different values of p. The results for $\mu(p)$ are presented in Fig. 5. Our results for p_c coincide with our numerical data to within the precision of the simulation. We remarkably see three types of transitions in the value of $\vec{\xi}$ as a function of p, as the simulations indicated. For $k \leqslant 7$, we observe a second-order transition. For $k \geqslant 9$, we find a first-order transition. For $k = 8$, as p decreases, we first see a first-order transition at $p = p_{cI} = 0.2762$, where μ changes dramatically (but not to zero) and a second-order transition to $\mu = 0$ when p decreases to $p_{cII} = 0.2688$.

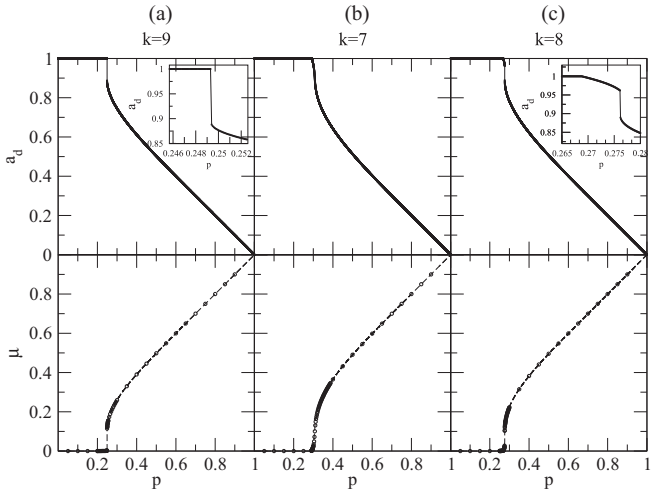

FIG. 5. The fraction of nodes in the mutual giant component $\mu(p)$ for $\ell = 1$ and (a) $k = 9$, (b) $k = 7$, and (c) $k = 8$. Dotted lines in the lower graph represent the fraction of nodes in the mutual giant component (μ) derived from the analytic solution. The circles represent the values obtained in simulation of networks with $N = 10^6$ nodes, which show excellent agreement with theory. The upper graph represents a_d, one of the probabilities defined in Eqs. (4)–(8). The other probabilities behave similarly. Insets show the detailed behavior of $a_d(p)$ near the transition revealing, respectively, a discontinuous increase to 1 characteristic of the first-order transition ($k = 9$), a continuous increase to 1 characteristic of the second-order transition ($k = 7$), and a discontinuous increase to 0.9616 at $p = p_{c1} = 0.2762$ followed by a continuous increase to 1 at $p = p_{c11} = 0.2688$, indicating the existence of two transitions ($k = 8$). In all cases, $a_d = 1$ indicates the complete destruction of the mutual giant component, as shown in the lower graph.

IV. CONCLUSION

In conclusion, our results demonstrate an interesting behavior of collapsing random networks with distance-restricted dependency links. We find that networks with long dependency distances are much more vulnerable than networks with short dependency distances. Moreover, the networks with large dependency distance and large degree collapse via an abrupt first-order transition, near which a removal of a single additional node can create a collapse of the entire system, while the networks with short dependency distances and low degree gradually disintegrate via a continuous second-order transition without catastrophic cascades, which make such networks safer. For the transitional case, we have found a two-stage collapse. The first-order transition is sudden, as expected, but the network survives until the second transition, which completely destroys the networks. In contrast to interdependent two-dimensional lattices [14], for which the critical dependency distance associated with the change of the transition order from the second to the first coincides with the distance of maximal vulnerability, for interdependent random networks of any degree k, the vulnerability always monotonically increases with the dependency distance. This difference is caused by the fact that for large dependency distances in two dimensions, the phase boundary between the destroyed and intact phases must be created in order for the cascade of failures to spread over the system. In random networks, which are effectively infinitely dimensional, the boundary between phases is not well defined and hence the cost of its formation for large dependency distances does not affect the system behavior.

