A Framework for Simulating Multiple Contagions Over Multiple Networks

Aparna Kishore[1], Lucas Machi[1],
Chris J. Kuhlman[1], Dustin Machi[1], and S. S. Ravi[1]

University of Virginia, Charlottesville VA 22904, USA,
appu88@gmail.com, lhm4v@virginia.edu, hugo3751@gmail.com,
dm8qs@virginia.edu, ssr6nh@virginia.edu

Abstract. Many contagion processes evolving on populations do so simultaneously, interacting over time. Examples are co-evolution of human social processes and diseases, such as the uptake of mask wearing and disease spreading. Commensurately, multi-contagion agent-based simulations (ABSs) that represent populations as networks in order to capture interactions between pairs of nodes are becoming more popular. In this work, we present a new ABS system that simulates any number of contagions co-evolving on any number of networked populations. Individual (interacting) contagion models and individual networks are specified, and the system computes multi-contagion dynamics over time. This is a significant improvement over simulation frameworks that require union graphs to handle multiple networks, and/or additional code to orchestrate the computations of multiple contagions. We provide a formal model for the simulation system, an overview of the software, and case studies that illustrate applications of interacting contagions.

Keywords: interacting contagions, network discrete dynamical systems, opinion dynamics, multi-contagion agent-based simulation systems

1 Introduction

1.1 Background and Motivation

Many contagion processes evolving on populations do so simultaneously, interacting over time. Examples are co-evolution of social processes and diseases, such as increased hand washing during the influenza (flu) season and mask wearing during pandemic outbreaks, e.g., [9]; and co-transmission of information and evacuation decisions during natural disasters [29]. Commensurately, multi-contagion agent-based simulations (ABSs) that represent populations as networks in order to capture interactions between pairs of nodes (e.g., people are represented as nodes and interactions as edges in a network) are becoming more popular.

Currently, several simulation systems can compute interacting contagion dynamics, but they do so by incorporating data that must be preprocessed for a particular simulation, or additional software must be written. For example,

consider each of two interacting contagions, c_1 and c_2, spreading on separate networks, G_1 and G_2, respectively. Generally, there are common nodes and/or edges in the graphs so that there is interaction between contagions. Simulators are typically set up to store one graph, so that they require that a union graph G' be formed ($G' = G_1 \cup G_2$) where labels on edges designate whether an edge is in G_1, G_2, or both. Graph nodes are handled analogously. Consequently, if there are three graphs and simulations need to be performed with all combinations of two graphs, then three union graphs need be generated. With n_g of distinct graphs and n_{inst} graphs used in each simulation, the number of required union graphs is the binomial coefficient $C(n_g, n_{inst})$, which can be large.

Also, software must be added to frameworks to manage contagions. As an example for discrete time simulation, one might execute one contagion at each time step, so that if there are n_c contagions and t_{max} simulation time steps (e.g., days), a simulator might execute $n_c \times t_{max}$ time steps. But this approach adversely affects down-stream post-processing because "time" must be manipulated accordingly. For other frameworks, code has to be written to coordinate the execution of contagions at each time (see Section 1.3). Our system obviates the need for these changes.

We present a new ABS framework for modeling the spread of any number of contagions on any number of networks, using a structured and principled approach in both the software system and its use in applications. We also provide a formal model for our simulation system and case studies to illustrate its use in applications.

1.2 Our Contributions

1. Multi-Contagion Graph Dynamical Systems Formalism. We start with a graph dynamical systems (GDS) formalism [1] for single-contagion discrete dynamical systems. We extend that formalism to a *multiple contagion* GDS, MCGDS, operating over multiple graph instances. Any number $n_c \geq 1$ of contagions can operate on any number n_g of graphs, where $1 \leq n_g \leq n_c$, because while each contagion spreads on one graph, two or more contagions may spread on a single graph. The formal model is provided in Section 2.

2. Parallel Implementation of Multi-Contagion Simulation Framework. The software framework, called CSonNet (Contagion Simulation ON NETworks) is written in Python. Our simulation system implements concurrency through multi-processing. The system can simulate any number $n_c \geq 1$ of contagions operating over any number n_g of graphs, where $1 \leq n_g \leq n_c$. The graph instances, in general, can be disjoint, the same graph, or have common subsets of nodes and edges. Contagions can evolve independently or can interact as they evolve. Contagion models (e.g., SIR epidemic models, various threshold models for social contagions) are added to the system in one of two ways: (i) Python code that conforms to a specified interface is written and added to the system, or (ii) a user composes models through input files, using a set of rules provided by the system. Option (i) is the only case where new code is written; all other inputs (e.g., specification of graphs, the mapping of contagions to the networks

on which they evolve, rule-based models) are given through input files for a simulation. The system is overviewed in Section 3 and its strong scaling performance is evaluated in Section 4.

