GEMF: GENERALIZED EPIDEMIC MODELING FRAMEWORK SOFTWARE IN C

FUTING FAN

Network Science and Engineering Group (NetSE)
Department of Electrical and Computer Engineering
Kansas State University
Manhattan, KS 66502, USA

1. Examples in C

In order to the determine nodal and edge-based transitions rates, we use node transition graph. We used NetworkX in our examples because of due to its ubiquitous use, although a user can consider other modules with minor alterations in the codes. For individual-based epidemic models, transition graphs represent only the transition mechanism for each node in the network and not for the entire population.

When defining an epidemic model, compartments should be defined first; For example, the SIS model has only two compartments: susceptible and infected.

Second, transitions between compartments, transition rates and their types should be defined. The inducer compartment and layers that define neighbor nodes must also be specified.

The following sections present examples of epidemic models that we simulated with Gemf.

For Windows users, we provided execution files for both 32-bit and 64-bit. If you choose to compile on your own machine with an IDE like Visual Studio.

For Visual Studio, remember to choose 'Win32 console application' and 'empty project' for your project. After include all header files and C files into the project, you should be able to build a solution.

For other IDEs, you need to specify a designed preprocessor macro to trigger the conditional compilation compatible with Windows. You can choose either one from 'WIN32', 'WIN_X64' and '_CONSOLE'.

For Linux/Unix users, a working Makefile has been provided together with source files. A 'make' command will generate the working execution file for you machine.

The command for running the program is shown below, if none argument is specified, a default [para.txt] will be assigned.[1].

```
./GEMF para_file.txt
```

There are 3 input and output files specified inside the configuration file. To make things clear, all 3 of them should in the right form as they are located via relative path, depending on the present working directory while you execute the program.

1.1. **Sample.** One sample of parameter file.

```
[DATA_FILE]
network.txt

[STATUS_FILE]
status_SIS.txt

[OUT_FILE]
output.txt

[STATUS_BEGIN]
```

[1]The latest version of Gemf can be found here.

```
1

[DIRECTED]
0

[STATUS_BEGIN]
1

[MAX_TIME]
80.000

[MAX_EVENTS]
10000

[RUN_TIMES]
1

[SAMPLE_SIZE]
1000

[SIM_ROUNDS]
1

[INTERVAL_NUM]
1000

[NODAL_TRAN_MATRIX]
0   0   0
0   0   1
0   0   0

[EDGED_TRAN_MATRIX]
0   1.2 0
0   0   0
0   0   0

[INDUCER_LIST]
2

[NETWORK_INFO]
#This section is optional
#weight sign: 0 for unweighted, 1 for weighted
0
#min max node number
1 1024
#edge number for each layer, one number per line
10086

[SHOW_INDUCER]
1
```

1.2. **Explanation.** Explanation for all sections in parameter file. Over all, all section are started with a section name surrounded by '[]'. Comment is allowed, with '#' at the beginning of the line. Each file name must occupy one line exclusively, space in the directory is allowed. Order of the sections is trivial.

1.2.1. *Network File.* A network file is a file of adjacence list, weighted or unweighted. Each line contains 2 integer for a pair of nodes, or 3 integer for a pair of nodes and a number for weight.

Comment is allowed only on top of the file, with a '#' at the beginning of the line. No comment allowd among data list.

1.2.2. *Status File.* There are three modes to config a status file.

In mode 1, status of all nodes are specified specificly while for the other modes one compartment can be dynamic.

In mode 2, the population of each compartment is specified. Program will pick node randomly to meet the requirment. Only one compartment can be configured as -1, meaning this one is dynamic, taking all the rest population.

In mode 3, status of some nodes are specified while all the rest all in the same compartment.

```
1  //MODE 1. full list mode
2  /*
3   *e.g. for 3 compartments, 10nodes
4   *{
5       0
6       1
7       1
8       2
9       1
10      1
11      1
12      0
13      1
14      2
15  *}
16  *
17  */
18 //MODE 2. statistic mode
19 /*
20  *e.g. for 3 compartments, 10nodes
21  *
22  *{
23      6 2 2
24      or
25      -1 2 2
26  *}
27  */
28 //MODE 3. compensate list mode
29 /*
30  *e.g. for 3 compartments, 10nodes, node 1,2,9 is in compartment 1, node
        4,8 is incompartment 2, all rest in the other compartmetn
31  *{
32      0
33      1 1
34      2 1
```

```
35 ||       9 1
36 ||       4 2
37 ||       8 2
38 ||  *}
39 ||  */
```

1.2.3. *DATA_FILE.* L lines(L is the number of network layers for this simulation). Each line is consisted of a file name, indicating the adjacency list for the corresponding layer. If there are three columns, then the third one should be weight. Otherwise there should be two columns only.

1.2.4. *STATUS_FILE.* One line, consisted of a file name for status input, no sapce. The file has N lines of integer indicating the initial status of all N nodes. All value must be within the range of all compartments.

1.2.5. *OUT_FILE.* One line, consisted of a file name for output data.

1.2.6. *STATUS_BEGIN.* An integer indicating the mininum serial number representing all compartments. Shoud be 1 for data prepared for most script language or 0 for data prepared for most compile language.

1.2.7. *DIRECTED.* 0 for undirected network and 1 for directed network

1.2.8. *MAX_TIME.* A float number treated as stop condition. Simulation will stop after exceed this time. Value is based on time calculated in simulation, not actual cpu time.

1.2.9. *MAX_EVENTS.* A integer treated as stop condition, maximum number of events. This works together with MAX_TIME, simulation will stop if reach eigher limit.

1.2.10. *RUN_TIMES.* A positive integer to control how many times of simulation are to expect. If it's greater than 1, the section of SAMPLE_SIZE will be efective.

1.2.11. *SAMPLE_SIZE.* A positive integer. If the value of run_times is greater than 1, the the result will be calculated based on this value S. Pick S time points uniformly from the whole time period. Calculate the average population of each compartment among all times of simulations at there sample time points.

1.2.12. *SIM_ROUNDS.* Run only 1 time if valued 1 or below, otherwise run several times and calculate results into M even intervals. M is specified in the next section.

1.2.13. *INTERVAL_NUM.* Number of intervals for sampling if run more than once.

1.2.14. *NODAL_TRAN_MATRIX.* Transition rate matrix for nodal transition. Should be a M*M matrix.(M is the number of compartments)

1.2.15. *EDGED_TRAN_MATRIX.* Transition rate matrixes for edge based transition. Should be L M*M matrixes.(M is the number of compartments, L is the number of layers)

1.2.16. *INDUCER_LIST.* Inducer list for all layers. Should be a list of integers seperated by space. All the value should be in the range of the values of compartments. The length of this list shoud be L.(L is the number of layers)

1.2.17. *NETWORK_INFO.* This section is optional. If not specified or imcomplete, program will automaticly collect infomation from network file. However, this might take some time if network is large.

1.2.18. *SHOW_INDUCER.* If this section is present with nonzero value and the simulation is run only once, then show the inducer nodes took effect for each event in the output file. Inducer nodes in each layer are contained in '[]', with the first one representing nodal transition. So there should be 1 more pair of brackets than the number of layers.