As we mention above, all the results are obtained for a model in which the dependency isomorphism was created by the "parking algorithm." These results obtained by an exact analytical method and verified by extensive simulations can serve as a benchmark for more complex cases. We compare these results with the results produced by other algorithms which incorporate the creation of long loops in the dependency permutations (Appendix C). We find that in the presence of loops, the critical fraction of nodes p_c decreases by no more than 5%, with the exact value depending on the algorithm used. However, the presence of loops makes the transition more abrupt, so that for $\ell = 1$ the transitions become first order for smaller values of k, with the exact value depending on the algorithm.

ACKNOWLEDGMENTS

We wish to thank DTRA for financial support and Shlomo Havlin for stimulating discussions. We acknowledge the partial support of this research through the Dr. Bernard W. Gamson Computational Science Center at Yeshiva College.

APPENDIX A: DERIVATION OF PARKING CONSTANT

In our process, we combine neighboring nodes into dimers or assign single nodes to be monomers. Let t represent the time. It will increase by 1 whenever we select a new node, but not when we select the second node of a dimer. At the end of the assignment process, $t_{\text{final}} = N(m + d)$, where Nm is the number of monomers and Nd is the number of dimers. The number of nodes in the dimers is $2dN$; thus, $m + 2d = 1$.

Let $E(t)$ represent the number of links that leave unassigned nodes at time t, and $F(t)$ represent the number of links that connect one unassigned node with one assigned node at time t. Then

$$A(t) \equiv \frac{F(t)}{E(t)} \tag{A1}$$

is the probability that a link leaving an unassigned node will arrive at an assigned node.

Since $E(t)$ is simply the number of unassigned nodes multiplied by k,

$$\frac{\Delta E(t)}{\Delta t} = -k\{1 + [1 - A(t)^k]\}. \tag{A2}$$

The first term in the braces represents the decrease in unassigned nodes associated with the node originally chosen for assignment, while the second term is the additional decrease when a dimer is formed, which occurs with probability $1 - A(t)^k$.

To calculate the change in $F(t)$, we will first consider the effect of a single link that leaves the node that we are considering. This link will, with probability $1 - A(t)$, reach an unassigned node. If it does, $F(t)$ will increase by 1. If the link leads to an already assigned node [which will occur with probability $A(t)$], $F(t)$ will decrease by 1, because the link

will no longer connect an unassigned node with an assigned one. Adding the two effects, we see that each link contributes $1-2A(t)$ to $F(t)$. There is a $1-A(t)^k$ probability that the assigned node will form a dimer, in which case we examine the effect of the second node of this dimer. This calculation is nearly the same as for the first node of the dimer. However, in this case, only $k-1$ links must be examined, because one link leads to the first node of the dimer. Additionally, the choosing of the second node causes $F(t)$ to decrease by 1, since the link connecting the two nodes in the dimer no longer connects an unassigned node with an assigned node. Considering both possibilities, we arrive at the equation

$$\frac{\Delta F(t)}{\Delta t} = k[1-2A(t)] + [1-A(t)^k]\{[k-1][1-2A(t)]-1\}. \tag{A3}$$

We can redefine the variables $F(t) \equiv F(t)/N$, $E(t) \equiv E(t)/N$, and $t \equiv t/N$ so that we can take the continuous limit as $N \to \infty$.

Recalling Eq. (A1),

$$\frac{dA}{dt} = \frac{\frac{dF}{dt} - \frac{dE}{dt}A}{E}. \tag{A4}$$

Differentiating by $E(t)$ instead, we arrive at

$$\frac{dA}{dE} = \frac{\frac{dF}{dE}}{E} - \frac{A}{E}. \tag{A5}$$

Integrating Eq. (A5), we, after some algebra, arrive at

$$E(A) = k(1-A) \times \exp\left(2\int_0^A \frac{B^k - 1}{[B^k(k-2) - 2(k-1)](B-1)} dB\right). \tag{A6}$$