3. Case Studies. We provide two case studies, each with two non-interacting and two interacting contagion simulations: (*i*) SIR_1-SIR_2 epidemic contagions, and (*ii*) Threshold-SIR mixed social and epidemic contagions. Social networks with up to 75,877 nodes are used. While the system can compute contagions on much larger graphs, we choose graphs where fairly demanding multi-contagion simulations can be completed in less than 300 seconds of computational time, since in practice parametric simulation studies over large parameter spaces are typically required. These illustrate several features of CSonNet (see Section 5).

This simulation system significantly extends the single-contagion system described in [23]; that system does not implement the MCGDS of Contribution 1. Our implementation here (Contribution 2) is a large extension of the previous system in terms of capabilities (e.g., software added and software modified). None of the case studies here (Contribution 3) could be executed with the system in [23] without the workarounds specified in Section 1.1.

1.3 Related Work

Multiple contagion simulation systems. To the best of our knowledge, no simulator has multi-contagion capabilities "out of the box" as is the case for our framework. However, because several frameworks enable models to be added to them (as does ours), it is possible to use these simulators for multiple-contagion scenarios, as described in Section 1.1. But these workarounds come with the price of requiring additional code and/or requiring more preprocessing of graphs before running simulations. Among the frameworks for simulations on networks that fit this description, across a range of capabilities, are: NetLogo [24], NDLIB [25], MASON [15], and Repast HPC [10].

Studies of multiple contagion systems. Simulations of information spreading and evacuation decision-making in the context of hurricane evacuation modeling are discussed in [29]. Simulations of agents with limited attention spans for which multiple contagions (e.g., ideas) must compete are discussed in [28]. Agent-based models for competing languages are summarized in [21].

A number of papers have addressed models for competing and/or cooperating contagions (such as epidemics, product information, and misinformation in social media) over networks (see e.g., [17, 19, 28]). Polarization and consensus are studied with competing contagions [27]. One contagion that overtakes and halts a second contagion is studied in [7, 20]. A model for a simple contagion producing a complex contagion is given in [17]. Game theory is used to analyze competing contagions in [11]. Many publications have studied optimization problems (e.g., seeding methods for influence maximization, minimizing the spread of contagions) in the context of multiple contagions (see e.g., [6, 7]).

2 Multi-Contagion Graph Dynamical System

The underlying model implemented in our simulation system CSonNet is called a **graph dynamical system** (GDS). The formalism [1, 18] addresses a single contagion, but naturally extends to multiple contagions, which is done here. A GDS can simulate Turing machines for specific complexity classes [1, 4, 5].

2.1 Multiple Contagion GDS Formalism

A **multi-contagion GDS**, MCGDS, incorporating $n_c \geq 1$ contagions, is a quadruple $\mathsf{S}(\mathsf{G}^c, \mathsf{F}^c, \mathsf{K}^c, \mathsf{R}^c)$: (i) G^c: a sequence of graphs $G^j(V^j, E^j)$, where G^j is the graph on which the contagion c_j propagates, V^j and E^j denote its vertex (i.e., node) and edge sets, $1 \leq j \leq n_c$; (ii) F^c: a sequence of sequences F^j, where each F^j is the sequence of local functions for contagion c_j; (iii) K^c: a sequence of vertex (node) state sets K^j, where each K^j is the set of admissible vertex states for contagion c_j; and (iv) R^c: a sequence of specifications R^j, where each R^j is the order in which local functions are executed for c_j. Each of these elements is defined below.

The graph $G^j(V^j, E^j)$ represents the **interaction network** for c_j. Let $n_j = |V^j|$ and $\mu_j = |E^j|$, with $n = |V^1 \cup V^2 \cup \ldots \cup V^{n_c}|$ and $\mu = |E^1 \cup E^2 \cup \ldots \cup E^{n_c}|$. Agents are vertices and pairwise agent interactions are edges in G^j. In general, edges may be directed or undirected. In this paper, for simplicity, we will consider graphs with undirected edges. Vertex (resp., edge) sets across contagions may be the same, disjoint, or have some common subset of nodes (resp., edges).