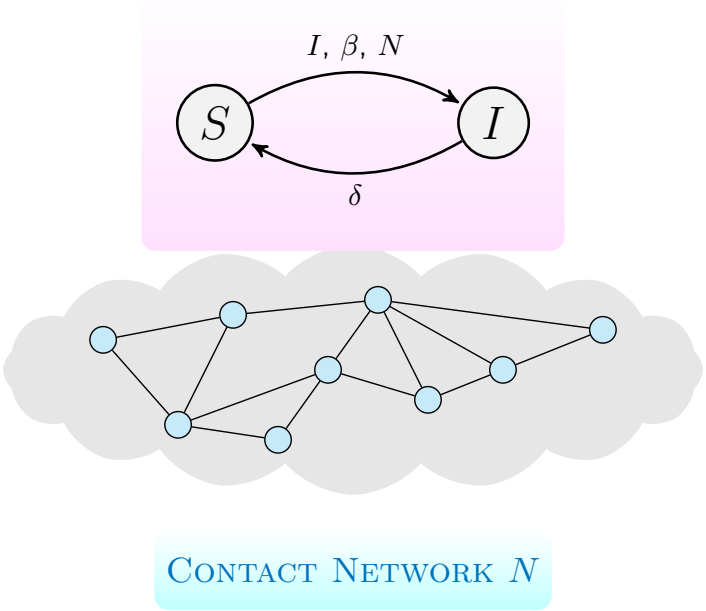

FIGURE 1. Schematic of the network-based SIS model

1.3. **SIS.** As mentioned, each node in an SIS model can be susceptible or infected; therefore, the number of compartments was denoted by $M = 2$. A susceptible node can become infected if it is surrounded by infected nodes. Infection process of a node with one infected neighbor is a Poisson process with transition rate β. The infection processes are stochastically independent of each other; therefore, for a susceptible node with more than one infected node in its neighborhood, the transition rate is the infection rate β times the number of infected neighbor nodes. The neighborhood of each node is determined by a contact network N. In addition to the infection process, a recovery process also exists. An infected node becomes susceptible again with a curing rate δ. The main characteristics and a node transition graph for the SIS model are shown in Table 1 and Figure 1.

TABLE 1. Descriptions of the SIS model

SIS					
State	Transition	Type	Parameter	Inducer	Layer
S	$(S \to E)$	edge-based	β	Neighbors in I	1
I	$(I \to S)$	node-based	δ		

Suppose beta = 0.2; delta = 1, parameters in Table 1 can be entered by the following lines:

```
[NODAL_TRAN_MATRIX]
0    0
1    0
[EDGED_TRAN_MATRIX]
0    0.2
0    0
[STATUS_BEGIN]
1
[INDUCER_LIST]
2
```

1.3.1. *Simulation.* Status file is consisted of N integers indicating the initial status of all N nodes. Both the node serial and status serial must be successive. The mininum serial number of all statuss must be specified in [STATUS_BEGIN] section. After defining Para for SIS model, we simulated an SIS model with $\beta = 0.2$ and $\delta = 1$, as shown in Figure 2. To run this simulation several other sections need to be configurated.

Define the duration of simulation:

```
[MAX_TIME]
30.000
```

Define input and output file name:

```
[DATA_FILE]
network.txt

[STATUS_FILE]
status.txt

[OUT_FILE]
output.txt
```

For each layer, there are specific entry in [DATA_FILE], [EDGED_TRAN_MATRIX],[INDUCER_LST] three sections, Which are a file name for network, a M by M matrix indicating transition relation, and a integer indicating inducer serial for this layer.

The input file file should be consisted with three columns, indicating i, j, w which means there is a link weighted w pointing from i to j. Below is an example network consisted of 3 nodes and 4 directed links all weighted 1.

```
1 2 1
1 3 1
2 3 1
3 1 1
```

The output file is consisted of 4+M columns, in which M means number of compartments.
Each line represents an event. The first column is a float number indicating time of this event, second column is the serial of the node that changes status, third and fourth line are status before and after changing respectively. The rest M columns are the populations for every compartment.
For a network of 379 nodes, with 4 susceptible and 374 infected as well as other parameters specified above, simulation result is shown in Figure 2

1.4. **SIR.** In the Susceptible-Infected-Recovered (SIR) model, each node can be either susceptible, infected, or recovered (immune). Therefore, the number of compartments, denoted by M,in the SIR model, was $M = 3$. A susceptible node can become infected if it is surrounded by infected nodes. The infection process of a node with one infected neighbor is a Poisson process with transition rate β. Similar to SIS, infection processes are stochastically independent of each other. In addition to the infection process, a recovery process also exists. An infected node recovers and becomes immune with a recovery rate δ. The main characteristics and a node transition graph for the SIR model are shown in Table 2 and Figure 3.

Table 2. Descriptors of the SIR model

SIR multilayer					
State	Transition	Type	Parameter	Inducer	Layer
S	$(S \to E)$	edge-based	β	Neighbors in I	1
I	$(I \to R)$	node-based	δ		
R					

Parameters in Table 2 (beta = 1.2; delta = 1) can be entered by the following lines:

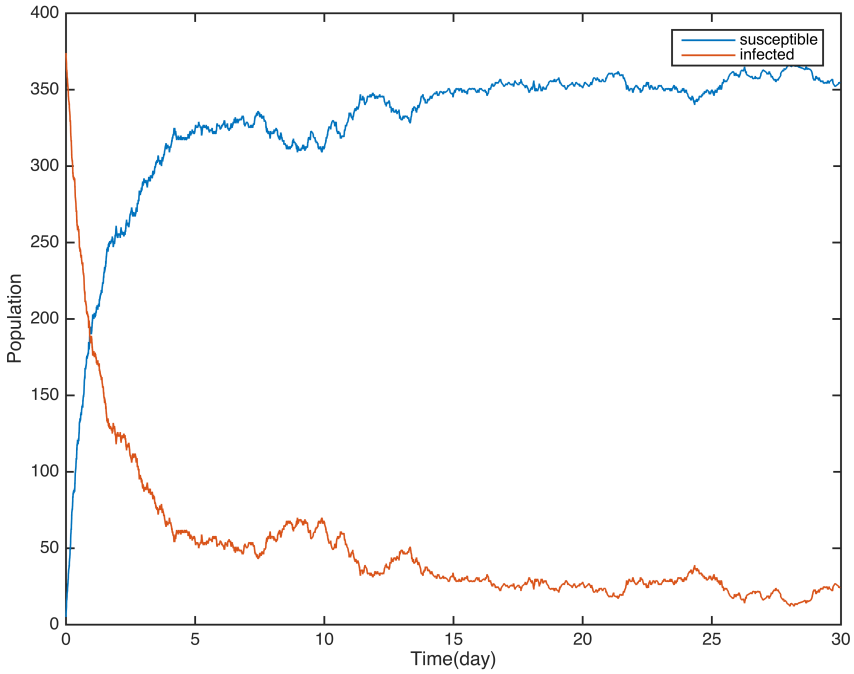

FIGURE 2. Simulation of the SIS model

```
[NODAL_TRAN_MATRIX]
0    0    0
0    0    1
0    0    0

[EDGED_TRAN_MATRIX]
0    1.2  0
0    0    0
0    0    0
[INDUCER_LST]
2
```

1.4.1. *Simulation.* After defining PARA for SIR model, we simulated an SIR model with $\beta = 1.2$, $\delta = 1$, as shown in Figure 3 for a Barabasi-Albert network with 500 nodes. Method is similar to SIS simulation in Section 1.3.1.