Equation (A2) can be restated as

$$\frac{dt}{dA} = \frac{dE}{dA}k(A^k - 2). \tag{A7}$$

At the very end of the assignment process, $A = \frac{F}{E} \to 1$. Differentiating Eq. (A6) with respect to A, substituting dE/dA into Eq. (A7), and integrating it over the entire process, i.e., from $A = 0$ to $A = 1$, we obtain

$$t_{\text{final}} = \int_0^1 \frac{-k}{A^k(k-2) - 2(k-1)} \times \exp\left(2\int_0^A \frac{(B^k - 1)dB}{[B^k(k-2) - 2(k-1)](B-1)}\right) dA. \tag{A8}$$

Recalling that $m + d = t_{\text{final}}$ and $m + 2d = 1$, m, as found in Eq. (1), can trivially be obtained as $2t_{\text{final}} - 1$. The resulting numerical values $m(k)$ are presented in Fig. 6.

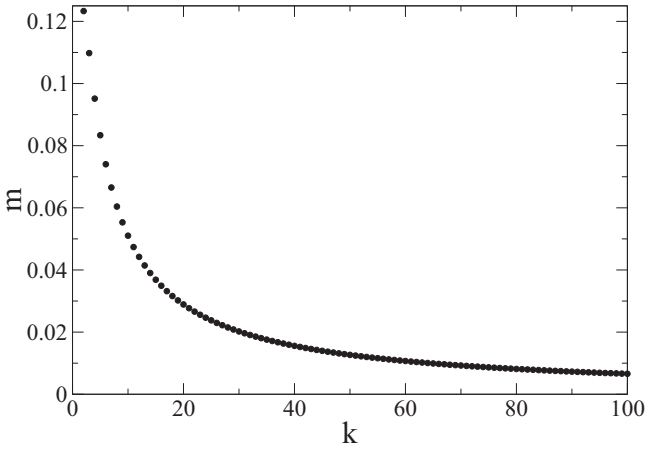

FIG. 6. The fraction of monomers as a function of the network degree k obtained by numerical integration of Eq. (1).

APPENDIX B: DERIVATION OF PERCOLATION EQUATIONS

In our calculation of a, we examine a link that leads from $A_i \to A_j$ and calculate the probability that the link does not lead to a giant component, given our *a priori* assumption. The link $i \to j$ can lead to a dimer, an unmatched dimer, or a monomer. We will denote these cases by referring to the nodes involved; for example, md is associated with the case in which a link goes from a monomer to a matched dimer. For the case md we define an event, also denoted by md, that a link leading from a monomer to a matched dimer does not lead to the giant component, and denote the probability of this event $P(md)$. Similarly, $P(mdx)$ is the probability of event md occurring due to x, isolation in the A network. Analogous definitions apply for the other seven cases: mu, dm, dd, du, um, ud, and uu, some of which are illustrated by diagrams in Fig. 7. Note that the case mm does not exist because in our model, a monomer cannot be connected to another monomer. The probability of cases dm and um must include the factor q, while the cases dd, du, ud, and uu must include the factor $1-q$.

We first develop equations for a_d and a_m. The link could lead to a node A_j that is any of our three types, or to a node that was destroyed in the initial attack. We express $P(x \cup y)$ in each case in terms of A_j's respective a and multiply this term by that type's specific probability. Let us begin with the cases md,dm, and dd. Consider the link that leads from node A_i to node A_j. The counterpart of this link in network B leads from node B_i to node B_j, where B_j, as part of a matched dimer or monomer, also survives the initial attack. In the case md, $B_i = D(A_i)$ belongs to the mutual giant component. Thus, B_j, by virtue of being linked to B_i, also belongs to the mutual giant component. Since $d = 1$, $D(A_j) \equiv B_l$ is an immediate neighbor of B_j, and having survived the initial attack, is also connected to the mutual giant component in network B.