Each agent $v_i \in V^1 \cup \ldots \cup V^{n_c}$ has an **agent state** or **vertex state** s_i which is a sequence $s_i = (s_{i,c_1}, s_{i,c_2}, \ldots, s_{i,c_{n_c}})$ of n_c elements. Each $s_{i,c_j} \in K^j$. The **system state** (also called a **configuration**) $s = (s_1, s_2, \ldots, s_n)$ is the sequence of n vertex states. Let the **local states**, denoted by $s[v_i]$, represent the sequence of states of vertex v_i and all of its distance-1 neighbors in each G^j. That is, $s[v_i] = (s_{c_1}[v_i], s_{c_2}[v_i], \ldots, s_{c_{n_c}}[v_i])$, where $s_{c_j}[v_i]$ is the sequence of length $d_j(v_i) + 1$ of the states of v_i and each of its distance-1 neighbors in G^j for contagion c_j, and $d_j(v_i)$ is the degree of v_i in G^j. These quantities, at time t, are denoted by s_i^t, s_{i,c_j}^t, s^t, and $s^t[v_i]$ respectively.

Each agent v_i has a **local function** $f_{i,c_j} \in F^j$ that determines its state transitions for contagion c_j. The state of agent v_i at time $t+1$ for c_j is given by

$$s_{i,c_j}^{t+1} = f_{i,c_j}(s^t[v_i]) \quad \text{for each } 1 \leq j \leq n_c \,. \tag{1}$$

Thus, the next state of v_i, with respect to each contagion c_j is a function of the current state of v_i and those of its distance-1 neighbors in *every* G^j over *all* contagions. That is, the argument on the right hand side of Equation (1) is the same for each f_{i,c_j}, for a fixed v_i. This expression—and specifically the local functions for each contagion—indicates explicitly how each contagion affects (i.e., interacts with) every other contagion.

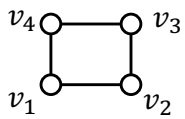
We assume that specification of **update order** R^j for the local functions for c_j corresponds to the **synchronous** update scheme. That is, all agents v_i

evaluate their local functions f_{i,c_j}, $1 \leq i \leq n$, for c_j and update their states s_{i,c_j}^{t+1} *simultaneously* at each time step. Furthermore, we assume the updates across contagions are also done in parallel, i.e., all R^j are parallel with each other. Hence, f_{i,c_j} are computed in parallel for all $1 \leq i \leq n$ and for all $1 \leq j \leq n_c$ in Equation (1). This enables greater parallelization of simulation computations, leading to more efficient calculations. However, other update orders R^j can be used. One example is a sequence W of the node IDs in G^j (of length n_j) and the local function for each node is computed in the order specified by W.

2.2 Example MCGDS

Two contagions propagate on the single network at the left in Figure 1, where nodes are people and edges are interactions. The two contagions are: a social contagion c_1 on mask wearing and a disease contagion c_2. Contagion c_1 uses a threshold model [12], with $K^1 = \mathbb{B} = \{0, 1\}$, where state 0 (resp., state 1) means that a person (node) is not (resp., is) wearing a mask. The local function f_{i,c_1} for all $i \in \{1, 2, 3, 4\}$ and c_1 is as follows. A node v_i changes from state 0 to 1 if at least a threshold $\theta = 1$ number of its neighbors are in state 1. Contagion c_2 is a disease model with states $K^2 = \{S, I, R\}$, where the states are susceptible (S), infectious (I), and recovered (R). The local function f_{i,c_2} for all $i \in \{1, 2, 3, 4\}$ and c_2 is as follows. If v_i is in state S, then v_i changes to state I with probability $p = p_{base} \cdot m_I \cdot m_S$ for each neighbor in state I, where $p_{base} = 0.008$, $m_I = 0.2$ if the infected neighbor is wearing a mask and $m_I = 1$ otherwise, and $m_S = 0.2$ if v_i is wearing a mask and $m_S = 1$ otherwise. If v_i is in state I and transitioned to state I at time t_I, then it transitions to state R at time $t = t_I + t_{inf}$, where $t_{inf} = 3$ is node v_i's infectious duration. If v_i is in state R, it remains in state R. Contagion c_1 affects c_2 in that people that are wearing masks have a lesser probability of contracting and transmitting the disease.