1.4.2. *SEIR.* In the Susceptible-Exposed-Infected-Recovered (SEIR) model, each node can be susceptible, exposed, infected, or recovered (immune). Therefore, $M = 4$. A susceptible node can become exposed, if it is surrounded by infected nodes. The infection process of a node with one infected neighbor is a Poisson process with transition rate β. The neighborhood of each node is determined by a contact network N. An exposed node is not yet infectious, but it will transition to the infected state with rate λ. Finally, an infected node recovers with a recovery rate δ. The main characteristics and a node transition graph for the SEIR model are shown in Table 3 and Figure 5.

Parameters in Table 3 (beta = 1.5; delta = 1; Lambda = .5) can be entered by the following lines:

```
[NODAL_TRAN_MATRIX]
0    0    0    0
```

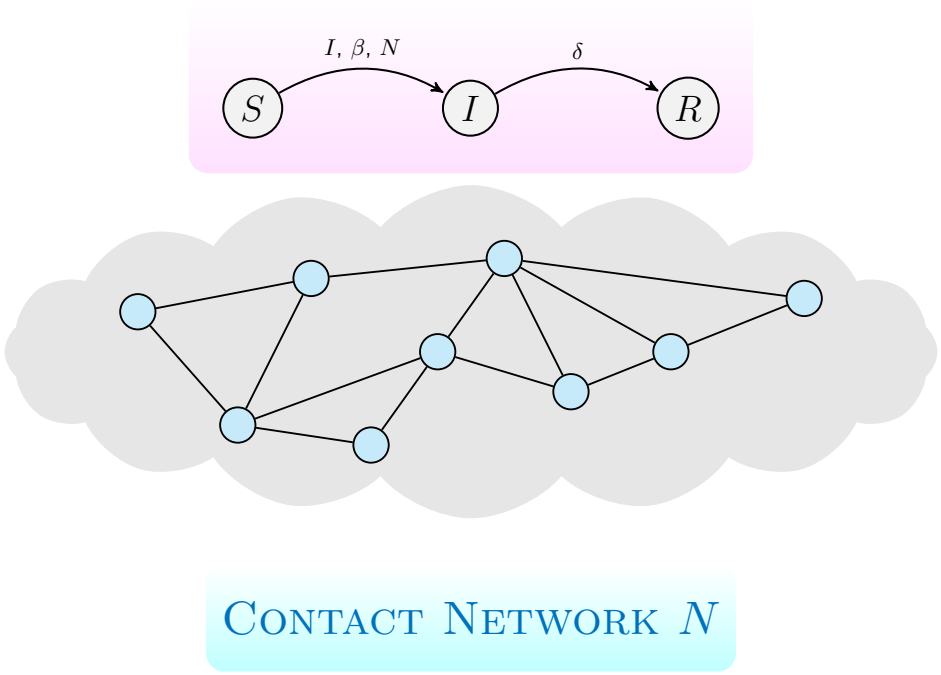

Figure 3. Node transition graph for the SIR model for nodes in N

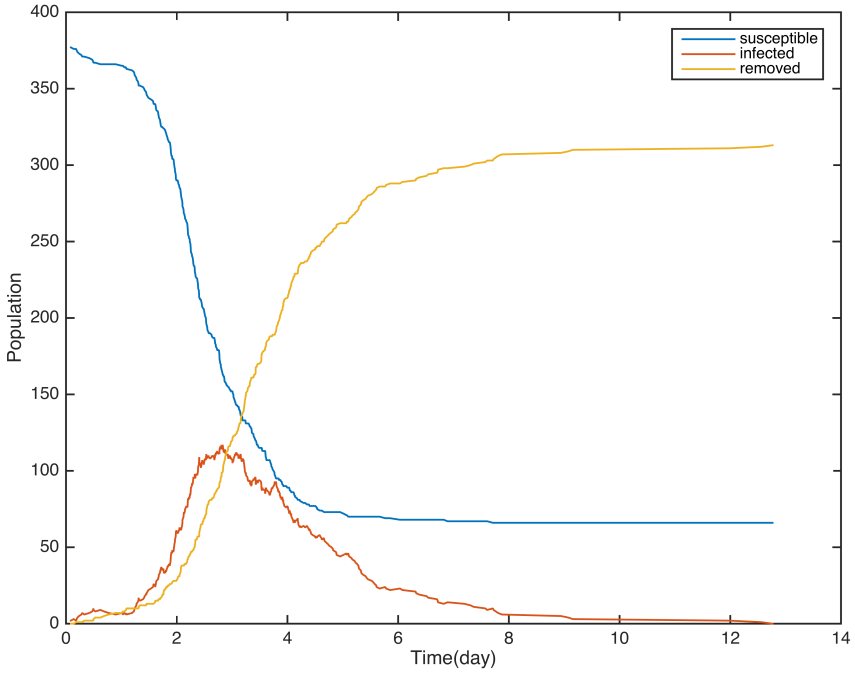

Figure 4. Simulation of the SIR model

TABLE 3. Descriptors of the SEIR

SEIR multilayer					
State	Transition	Type	Parameter	Inducer	Layer
S	$(S \to E)$	edge-based	β	Neighbors in I	1
E	$(E \to I)$	node-based	λ		
I	$(I \to R)$	node-based	δ		

```
0    0    0.5  0
0    0    0    1
0    0    0    0

[EDGED_TRAN_MATRIX]
0    1.5  0    0
0    0    0    0
0    0    0    0
0    0    0    0

[INDUCER_LST]
3
```

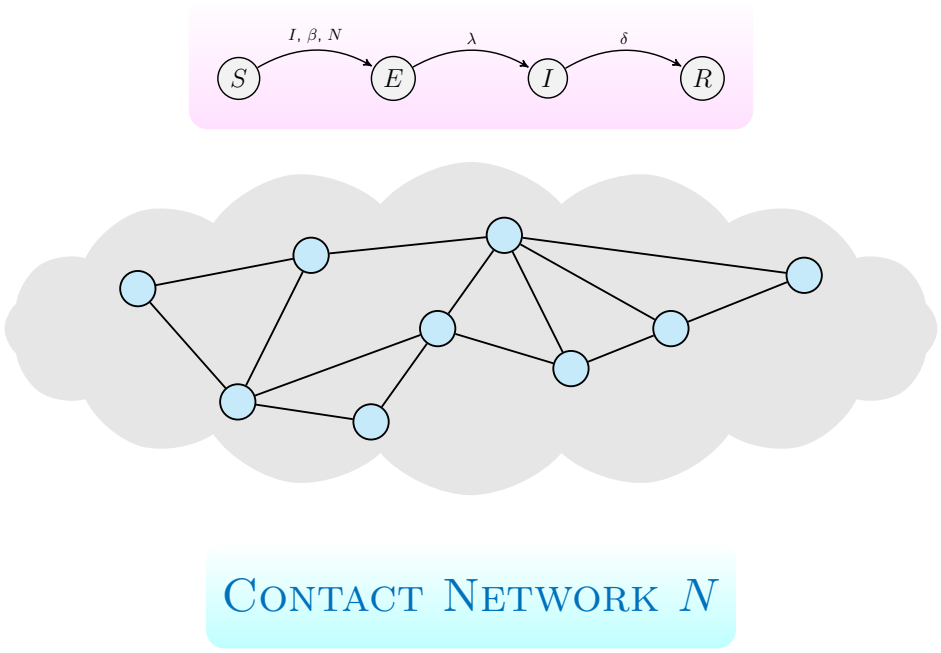

FIGURE 5. Node transition graph for the SEIR model for nodes in N

1.4.3. *Simulation.* After defining PARA for SEIR model, we simulated an SEIR model with $\beta = 1.5$, $\delta = 1$ and $\lambda = .5$, as shown in Figure 6.