In cases dm and dd, $B_k \equiv D(A_i)$ belongs to the mutual giant component. Since $d = 1$, A_i and A_k must be immediate neighbors, so B_k is connected to B_i. B_i is connected to B_j, and hence B_j belongs to the mutual giant component. In case dd, $B_l \equiv D(A_j)$ is linked to B_j and hence also belongs to the mutual giant component. Thus, in md,dm, and dd, $y = \emptyset$ and hence $P(x \cup y) = P(x)$. Accordingly, the contributions for a_m and a_d from those three cases are

$$P(dm) = a_m^{k-1}, \tag{B1}$$

$$P(dd) = P(md) = a_d^{2k-3}, \tag{B2}$$

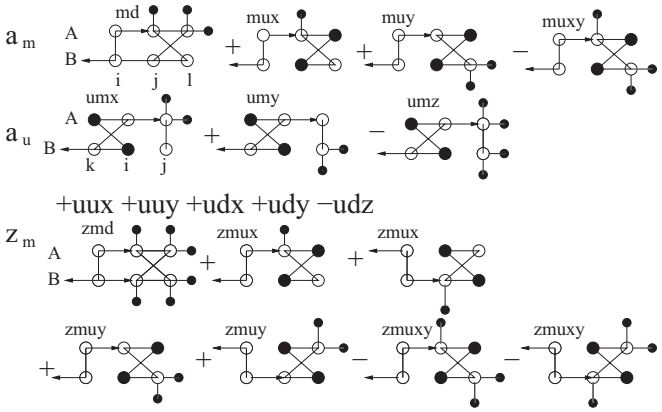

FIG. 7. Diagrams presenting various terms in Eqs. (5), (6), and (8). The corresponding terms is Eqs. (4) and (7) follow the same patterns. The top row in each diagram represents network A; the bottom row represents network B. The nodes are shown by circles. The white circles are the nodes which survive the initial attack; the black circles are the dead nodes which do not survive the initial attack. Connectivity links are represented by horizontal lines; vertical and diagonal lines denote dependency links. The link under examination is the one connecting A_i and A_j; unmarked nodes follow the previously established indices. In z_m, we explore both networks. The arrows that point left indicate that $D(A_i)$ is assumed *a priori* to belong to the mutual giant component. Small black circles at the end of the outgoing links identify links that do not lead to the mutual giant component; the absence of such circles indicates that connection of this node to the mutual giant component via its outgoing links is irrelevant. The number of such circles helps identify the power of the terms in a particular case of $k = 3$.

where the powers of a_m and a_d are, respectively, the numbers of outgoing links in network A leaving the monomers and matched dimers. The fact that these links do not lead to the mutual giant component is depicted by a small filled circle at the end of these links in Fig. 7.

In contrast, in the cases du and mu, a selected link arrives at a surviving node of an unmatched dimer A_j. Thus B_j is dead, as is $A_l \equiv D(B_j)$. Therefore, the node $B_l = D(A_j)$ is not connected to the node $B_k = D(A_i)$ by the links shown on the diagram. Thus, if all of its survived links do not lead to the mutual giant component, it will not belong to the mutual giant component. The event that $B_l = D(A_j)$ is connected to the mutual giant component (that is, that y does not occur) is independent of the event that A_j is connected to the mutual giant component via its outgoing links (that is, that x does not occur). This independence stems from the fact that the node B_l may be connected to the giant component by paths which are totally independent of the paths of the outgoing links of network A_j. This is because locally, a large randomly connected network has a tree structure, so the paths leading to the mutual giant component can meet only after infinitely many steps. Furthermore, the nodes on these paths connecting A_j and $B_l = D(A_j)$ to the giant component cannot be interdependent due to the short range of the dependency links. Thus, $P(x \cap y) = P(x) + P(y) - P(x)P(y)$. In Fig. 7, the first term $P(x)$ corresponds to the diagram mux, while the second term $P(y)$ corresponds to the diagram muy and the third term $P(x)P(y)$ corresponds to the diagram $muxy$. Note that node B_l does not belong to the mutual giant component if and only if all of its $k-1$ survived links in network B are not connected to the mutual giant component, because its support node A_j belongs to the mutual giant component due our *a priori* assumption. Accordingly,