The system states s at $t = 0$, 1, and 2 are given in Figure 1. Contagion c_1 states are under "Social" and c_2 states are under "SIR." At $t = 1$, c_1 spreads to each of v_2 and v_4 because these latter two nodes have v_1 as a neighbor, $s_{1,1}^0 = 1$, and the threshold for all nodes is $\theta = 1$. So at the end of $t = 1$, three of the four nodes are wearing masks. Also at $t = 1$, for contagion c_2, v_3 is initially infected, but the Bernoulli trials for nodes v_2 and v_4 do not result in contagion spread, so the states remain $s_{2,2}^1 = s_{4,2}^1 = S$. (Note that at $t = 1$ for c_2 and node v_3, $t < t_I + t_{inf} = 0 + 3$, so f_{3,c_2} returns the next state as $s_{3,c_2} = I$, which is the current state. The local function for v_3 will return the next state as I until $t = 3$, at which time $s_{3,c_2} = R$.) At $t = 2$ and for c_1, v_3 changes to $s_{3,1}^2 = 1$ due to simple contagion spread from v_2 and v_4. For c_2 at $t = 2$, the edge probability $p = p_{base} \cdot m_I \cdot m_S = (0.008)(1)(0.2) = 0.0016$ for the edge from v_3 to v_2 and from v_3 to v_4 because v_2 and v_4 are wearing masks at the end of $t = 1$ but v_3 is not. The random number in [0, 1] generated for the first edge is $0.0013 < p = 0.0016$ (i.e., the Bernoulli trial is successful) and so v_3 infects v_2 and $s_{2,2}^2 = I$. However, the Bernoulli trial for v_3 to infect v_4 is not successful because the drawn random number is $0.732 > p$, so $s_{4,2}^2 = S$ and therefore the states at $t = 2$ are as shown in Figure 1. A case study using this two-contagion model is given in Section 5.1.



time t=0			time t=1			time t=2		
Node	Social	SIR	Node	Social	SIR	Node	Social	SIR
v_1	1	S	v_1	1	S	v_1	1	S
v_2	0	S	v_2	1	S	v_2	1	I
v_3	0	I	v_3	0	I	v_3	1	I
v_4	0	S	v_4	1	S	v_4	1	S

Fig. 1: Illustrative two-contagion dynamics for a MCGDS. Contagion c_1 is a social contagion of mask wearing represented by a threshold model; contagion c_2 is a disease contagion represented by an SIR model. Both contagions spread on the network on the left. Contagion c_1 affects c_2. States for each contagion, for all nodes, are provided for three time steps.

3 CSonNet Modeling and Simulation Software System

3.1 Overview of Simulation Steps

CSonNet is a discrete-time multi-contagion ABS framework for networked populations. CSonNet implements the MCGDS model of Section 2 in the form of simulations. A **simulation** begins with reading in from file: (i) all graph instances G^{c}; (ii) all local functions F^{c} over all nodes v_i, $1 \leq i \leq n$, and all contagions c_j, $1 \leq j \leq n_c$; (iii) the mapping $\mathsf{M}_{\mathsf{j},\ell}$ of contagion c_j to graph G^ℓ; (iv) the initial state assignments $\mathbb{I}$ to all v_i for each c_j; (v) the number n_i of iterations (i.e., simulation instances) to run; (vi) the maximum number t_{max} of time steps to run per iteration; and (vii) the number n_{wp} of worker processes that perform the computations. The number n_g of graphs and number n_c of contagions are determined from G^{c} and F^{c}, respectively. In particular, the number and types of contagions are completely specified by the local functions f_{i,c_j} of Equation (1) in the sequence F^{c} and K^{c}. The local functions are not entered as equations in input files, but rather as models $M^{st}_{i,j}$, that implement these local functions for v_i and c_j. Section 3.2 below provides an example.

After reading all inputs, the main process of the simulation instantiates n_{wp} worker processes that carry out iterations in parallel on multi-core hardware computing nodes. The main process provides to each worker process the appropriate graphs, local functions, mapping of contagion to graph, initial conditions for particular iterations, and other parameters that the worker process needs, and then starts the worker processes. An **iteration** consists of computing the dynamics based on the states of all nodes at time $t = 0$ and over all contagions, up through time t_{max}. Specifically, an iteration consists of computations over the following nested loops: (i) over all time steps $t \in [0, t_{max} - 1]$; (ii) for each time, over all contagions c_j, $j \in [1, n_c]$; and (iii) for each contagion, over all nodes v_i, $1 \leq i \leq n$. For each combination of (t, c_j, v_i), Equation (1) is evaluated, generating v_i's next state s^{t+1}_{i,c_j}. If the next state is different from the current state of v_i, then this next state is written to file, along with the corresponding iteration number, t, c_j, and v_i; this is the simulation output.