1.5. **SAIS.** The Susceptible-Alert-Infected-Susceptible (SAIS) model was developed to incorporate individual reactions to the spread of a virus. In the SAIS model, each node (individual) can be susceptible, infected,

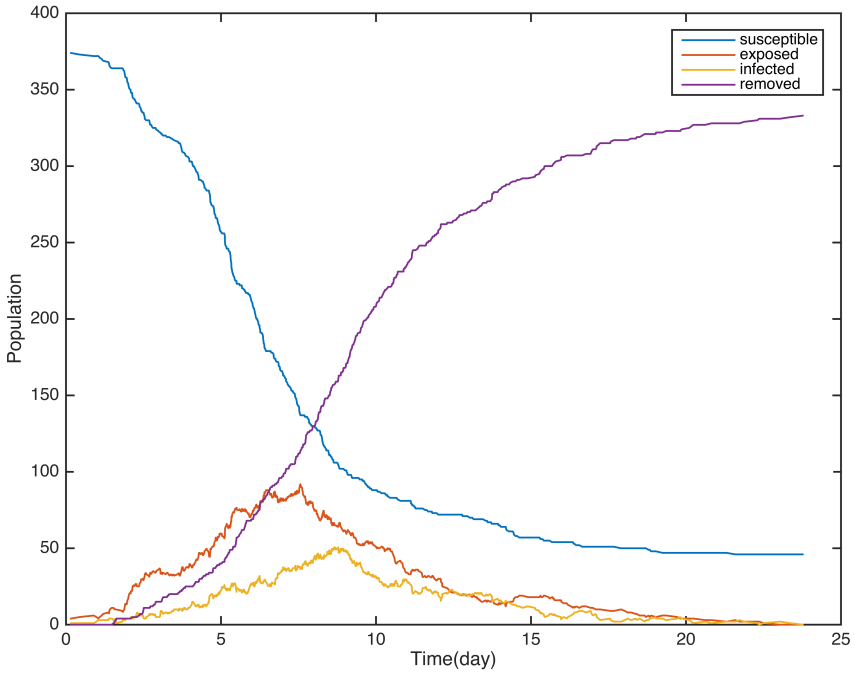

FIGURE 6. Simulation of the SEIR model

or susceptible-alert. Therefore, the number of compartments in the SAIS model was $M = 3$. The recovery process is similar to recovery process in the SIS model, characterized by the recovery rate δ. The infection process of a susceptible agent is also similar to the infection process of the SIS model, determined by infection rate β and contact network N. However, in the SAIS model, a susceptible node can become alert if it senses infected agents in its neighborhood. The alerting transition rate is κ times the number of infected agents. An alert node can also become infected by a process similar to the infection process of a susceptible node. However, the infection rate for alert nodes is lower than susceptible nodes due to the adoption of preventive behaviors. The alert infection rate is denoted by β_a with $0 < \beta_a < \beta$. The main characteristics and a schematic for the SAIS model are shown in the following Table 4 and Figure 7.

TABLE 4. Descriptors of the SAIS single layer model.

SAIS single Layer					
State	Transition	Type	Parameter	Inducer	Layer
S	$(S \to I)$	edge-based	β	Neighbors in I	1
	$(S \to A)$	edge-based	κ	Neighbors in I	1
I	$(I \to S)$	node-based	δ		
A	$(A \to I)$	edge-based	β_a	Neighbors in I	1

Parameters in Table 4 (delta= 1, beta= 2, beta_a= 0.4, kappa= 0.2)can be entered by the following lines:

```
[NODAL_TRAN_MATRIX]
0    0    0
0    0    0
1    0    0

[EDGED_TRAN_MATRIX]
```

10

```
0    0.4 2
0    0   0.2
0    0   0

[INDUCER_LST]
3
```

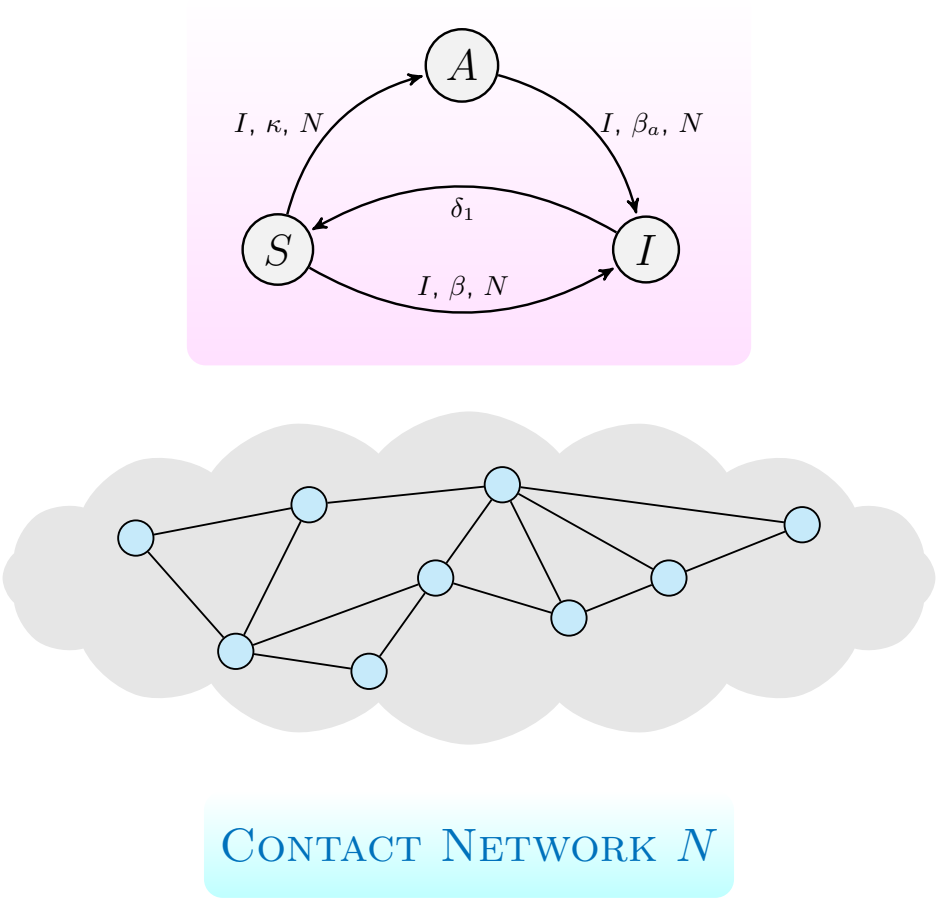

FIGURE 7. Node transition graph for the SAIS one layer model for nodes in N

1.5.1. *Simulation.* After defining PARA for SAIS model, we simulated an SAIS model in one-layer (N), with $\beta = \frac{5}{\lambda_1(\mathcal{G}_1)}$, $\delta = 1$ and $\beta_a = \frac{0.5}{\lambda_1(\mathcal{G}_1)}$, and $\kappa = 0.2\beta$, as shown in Figure 8.