$$P(mu) = P(du) = a_u^{k-2} + \left(1 - a_u^{k-2}\right)a_u^{k-1}. \tag{B3}$$

Combining all these terms, we find, equivalently to Eq. (5),

$$a_m = (1-p) + p\left[(1-p)P(mu) + pP(md)\right], \tag{B4}$$

and [equivalently to Eq. (4)],

$$a_d = (1-p) + p\{qP(dm) + (1-q)[(1-p)P(du) + pP(dd)]\}. \tag{B5}$$

The calculation of a_u is more difficult. In cases ud and um, the events x and y are not independent because the topologically correspondent links $A_j \to A_n$ and $B_j \to B_n$, which connect corresponding pairs of nodes in networks A and B, are not independent. Their paths to the mutual giant component will involve many of the same obstacles, due to the similarity of the two networks. Thus, the event that A_j is connected to the giant component is not independent of the event that B_j is connected to the giant component. Therefore, we need to find $P(x \cap y) \neq P(x)P(y)$. In the case um, this event occurs when all of the outgoing links of A_j and all links of B_j do not lead to the giant component. In ud, we must add the links outgoing from nodes A_l and B_l. We can group these links in the two networks in pairs of topologically correspondent links, none of which should lead to the giant component. The number of such pairs is $k-1$ in um and $2k-3$ in ud. Therefore, we introduce another two variables, z_m and z_d, which denote the probabilities that both links in the pair of correspondent links in a monomer or a matched dimer, respectively, do not lead to the giant component. As we did previously, we will consider the links going from A_i and B_i to A_j and B_j. As an illustration, the terms included in z_m are shown in Fig. 7. Then

$$P(um) = P(umx) + P(umy) - P(umz) = 2a_m^{k-1} - z_m^{k-1}, \tag{B6}$$

$$P(ud) = P(udx) + P(udy) - P(udz) = 2a_d^{2k-3} - z_d^{2k-3}. \tag{B7}$$

The case uu is totally analogous to the case du:

$$P(uu) = a_u^{k-2} + \left(1 - a_u^{k-2}\right)a_u^{k-1}. \tag{B8}$$

Combining these three cases, we reproduce Eq. (6):

$$a_u = (1-p) + p\{qP(um) + (1-q)[(1-p)P(uu) + pP(ud)]\}. \tag{B9}$$

The calculation of z_m and z_d is analogous to the calculation of a_m and a_u except that the cases zdu and zmu must be taken twice because now we are interested in cases both when A_j survives the initial attack and B_j is dead and vice versa. For this reason, we must also replace the initial $1-p$ term in Eqs. (5) and (4) with the term $(1-p)^2$, because when calculating z, we require that both nodes A_j and $A_l \equiv D^{-1}(B_j)$ must be dead in order for events zmu and zdu to not occur. Note that event zmu is identical to event mu and event zdu is identical to event du. In this case, because the link leads to an unmatched dimer,

one of its nodes did not survive the original attack, allowing us to examine only the links leaving the other node. Thus,

$$P(zdm) = z_m^{k-1}, \tag{B10}$$

$$P(zdd) = P(zmd) = z_d^{2k-3}, \tag{B11}$$

and [Eqs. (7) and (8)]