3.2 Agent State Transition Models From Rules

State transition models $M_{i,j}^{st}$, which are specified in input files, represent the local functions of Section 2. Below are examples of two state transition model files; each specifies the contagion number, followed by a row of entities that constitute the elements of a rule; all subsequent lines contain particular rule names and parameter values for these rules. Hence, a model is composed of rules. Currently, there is a fixed but extensible set of rules. Moreover, the model files below are used to perform a simulation such as that in Section 2.2.

Contagion 1 (c_1) has one rule. It is a (deterministic) threshold model that applies to all nodes, and describes the transition from state 0 to state 1 when at least 3 neighbors of a node are in state 1 (the cause). The threshold change based on influence from the SIR model is zero, meaning that the SIR contagion does not affect the threshold-based contagion. The last value of 1 represents a minimum threshold for the nodes, in the event of a threshold decrease; the threshold cannot go below 1.

Contagion 2 (c_2) is an SIR model. There are two rules, one for the state transition S $\rightarrow$ I and one for the transition I $\rightarrow$ R. The transition S $\rightarrow$ I is governed by an edge probability of 0.006. If the infected (resp., susceptible) node is in state 1 for c_1, then probability is reduced by the factor 0.5 (resp., 0.5). The Python list "[1,I]" indicates that states 1 from c_1 and state I from c_2 influence the transition S $\rightarrow$ I. Thus, c_1 influences c_2. The second rule states that a node stays in the infected state for 10 time units before transitioning I $\rightarrow$ R.

```
Contagion 1
node from_state to_state cause rule param_1  param_2 param_3
all 0 1 [1] deterministic_progressive_node_threshold 0 3 1

Contagion 2
node from_state to_state cause rule param_1 param_2 param_3
all S I [1,I] edge_probability 0.006 0.5 0.5
all I R auto discrete_time_auto 10
```

4 Performance Evaluation

The networks in Section 4.1 are used to perform strong scaling studies of simulation time in Section 4.2.

4.1 Networks

Networks used in performance analyses and the case studies of Section 5 are given in Table 1. Two are face-to-face human social contact networks and the third (Epinions) is an online social network.

4.2 Strong Scaling Results

Figure 2 shows strong scaling results for two-contagion simulations for each of the three networks. This is the total execution time over all worker processes (from the start of the first worker process to the end of the last executing process).

Table 1: The city-based human contact networks (first two entries) were made with the procedures in [3]. Each network is the giant component of the network, since we run dynamics on these networks. Property computations were performed with [2].

Network	**Type**	**Num. Nodes**	**Num. Edges**	**Ave. Deg.**	**Max. Deg.**	**Ave. Clus. Coef.**	**Diameter**
Danville, VA	human contact	12961	44393	6.85	93	0.277	16
Newport News, VA	human contact	64425	418879	13.00	344	0.261	22
Epinions	online social	75877	405739	10.69	3044	0.138	15

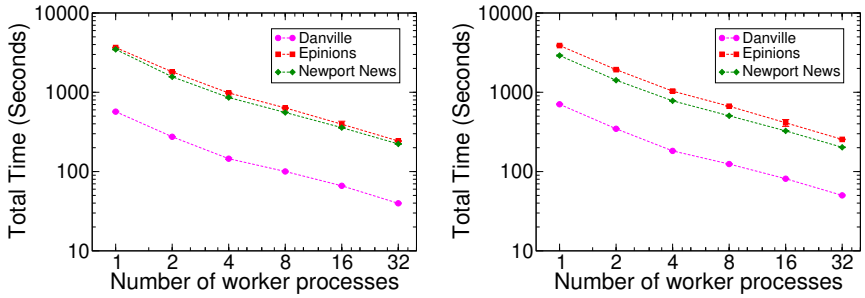
The two contagions do not interact in Figure 2a; there is interaction between contagions in Figure 2b. The times are greater in the latter plot because all neighbors for both contagions must be iterated over to update the state of each contagion; these neighborhood iterations take place at each time step of each iteration for one simulation. Each data point, with $\pm$ one standard deviation error bars, represents the results of ten simulations for each set of conditions. The data in both plots indicate that CSonNet exhibits strong scaling for independent and interacting contagions. The simulation conditions are appreciably onerous. Each time data point is for a simulation with 100 iterations, each over 100 time steps. In practice (e.g., for epidemic simulations), less than 100 iterations are performed (often on the order of 30), and typically about 14-30 time steps (days) are simulated. Yet, computations by worker processes complete in under 300 seconds for $n_{wp} = 32$ in Figure 2b. We are interested in determining the sizes of networks on which simulations can be run in under five minutes because typical studies involve large parametric studies with many simulations.