1.5.2. *SAIS Multilayer.* The SAIS model on a two layer network was developed to incorporate multiple sources of information to react to the spread of the virus. In the SAIS spreading model, each node (individual) can be either susceptible, infected, or susceptible-alert. Again, the number of compartments in the SAIS model was $M = 3$. The infection process of a susceptible agent was also similar to the infection process of the SIS model, determined by infection rate β and contact network N_A. However, in this version of the SAIS model, a susceptible node can become alert if it senses infected agents in its contact neighborhood or if it is notified about infected neighbors in an information network N_B. The alerting transition rate is κ times the number of infected agents in the contact network and μ times the number of infected agents in the notification network. An alert node can also become infected by a process similar to the infection process

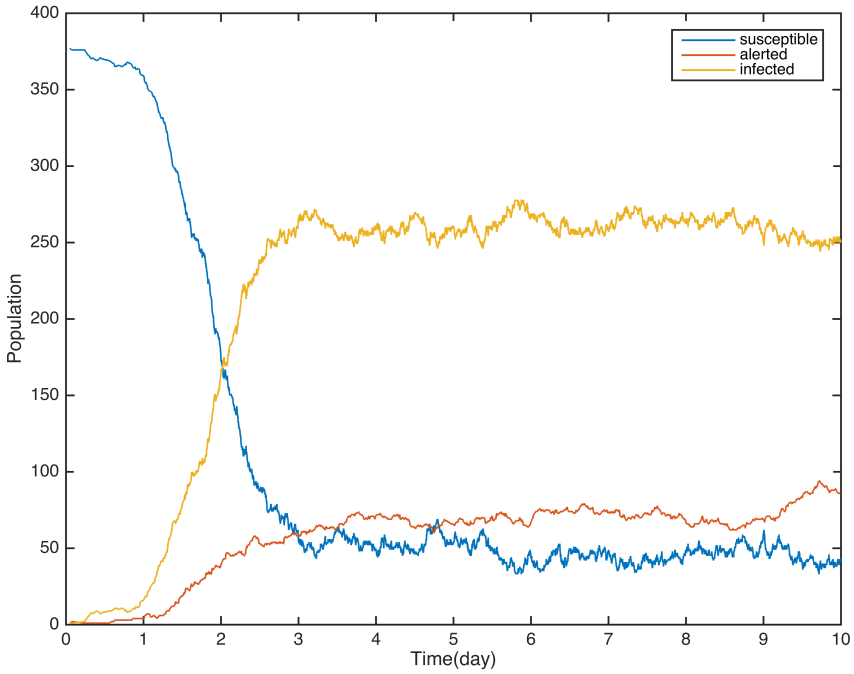

FIGURE 8. Simulation of the SAIS single layer model.

of a susceptible node. However, the infection rate for alert nodes β_a is lower than β due to the adoption of preventive behaviors such as using masks. The main characteristics and a schematic for the SAIS-2 layer model are shown in Table 5 and Figure 10.

TABLE 5. Descriptors of the SAIS two-layer model

SAIS multilayer					
State	Transition	Type	Parameter	Inducer	Layer
S	$(S \to I)$	edge-based	β	Neighbors in I	1
	$(S \to A)$	edge-based	κ	Neighbors in I	1
	$(S \to A)$	edge-based	μ	Neighbors in I	2
I	$(I \to S)$	node-based	δ		
A	$(A \to I)$	edge-based	β_a	Neighbors in I	1

Parameters in Table 5 (delta= 1, beta= 2, beta_a= 0.2, kappa= 0.4)can be entered by the following lines:

```
[DATA_FILE]
edgewd.txt
edgewd2.txt

[NODAL_TRAN_MATRIX]
0    0    0
0    0    0
1    0    0

[EDGED_TRAN_MATRIX]
0    0.4  2

```

```
0    0    0.2
0    0    0

0    1    0
0    0    0
0    0    0
[INDUCER_LST]
3 3
```

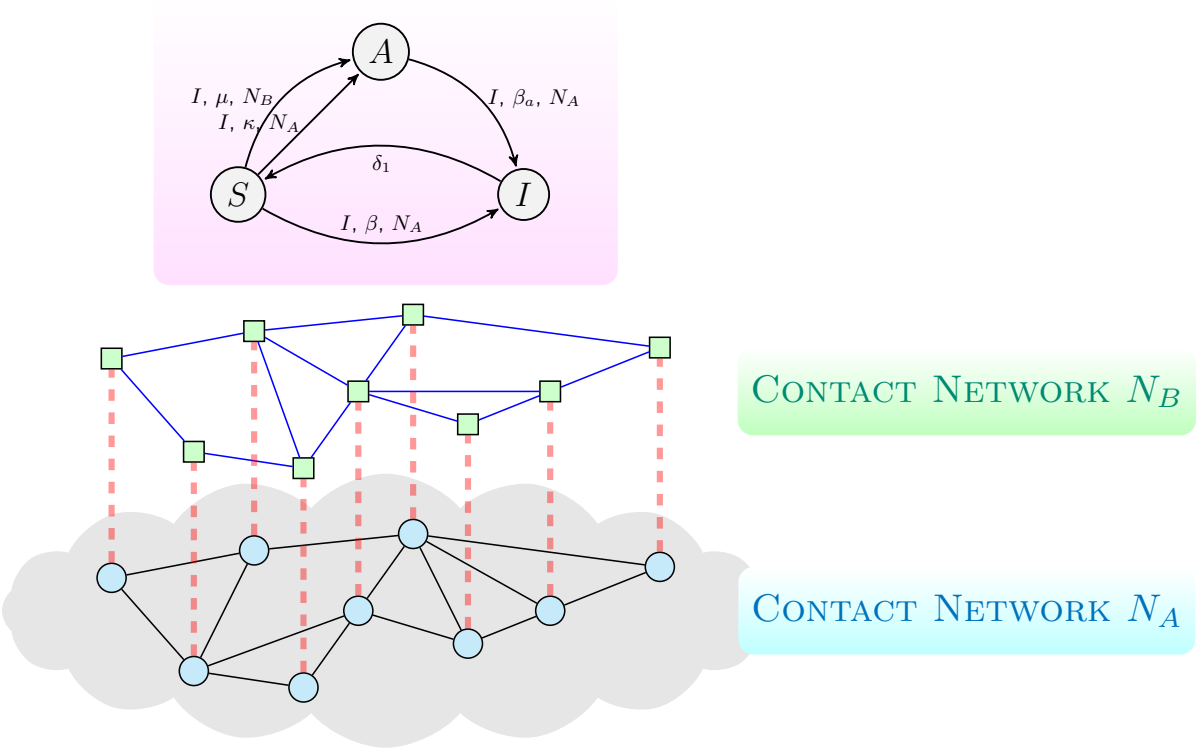

FIGURE 9

FIGURE 10. Node transition graph for the SAIS two-layer model on network with layers N_A and N_B.

1.5.3. *Simulation.* After defining PARA for SAIS model, we simulated the process with $\beta = \frac{5}{\lambda_1(\mathcal{G}_1)}$, $\delta = 1$ and $\beta_a = \frac{0.5}{\lambda_1(\mathcal{G}_1)}$, $\kappa = 0.2\beta$, and $\mu = 0.5\beta$, as shown in Figure 11.