$$z_m = (1-p)^2 + 2p(1-p)P(mu) + p^2 P(zmd), \tag{B12}$$

$$z_d = q[1-p+pP(zdm)] + (1-q)[(1-p)^2 + 2p(1-p)P(du) + p^2 P(zdd)]. \tag{B13}$$

APPENDIX C: OTHER DEPENDENCY ISOMORPHISMS

To investigate how different ways of assigning the dependency links can affect the results, we perform simulations in several test cases using other algorithms of establishing the dependency isomorphism. It is evident that establishing a dependency isomorphism is equivalent to a permutation of node indices that satisfies the distance restriction. The parking algorithm employed in this paper creates permutations with maximal cycle loops of length 2, corresponding to "dimers." Some fraction of indices, m, is left unchanged in the permutation created by the parking algorithm. These indices correspond to "monomers." If the maximal distance $\ell \geqslant 2$, the transposition algorithm employed by Li *et al*. [14] can produce permutations with loops of considerable length but these loops still do not follow the large loops in the biconnected components ("blobs") of the networks. We find that the p_cs obtained with the transposition algorithm differ from the values obtained with the parking algorithm by no more than 5% and that the change in algorithm does not change the order of the transition.

If $\ell = 1$, the transposition algorithm fails to produce permutation cycles longer than 2. Therefore, we employ the Hopcroft-Tarjan algorithm [15] to identify blobs and singly connected links ("red bonds") of the networks A and B (which are identical due to the topological isomorphism of the two networks). In this algorithm, we select nodes at random one-by-one, as we do in the parking algorithm, using the following four steps. Step I: A node is randomly selected among the nodes with yet unpermuted indices. If and only if all the nearest neighbors of this node are already permuted, the node becomes a "monomer." Step II: Otherwise, we randomly select one of the neighbors. If and only if the link connecting these nodes is a red bond, we establish a dependency between these nodes, creating a "dimer." Step III: Otherwise, we produce a random walk on a subnetwork of nodes with yet unpermuted indices, such that the walk can use only links of the same blob, but cannot use the same link twice. Once this walk forms a loop, we select the indices of the nodes of this loop into a permutation cycle. If some of the originally selected nodes do not belong to the loop, their indices will not be permuted at this point. Step IV: If the random walk is trapped among the nodes with permuted indices, we recalculate the blobs in the subnetwork of nodes with unpermuted indices by the Hopcroft-Tarjan algorithm, staring from the originally selected node, and repeat steps II and III.

This method creates permutations with very long loops. The distribution of these loop lengths t obeys a universal scaling law $P(t > x) = f[x/\langle t(N)\rangle]$, where $f(x)$ decreases slightly faster than exponentially and $\langle t(N)\rangle$ is the average loop length, which scales as $\langle t(N)\rangle = f_t(k)\sqrt{N}$, where $f_t(k)$ is a decreasing function of k, which converges to a positive value for $k \to \infty$. This scaling law follows from the analogy with the birthday paradox [20].