5 Case Studies

Two multi-contagion case studies are presented. The first case study is a Threshold-SIR system spreading on the Newport News network, and is motivated in part by [8,26]. The second is an SIR_1-SIR_2 system spreading on the Danville network. It is inspired by [16,22]. In the first case study, one contagion inhibits the other; in the second study, two contagions reinforce each other. The purpose of these case studies is to demonstrate multi-contagion capabilities of the code.

5.1 Threshold-SIR Two-Contagion Model and Simulations

One contagion model (for c_1) is a Granovetter threshold model [12], with threshold $\theta = 3$, and is used to model mask wearing during COVID. States 0 and 1 indicate that a person is *not* wearing a mask and *is* wearing a mask respectively. A node transitions from state 0 to state 1 if at least θ of its neighbors are in



(a) c_1, c_2 independent (b) c_1, c_2 affect each other

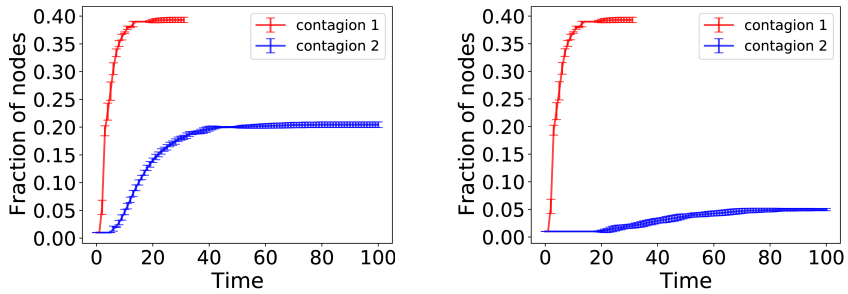
Fig. 2: Strong scaling of total execution time of worker processes. These strong scaling plots were generated for ten replicate simulations, where each simulation is run for 100 iterations. (a) Timing data for two non-interacting contagions, with one curve for each network. (b) Timing data are for interacting contagions.

state 1. The contagion model for c_2 is an SIR model for COVID. Contagion c_1 may affect c_2 as follows. Consider a person v_j in state I (i.e., infected with COVID) and another person who is v_i in state S (i.e., is susceptible). Each of v_i and v_j may or may not be wearing a mask during an encounter. When a person wears a mask, the probability of transmission is reduced by a multiplicative factor attributed to each person. For each person v_i wearing a mask, we let $m_i = 0.5$; otherwise, the factor is $m_i = 1$. Thus, the edge probability $p_{e,i}$ for transmission along the edge e between a susceptible person v_i and an infected person v_j is given by $p_{e,i} = w_{e,base,i} \cdot m_i \cdot m_j$. In our case study, $w_{e,base,i} = 0.006$ and the infectious duration $t_{inf} = 10$ days. COVID does not affect mask wearing in our model. The state transition model input files for this case study—for the interacting contagions simulation—were provided in Section 3.2.

Figure 3 provides cumulative infection curves (for c_2) and cumulative numbers of nodes in state 1 (for c_1) for the Newport News network. Seed nodes for each contagion are chosen uniformly at random and separately for each contagion: 0.0062 fraction of nodes for c_1 and 0.0062 fraction of nodes for c_2 in each iteration. The particular seed nodes are different for each of the 100 iterations, but the same seeds are used across the two simulations, for comparison. Figure 3a provides baseline data for non-interacting contagions. Figure 3b shows results for the case of mask wearing (c_1) affecting the spread of COVID (c_2). The latter plot indicates that mask wearing reduces the spread of COVID [14].