1.6. **Multiple interacting pathogen spreading SI_1SI_2S.** Assigning only one influencer compartment to one network layer allows different elegant analysis. However, a more general possibility is that an edge-based transition $m \to n$ occurs if a neighbor j, is in a subset of the compartments, such as $q_{l,1}$ or $q_{l,2}$. This case can be treated within the same structure, allowing the network layer to be counted twice. For example, we assumed that in the first layer the model had the influencer compartment $q_{l,1}$, and in the second layer, the graph has the influencer compartment $q_{l,2}$.

The SI_1SI_2S model is an extension of continuous-time SIS spreading of a single virus on a simple graph, to the modeling of competitive viruses on a two-layer network. In this model, each node is either susceptible, 1-infected, or 2-infected (i.e., infected by Virus 1 or 2, respectively). Virus 1 spreads through network N_1,

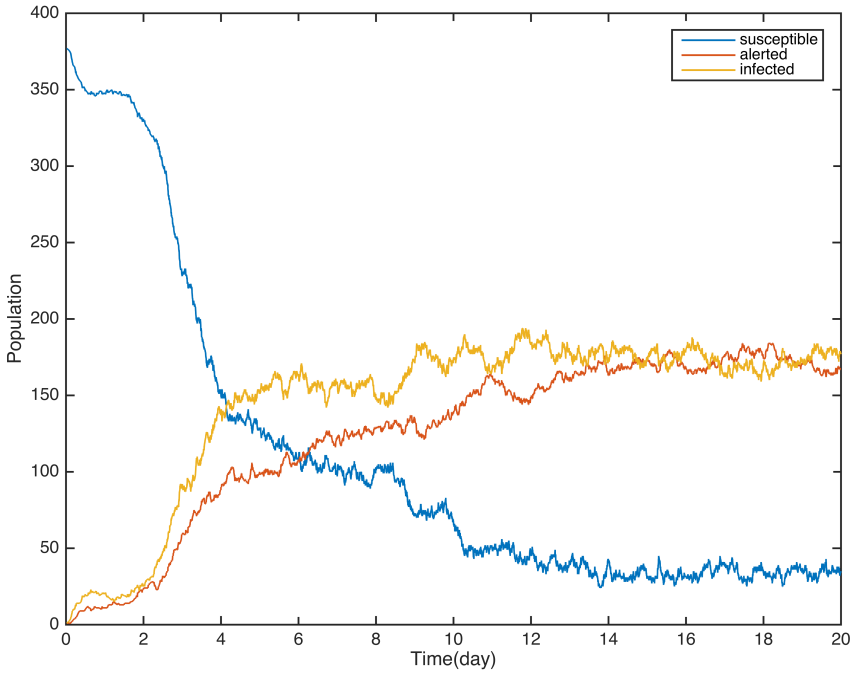

FIGURE 11. Simulation of the SAIS in 2 layer

virus 2 spreads through network N_2. In this competitive scenario, the two viruses are exclusive: a node cannot be infected by Virus 1 and Virus 2 simultaneously. Consistent with SIS propagation on a single layer, the infection and recovery processes for Virus 1 and 2 have similar characteristics. The curing process for 1-infected Node i is a Poisson process with recovery rate $\delta_1 > 0$. The infection process for susceptible Node i effectively occurs at rate $\beta_i Y_i(t)$, where $Y_i(t)$ is the number of 1-infected neighbors of node i at time t in layer N_1. Recovery and infection processes for Vvirus 2 are similarly described. The main characteristics and a node transition graph for the SI_1SI_2S model are shown in Table 6 and Figure 12.

TABLE 6. Descriptions of the SI_1SI_2S model. S: suscpetible, I_1: infected by virus 1, I_2: infected by virus 2,

SI_1SI_2S					
State	Transition	Type	Parameter	Inducer	Layer
S	$(S \to I_1)$	edge-based	β_1	Neighbors in I_2	1
	$(S \to I_2)$	edge-based	β_2	Neighbors in I_2	2
I_1	$(I_1 \to S)$	node-based	δ_1		
I_1	$(I_2 \to S)$	node-based	δ_2		

Parameters in Table 6 (delta1= 1, beta1= 2, delta2= 1, beta2= 1.5) can be entered by the following lines in two different networks N1 (G) and N2 (H):

```
[DATA_FILE]
edgewd.txt
edgewd2.txt

[NODAL_TRAN_MATRIX]
0    0    0
```

```
1    0    0
1    0    0

[EDGED_TRAN_MATRIX]
0    2    0
0    0    0
0    0    0

0    0    1.5
0    0    0
0    0    0
[INDUCER_LST]
2 3
```

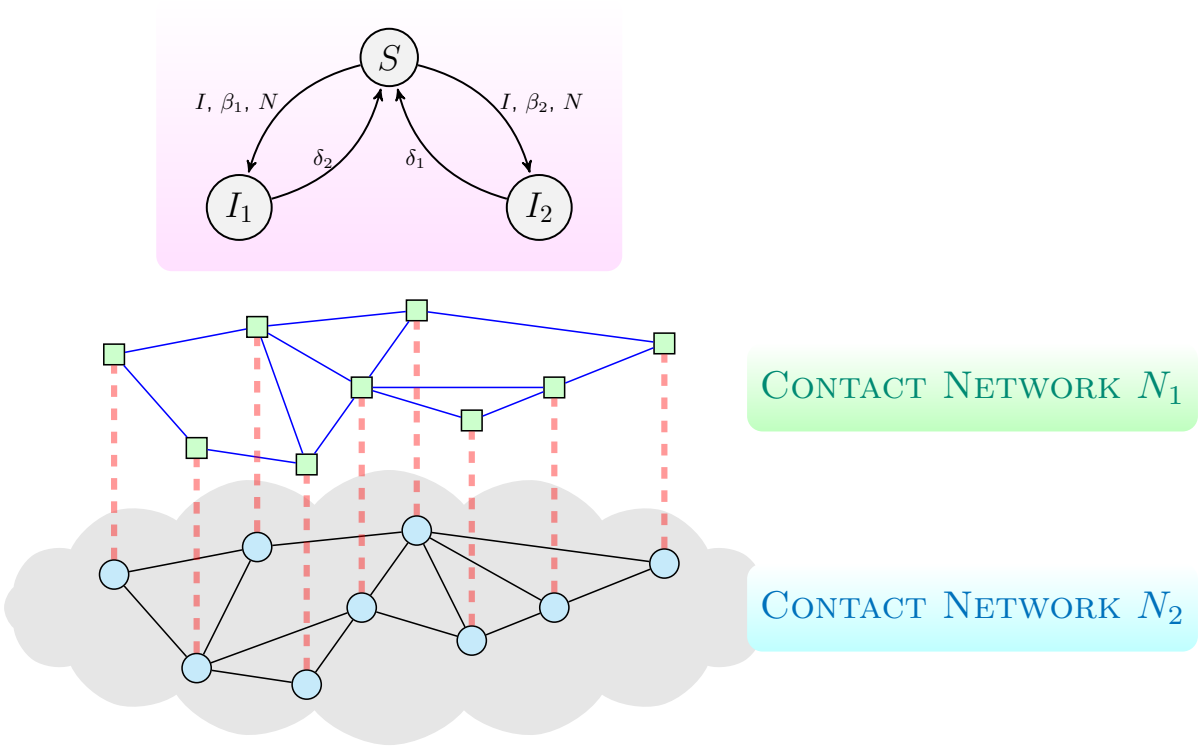

FIGURE 12. Node transition graph for the SI_1SI_2S in two layer model

1.6.1. *Simulation.* After defining PARA for this model, we simulate an S_1SI_2S model in one layer with $\beta_1 = \frac{5}{\lambda_1(\mathcal{G}_1)}$, $\beta_2 = \frac{5}{\lambda_1(\mathcal{H}_1)}$, $\delta_1 = 1$ and $\delta_2 = 1$ in Figure 13.