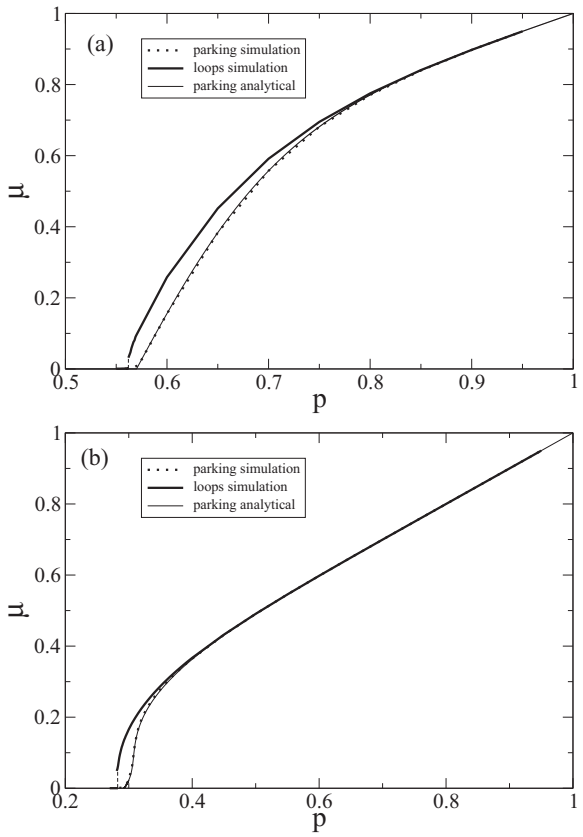

FIG. 8. Comparison of the behavior of the fraction of nodes in the mutual percolation component vs p for two different algorithms of assigning dependency links for $k = 3$ (a) and $k = 7$ (b). The thin solid line shows analytical results for the parking algorithm, while the bold line shows the results for the loop algorithm described in Appendix C. Dots show simulations results for parking algorithm, which demonstrate perfect agreement with theory. One can see that for $k = 7$, $p_c = 0.2925$ for the parking algorithm, while for the loop algorithm, $p_c = 0.2825$. However, in the presence of loops, μ changes abruptly from 0.048 for $p > p_c$ to 0 for $p < p_c$. For $k = 3$, p_c decreases from $p = 0.5700$ for the parking algorithm to $p = 0.562$ for the loop algorithm.

Figure 8 compares the behavior of the fraction of nodes in the mutual giant component for μ versus p for the parking algorithm and the loop algorithm described above. One can see that in the presence of loops p_c slightly decreases for all k, but the transition becomes sharper and becomes a discontinuous first-order transition for any k. Alternative algorithms of establishing dependency links with long permutation cycles produce similar results, but the change of the order of the transition from second to first takes place for $3 < k < 8$, depending on the algorithm. In general, the presence of loops in the permutation describing the dependency isomorphism makes interdependent networks slightly less vulnerable.

[1] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, Nature (London) **464**, 1025 (2010).
[2] Y. Hu, B. Ksherim, R. Cohen, and S. Havlin, Phys. Rev. E **84**, 066116 (2011).
[3] R. Parshani, S. V. Buldyrev, and S. Havlin, Proc. Natl. Acad. Sci. USA **108**, 1007 (2010).
[4] J. Shao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, Phys. Rev. E **83**, 036116 (2011).
[5] S. V. Buldyrev, N. W. Shere, and G. A. Cwilich, Phys. Rev. E **83**, 016112 (2011).
[6] X. Huang, J. Gao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, Phys. Rev. E **83**, 065101 (2011).
[7] G. Dong, J. Gao, L. Tian, R. Du, and Y. He, Phys. Rev. E **85**, 016112 (2012).
[8] G. Dong, L. Tian, D. Zhou, R. Du, J. Xiao, and H. E. Stanley, Europhys. Lett. **102**, 68004 (2013).
[9] J. Gao, S. V. Buldyrev, S. Havlin, and H. E. Stanley, Phys. Rev. E **85**, 066134 (2012).
[10] S. W. Son, P. Grassberger, and M. Paczuski, Phys. Rev. Lett. **107**, 195702 (2011).
[11] S. W. Son, G. Bizhani, C. Christensen, P. Grassberger, and M. Paczuski, Europhys. Lett. **97**, 16006 (2012).
[12] G. J. Baxter, S. N. Dorogovtsev, A. V. Goltsev, and J. F. F. Mendes, Phys. Rev. Lett. **109**, 248701 (2012).
[13] T. P. Peixoto and S. Bornholdt, Phys. Rev. Lett. **109**, 118703 (2012).
[14] W. Li, A. Bashan, S. V. Buldyrev, H. E. Stanley, and S. Havlin, Phys. Rev. Lett. **108**, 228702 (2012).
[15] J. Hopcroft and R. Tarjan, Commun. ACM **16**, 372 (1973).
[16] M. E. J. Newman, *Networks: An Introduction* (Oxford University Press, Oxford, 2010).
[17] R. Cohen and S. Havlin, *Complex Networks: Structure, Robustness, and Function* (Cambridge University Press, Cambridge, 2010).
[18] A. Rényi, Publ. Math. Inst. Hung. Acad. Sci. **3**, 109 (1958).
[19] H. G. Dehling, S. R. Fleurke, and C. Külske, J. Stat. Phys. **133**, 151 (2008).
[20] W. Feller, *An Introduction to Probability Theory and Its Applications*, Vol. 1 (John Willey & Sons, New York, 1968).