5.2 SIR_1-SIR_2 Two-Contagion Model and Simulations

In this SIR_1-SIR_2 system, if contagion c_1 affects contagion c_2, then this is realized by increasing the probability that a node v_ℓ contracts c_2, given that it has already contracted c_1. More formally, consider a two-contagion system where the contagion models are SIR_i and SIR_j with $i, j \in \{1, 2\}$ and $i \neq j$. If the



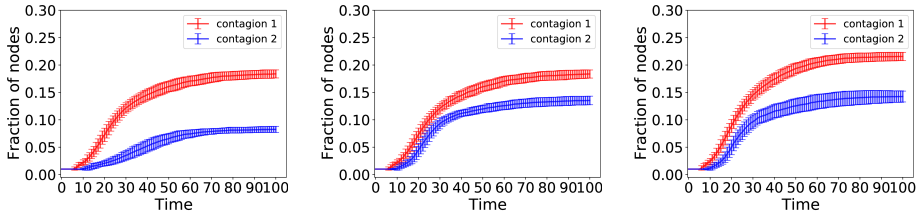
(a) c_1, c_2 independent (b) Asymmetric: c_1 affects c_2

Fig. 3: Results from a two-contagion model on the Newport News network; c_1 is a threshold model for mask wearing and c_2 is an SIR model. There are two types of interactions: (a) independent contagions c_1 and c_2 (i.e., they do not affect each other) and (b) asymmetric contagions (i.e, contagion c_1 affects contagion c_2, but not vice versa). Each data point in time is the average result over 100 iterations with error bars for $\pm$ one standard deviation.

simultaneously evolving contagions do not interact, then $m_k = 1$ for contagion SIR_k $(k = 1, 2)$ by definition. However, if contagion SIR_i affects SIR_j, and a node v_ℓ has already contracted contagion c_i, then the edge weight (i.e., probability of infection) for c_j, is $w_{e,j} = w_{e,base,j} \cdot m_j$. If $0 \leq m_j < 1$, then contracting contagion c_i reduces the probability of contracting c_j. If $m_j > 1$, then contracting contagion c_i increases the probability of contracting c_j. For SIR_1, the base edge weight is $w_{e,base,1} = 0.01$, the infectious duration is $t_{inf,1} = 10$ days, and for interacting contagions $m_1 = 8$. For SIR_2, $w_{e,base,2} = 0.005$, $t_{inf,2} = 12$ days, and for interacting contagions $m_2 = 5$. Since $m_1, m_2 > 1$, the contagions, when interacting, reinforce each other.

Figure 4 shows results from three simulations. Seed nodes for each contagion are chosen uniformly at random: 0.0077 fraction of nodes for c_1 and 0.0077 fraction of nodes for c_2 in each iteration. The particular seed nodes are different for each iteration, but the same seeds are used across the three simulations, for comparison. All plots are cumulative fractions of infected individuals as a function of time. Error bars at each time are $\pm$ one standard deviations over the 100 iterations per simulation. In Figure 4a, the two contagions do not interact. In Figure 4b, c_1 affects c_2, but not vice versa; so the cumulative fraction of infections increases for c_2 while remaining unchanged for c_1. In Figure 4c, the two contagions affect each other and the cumulative infected fractions increase for both contagions.

Acknowledgments: We thank the anonymous reviewers for their helpful feedback. We thank our colleagues at NSSAC and Research Computing at the University of Virginia for providing computational resources and technical support. This work has been partially supported by University of Virginia Strategic In-



(a) c_1, c_2 independent (b) Asymmetric: c_1 affects c_2 (c) c_1, c_2 affect each other

Fig. 4: Results from a two-contagion model on the Danville network where c_1 and c_2 are both SIR models. There are three types of interactions: (a) independent contagions c_1 and c_2 (i.e., they do not affect each other); (b) asymmetric contagions (i.e, contagion c_1 affects contagion c_2); and (c) symmetric contagions (i.e., c_1 and c_2 affect each other).

vestment Fund award number SIF160, NSF Grant OAC-1916805 (CINES), NSF Grant CMMI-1916670 (CRISP 2.0) and CCF-1918656 (Expeditions).

References

1. Adiga, A., et al.: Graphical dynamical systems and their applications to bio-social systems. Int J Adv Eng Sci Appl Math 11, 153–171 (2019)
2. Ahmed, N.K., Alo, R.A., Amelink, C.T., et al.: net.science: A cyberinfrastructure for sustained innovation in network science and engineering. In: Gateway Conference. pp. 71–74 (2020)
3. Barrett, C.L., et al.: Generation and analysis of large synthetic social contact networks. In: Winter Simulation Conference (WSC). pp. 1003–1014 (2009)
4. Barrett, C.L., et al.: Complexity of reachability problems for finite discrete dynamical systems. Journal of Computer and System Sciences 72(8), 1317–1345 (2006)
5. Barrett, C.L., et al.: Modeling and analyzing social network dynamics using stochastic discrete graphical dynamical systems. TCS 412(30), 3932–3946 (2011)
6. Borodin, A., Filmus, Y., Oren, J.: Threshold models for competitive influence in social networks. In: International Workshop on Internet and Network Economics (WINE). pp. 539–550 (2010)
7. Budak, C., Agrawal, D., Abbadi, A.E.: Limiting the spread of misinformation in social networks. In: WWW. pp. 665–674 (2011)
8. Catching, A., Capponi, S., Yeh, M.T., et al.: Examining the interplay between face mask usage, asymptomatic transmission, and social distancing on the spread of covid-19. Scientific Reports 11, 1–11 (2021)
9. Cheng, V.C.C., Wong, S.C., et al.: The role of community-wide wearing of face mask for control of coronavirus disease 2019 (covid-19) epidemic due to sars-cov-2. Journal of Infection 81, 107–114 (2020)
10. Collier, N., North, M.: Parallel agent-based simulation with repast for high performance computing. Simulation 89(10), 1215–1235 (2012)
11. Goyal, S., Kearns, M.: Competitive contagion in networks. In: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing. p. 759–774 (2012)