2. ANOTHER SIMULATE METHOD

There is another way to run this simulation, which is run T times with the same input data and average the result.
To make this happen, two parameter are envolved.

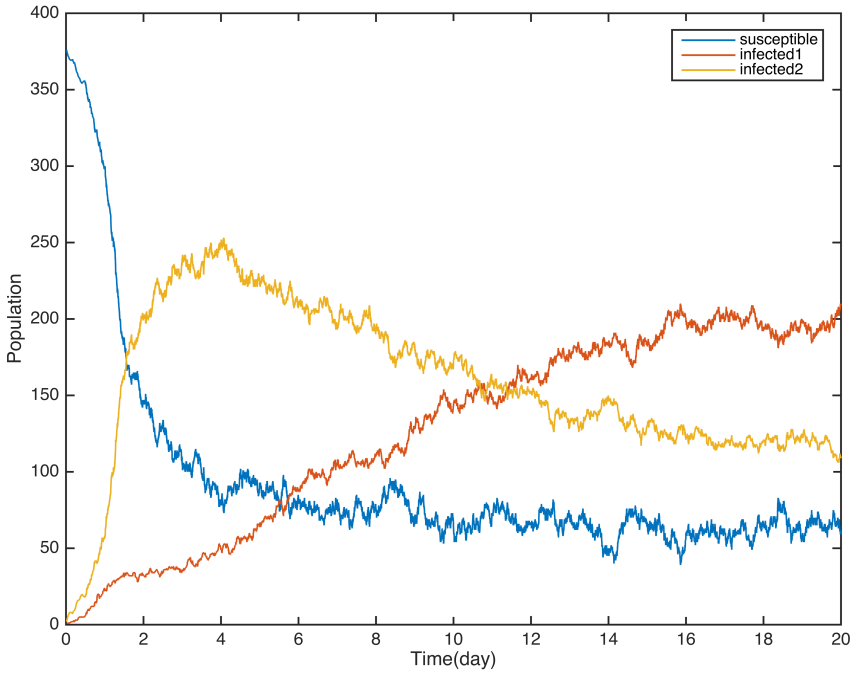

FIGURE 13. Simulation of the SI_1SI_2S model

```
[RUN_TIMES]
10

[SAMPLE_SIZE]
1000
```

If the [RUN_TIMES] value T is greater than 1, then the program will run T times with the same input, and pick S samplesuniformly, which is specified in [SAMPLE_SIZE] section. The output result will be the everaged population for each compartment at these S sample time. Respectively, there will be only 1+M columns in the output file. The first column is time, and rest M columns are for population of M compartments. The result is shown in Figure 14.

2.0.1. *Transition rates.* We used Para function to enter the required data for transition rates, as described in Section **??**. A nodal transition rate matrix is an $M \times M$ matrix in which entry mn represents the rate of nodal transition $m \to n$:

$$A_\delta \triangleq [\delta_{mn}]_{M\times M}\,. \tag{1}$$

An edge-based transition rate matrix, corresponding to the network layer l, is an $M \times M$ matrix in which entry mn represents the rate $\beta_{l,mn} > 0$ of edge-based transition $m \to n$:

$$A_\beta \triangleq \left[[\beta_{1,mn}]_{M\times M}, \cdots, [\beta_{L,mn}]_{M\times M}\right]_{1\times L} \tag{2}$$

Examples of how to use this functions are presented in Section 1.

2.0.2. *Initial condition.* With "InitialCondGen" function the initial status of each individual in the population can be determined and various approaches can be used to do this.

- User input: Initial condition is directly chosen by the user.
- Fixed initial infected population: N_J individuals randomly chosen to be in compartment J.

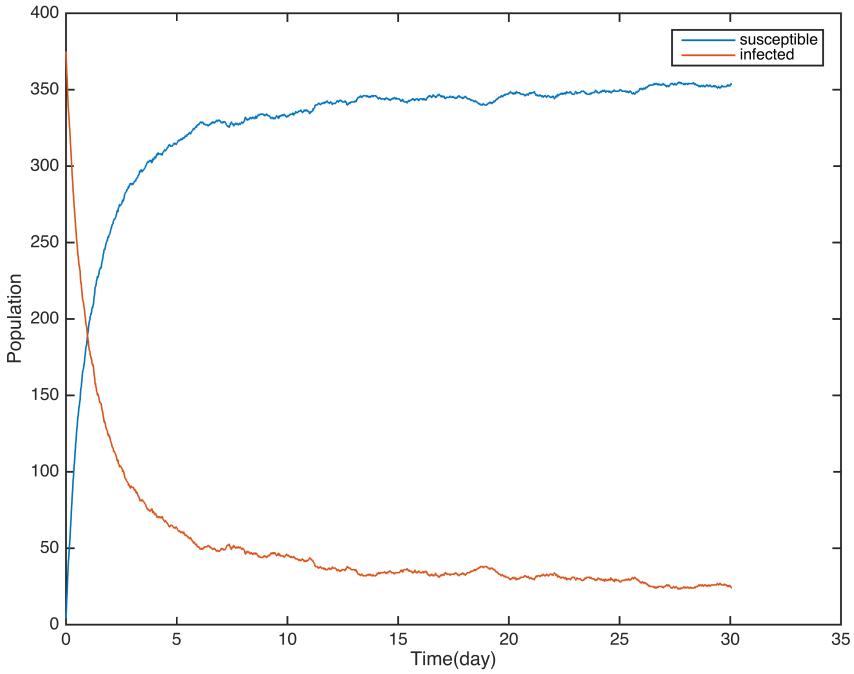

FIGURE 14. Run the simulation of the SIS model 10 times

2.1. **Simulations.** GEMF uses an event-driven approach to simulate the stochastic process. This method is advantageous compared to the discretized method. For example, in discretization approach, no transition may occur in several time increments dt or several transition may occur in one time increment; therefore, computation time for the event-based method is not unnecessarily longer and on the other side the solution is more accurate and captures more events compared to the discretized method (See [**?**], [**?**]).

Number of neighbors in influencer compartment N_q. As discussed in Section **??**, one of the key factors in edge-based transitions is the number of neighbors in influencer compartment, N_q. N_q is an $L \times N$ array, representing the number of influencer compartment for each node in each layer, weighted by edge weights. Because node status changes in each event, N_q is updated after each event. From Section 2.0.2, initial status of all nodes $X^0_{M\times N}$ is obtained. For example, if $X[:,4]^T = \begin{bmatrix}0 & 1 & \cdots & 0\end{bmatrix}_{1\times M}$, then node 4 is in compartment 2.

To compute N_q, GEMF goes over all nodes in each layer. Using network data from NETCOMB, all neighbors of node n in layer l can be derived via

$$N_{ln} = Neigh\left[l\right]\left[I_1\left[l,n\right] : I_2\left[l,n\right]\right] \tag{3}$$

with weights:

$$W_{ln} = NeighWeight\left[l\right]\left[I_1\left[l,n\right] : I_2\left[l,n\right]\right]. \tag{4}$$

Using (4), entries of N_q (influencer neighbors) can be determined by

$$N_q\left[l,n\right] = \sum_{i=1}^{|N_{ln}|} X\left[q\left[l\right], N_{ln}\left[i\right]\right] \cdot W_{ln}\left[i\right] \tag{5}$$

where $|N_{ln}|$ is the cardinality of set N_{ln}.

2.1.1. *Rate of changes.* From Section 2.0.1, we entered A_β and A_δ through PARA. The simulation code initially generated b_{il}, which is an arrays:

$$b_{il} \triangleq \begin{bmatrix} \begin{bmatrix} \sum_{i=1}^{M} \beta_{1,1i} \\ \vdots \\ \sum_{i=1}^{M} \beta_{1,Mi} \end{bmatrix}_{M\times 1} & \cdots & \begin{bmatrix} \sum_{i=1}^{M} \beta_{L,1i} \\ \vdots \\ \sum_{i=1}^{M} \beta_{L,Mi} \end{bmatrix}_{M\times 1} \end{bmatrix}_{1\times L} \tag{6}$$

where b_{il} represents the sum of edge-based transition rates of each compartment in each layer.

The array of edge-based transition rates matrix for each compartment in all layers b_i was

$$b_i \triangleq \begin{bmatrix} \begin{bmatrix} \beta_{1,11} & \cdots & \beta_{L,11} \\ \beta_{1,12} & \cdots & \beta_{L,12} \\ \vdots & \ddots & \vdots \\ \beta_{1,1M} & \cdots & \beta_{L,1M} \end{bmatrix}_{M\times L} & \cdots & \begin{bmatrix} \beta_{1,21} & \cdots & \beta_{L,21} \\ \beta_{1,22} & \cdots & \beta_{L,22} \\ \vdots & \ddots & \vdots \\ \beta_{1,2M} & \cdots & \beta_{L,2M} \end{bmatrix}_{M\times L} \end{bmatrix}_{1\times M} \tag{7}$$

For each compartment, the total leaving rate due to nodal transition was derived from 1 (by summing up each row of matrix A_δ):

$$d_i = \begin{bmatrix} \sum_{i=1}^{M} \delta_{1i} \\ \vdots \\ \sum_{i=1}^{M} \delta_{Mi} \end{bmatrix}_{M\times 1} . \tag{8}$$

2.1.2. *Total Rates.* Using d_i and b_{il}, total transition rates for each node were generated as

$$R_{in} = (d_i \mathbf{1}_{1\times N})_{M\times N} \circ X + (b_{il} N_q)_{M\times N} \circ X \tag{9}$$

where $\circ$ represents element-wise multiplication.

In order to find the total rate of change for the entire system, we re-added the rates. For example, for the total rate of change for each compartment in the entire network, we introduce R_i:

$$R_i = \begin{bmatrix} \sum_{i=1}^{N} R_{in}[1,i] \\ \vdots \\ \sum_{i=1}^{N} R_{in}[M,i] \end{bmatrix}_{M\times 1} \tag{10}$$

and for the total rate of change for the entire system, we introduced R:

$$R = \sum_{i=1}^{M} R_i[i] . \tag{11}$$

2.1.3. *Updating system status after an event.* The initial state for all nodes was generated according to Section 2.0.2. Because all random processes are Poisson processes, the assumption was made that the next event would occur in time δt:

$$\delta t = \frac{-\ln(\mathsf{rand})}{R} \tag{12}$$

where $0 \le \mathsf{rand} \le 1$ is a generated random number. During this event one of the nodes changes its status. We determined which compartment changed by drawing a sample among M compartments with probability distribution R_i; this compartment was called i_s.

Once the leaving compartment was identified, we wanted to know which node experienced the transition. Therefore, we drew a sample from N nodes with probability distribution $R_i n[i_s, :]$ (i.e., i_s row of matrix $R_i n$) and called this Node n_s.

To find the new status (compartment) of Node n_s, again GEMF randomly draws the new compartment j_s among M compartments with the following probability distribution:

$$p_{j_s}^T = A_\delta[i_s, :]^T + b_i[i_s] N_q[:, n_s] . \tag{13}$$

Drawing samples with given probability distribution is done with RNDDRAW function.

With δt, i_s, j_s, and n_s, GEMF had all necessary information to update the network status and apply required changes with the occurred event. However, GEMF had to update X matrix and the future rate of transitions.

Because Node n_s changed its status from i_s to j_s, we have:

$$X\left[i_s, n_s\right] = 0,\ X\left[j_s, n_s\right] = 1. \tag{14}$$

To update R_i, we subtracted the column in R_{in} that corresponded to Node n_s (i.e., $R_{in}\left[:, ns\right]$) and then we updated

$$R_{in}\left[:, n_s\right] = d_i \circ X\left[:, n_s\right] + \left(b_i N_q\left[:, n_s\right]\right) \circ X\left[:, n_s\right]. \tag{15}$$

Now we add $R_{in}\left[:, n_s\right]$ to R_i. Next if any of the old or new compartment are in influencer category in any layer, code should update N_q matrix. First, we find neighbors of node n_s:

$$N_{ln} = Neigh\left[l\right]\left[I_1\left[l\right]\left[n_s\right] : I_2\left[l\right]\left[n_s\right]\right] \tag{16}$$

$$WeightedNeigh = NeighW\left[l\right]\left[I_1\left[l\right]\left[n_s\right] : I_2\left[l\right]\left[n_s\right]\right] \tag{17}$$

(18)

Then we conducted the following steps for all these neighbors:

- If the old compartment i_s was an influencer compartment in layer l, we do the following removed n_s as their infected neighbors and recorded the weight of the edge. We also updated R_{in}. For n, the k'th neighbor of n_s was

$$N_q\left[l\right]\left[n\right] -= NNeighW\left[l\right]\left[NI_1\left[l\right]\left[n_s\right] + k\right] \tag{19}$$

$$R_{in}\left[:, n\right] -= NNeighW\left[l\right]\left[NI_1\left[l\right]\left[n_s\right] + k\right]\left(b_{il}\left[:, n\right] \circ X\left[:, n\right]\right) \tag{20}$$

  where $-=$ indicates subtracting to current value.

- If the new compartment j_s was an influencer compartment in layer l, we added n_s as their infected neighbors and recorded the weight of the edge. We also updated R_{in}. For n, the kth neighbor of n_s was

$$N_q\left[l\right]\left[n\right] += NNeighW\left[l\right]\left[NI_1\left[l\right]\left[n_s\right] + k\right] \tag{21}$$

$$R_{in}\left[:, n\right] += NNeighW\left[l\right]\left[NI_1\left[l\right]\left[n_s\right] + k\right]\left(b_{il}\left[:, n\right] \circ X\left[:, n\right]\right) \tag{22}$$

  where $+=$ indicates adding to current value.

We stacked n_s, j_s, and i_s into n_{index}, j_{index}, and i_{index}, respectively, and then we recalculated R_i and R and prepared for the next event.

[1] NETWORK SCIENCE AND ENGINEERING GROUP, DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, RATHBONE HALL, KANSAS STATE UNIVERSITY, MANHATTAN, KS 66502, USA

E-mail address: fft@ksu.edu