12. Granovetter, M.: Threshold models of collective behavior. The American Journal of Sociology 83(6), 1420–1443 (1978)
13. Kishore, A., Machi, L., Kuhlman, C.J., Machi, D., Ravi, S.S.: A framework for simulating multiple contagions over multiple networks. Technical Report (2021), accessible at https://tinyurl.com/zyf2htn
14. Li, T., Liu, Y., Li, M., Qian, X., Dai, S.Y.: Mask or no mask for covid-19: A public health and market study. PloS one 15(8), e0237691 (2020)
15. Luke, S., Balan, G.C., Sullivan, K., Panait, L.: MASON agent-based modeling framework (2019), https://github.com/eclab/mason and https://cs.gmu.edu/ eclab/projects/mason/
16. Martcheva, M., Pilyugin, S.S.: The role of coinfection in multidisease dynamics. SIAM Journal on Applied Mathematics 66(3), 843–872 (2006)
17. Min, B., Miguel, M.S.: Competing contagion processes: Complex contagion triggered by simple contagion. Scientific Reports 8(10422), 8 pages (2018)
18. Mortveit, H.S., Reidys, C.M.: An Introduction to Sequential Dynamical Systems. Universitext, Springer Verlag (2007)
19. Myers, S.A., Leskovec, J.: Clash of the contagions: Cooperation and competition in information diffusion. In: 12th International Conference on Data Mining (ICDM). pp. 539–548 (2012)
20. Nguyen, N.P., Yan, G., Thai, M.T.: Analysis of misinformation containment in online social networks. Computer Networks 57(10), 2133–2146 (2013)
21. Patriarca, M., Castello, X., Uriarte, J.R., Eguiluz, V.M., Miguel, M.S.: Influence of community structure on misinformation containment in online social networks. Advances in Complex Systems 15, 1250048–1–1250048–24 (2012)
22. Pawlowski, A., Jansson, M., Sköld, M., Rottenberg, M.E., Källenius, G.: Tuberculosis and hiv co-infection. PLoS pathogens 8(2), e1002464 (2012)
23. Priest, J.D., Kishore, A., et al.: Csonnet: An agent-based modeling software system for discrete time simulation. In: WSC (2021), accepted, accessible at: https://tinyurl.com/cnypt3u3
24. Railsback, S., Ayllón, D., Berger, U., Grimm, V., Lytinen, S., Sheppard, C., Thiele, J.: Improving execution speed of models implemented in netlogo. Journal of Artificial Societies and Social Simulation 20(1) (2017)
25. Rossetti, G., Milli, L., et al.: Ndlib: a python library to model and analyze diffusion processes over complex networks. International Journal of Data Science and Analytics 5(1), 61–79 (2018)
26. Sahneh, F.D., Scoglio, C.: Epidemic spread in human networks. In: 50th IEEE Conference on Decision and Control and European Control Conference. pp. 3008–3013 (2011)
27. Vasconcelos, V.V., Levin, S.A., Pinheiro, F.L.: Consensus and polarization in competing complex contagion processes. J. R. Soc. Interface 16, 20190196–1–20190196–8 (2019)
28. Weng, L., Flammini, A., Vespignani, A., Menczer, F.: Competition among memes in a world with limited attention. Scientific Reports 2(335), 9 pages (2012)
29. Yang, Y., Mao, L., Metcalf, S.S.: Diffusion of hurricane evacuation behavior through a home-workplace social network: A spatially explicit agent-based simulation model. Computers, Environment and Urban Systems 74, 13–22 (2019)