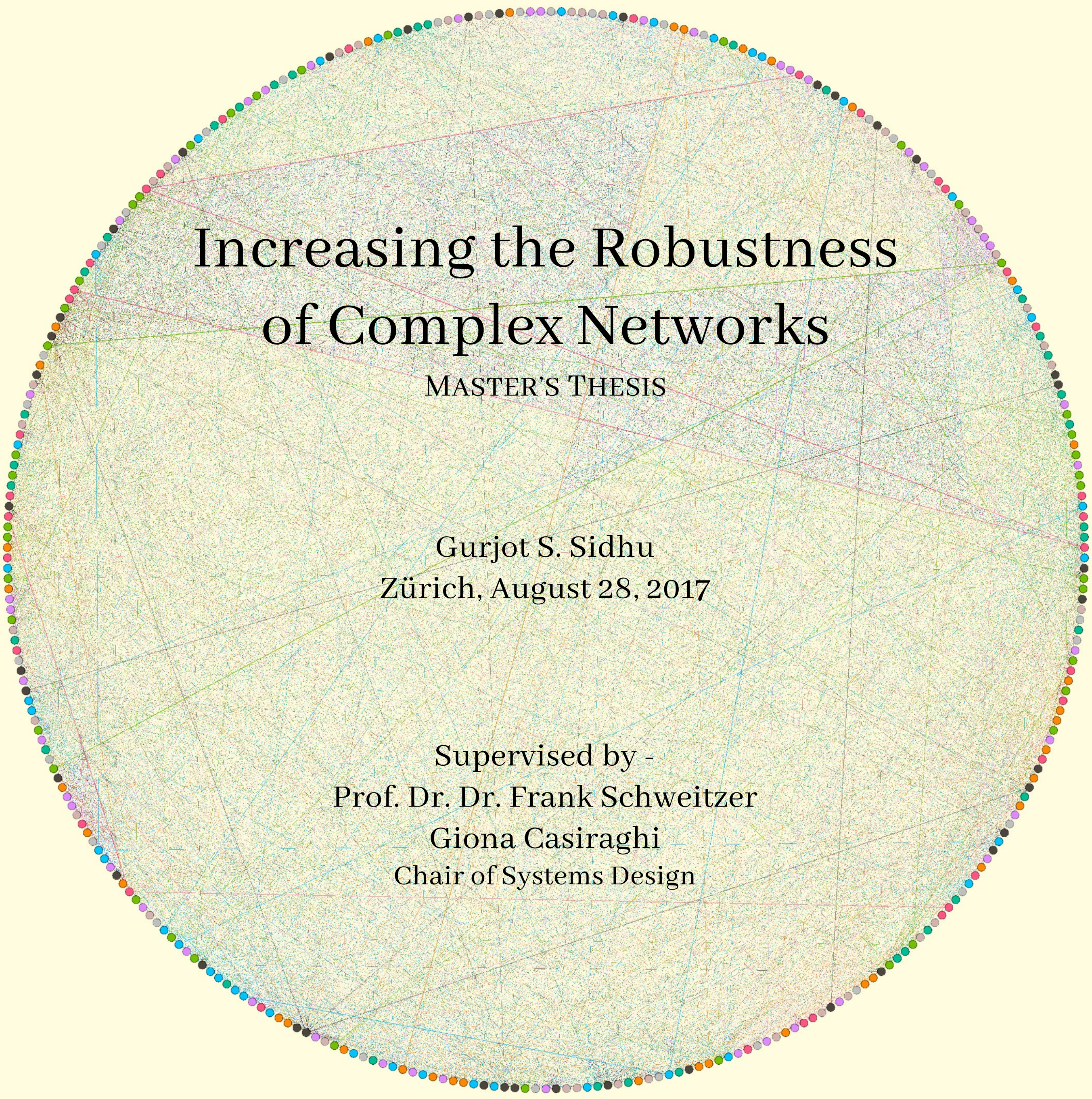
ETH zürich DPHYS



Increasing the Robustness of Complex Networks

MASTER'S THESIS

Gurjot S. Sidhu

Zürich, August 28, 2017

Supervised by -
Prof. Dr. Dr. Frank Schweitzer
Giona Casiraghi
Chair of Systems Design

If I have seen further, it is by standing on the shoulders of giants.
— Isaac Newton

ACKNOWLEDGEMENTS

I would like to thank Prof. Frank Schweitzer for graciously providing me with the opportunity to work in his esteemed research group on a topic of great personal interest to me. Despite being on a sabbatical, he kept himself involved in my progress and provided helpful guidance and advice. My sincerest gratitude to Giona Casiraghi for being the ideal supervisor by providing the perfect mix of competent suggestions and friendly curiousity that created the work environment integral to the success of this work. His constant counsel acted as a mental support that gave me the confidence to keep on keeping on. I would also like to thank Dr. Ingo Scholtes for taking the time to patiently assess my work and for providing his invaluable insights without which this thesis would have been as misleading as collecting time-aggregated results on a non-Markovian temporal network. I have greatly benefited on both a professional and a personal level from working at the Chair of Systems Design – my thanks to everyone involved in making this group a fun place to work in.

I am hugely indebted to my family, friends and colleagues for patiently listening as I ranted or raved about my work, and for providing the encouragement and support I needed to continue working.

Lastly, my gratitude to all the music artists and producers whose work kept me company as I worked on this thesis, and the online software development community that offered helpful solutions on each step of the way.

ABSTRACT

Human interaction networks have become an important focus of research as they can provide great insights into the workings of our society and also help develop unique strategies for prevention of epidemics. In this thesis we try to find strategies to modulate the speed of spread of an infection on a temporal network. For doing so it is important to understand the network topology and temporal structure. Though temporal networks have always been at the forefront of network science research, researchers have only recently shifted focus from a time-aggregated analysis to one that takes the non-Markovian dynamics of the network into account. We add to this pragmatic change with an analysis of a SocioPatterns data set of interactions in a high school. We find a number of key identifiers or interaction patterns of second-order temporal networks and are able to show that their non-Markovian dynamics occurs due to the existence of in-group memory. We show that ignoring this non-Markovian nature can lead to misleading results and at the same time confirm recent findings that memory can modulate the speed of an infection process. We validate these results with the help of an agent-based model. Our work adds to the existing body of literature by not just reaffirming the results of previous work but by also providing an algorithm for generating temporal data while preserving network topology. The algorithm and agent-based model developed in this thesis can be used for creating reliable null models for analysis of temporal networks thereby greatly simplifying future investigation.

Tags: complex networks, temporal networks, non-Markovian process, epidemic modeling, agent based model

CONTENTS

1 INTRODUCTION

The nature of interactions between individuals has become an important subject of interest in a number of research fields in the recent years given its importance in shaping social networks and its influence on the flow of information or infectious diseases in a population. Scientific collaborators have come together to establish large publicly available data sets such as SocioPatterns and MIT Reality Mining in order to facilitate this research.

In finding strategies to modulate the speed of spread of an infection or information on a network of human interactions, it is important to first understand the network structure from a topological, and more importantly, a temporal perspective. Though temporal networks have been at the forefront of network science research since the inception of the field, the non-Markovian nature of these networks has only recently gained importance and it appears to drastically shift the outlook from hitherto-employed static, time-aggregated analysis to a more involved one.

Infectious diseases spread by means of repeated contacts between individuals. These interactions are influenced by the relations between the individuals. By modifying these relations we can figure out how to make the network more, or less susceptible to the disease/external shock. We investigate whether it is possible to control the robustness of a system against epidemics by modifying these relational ties. The spread of an infection also relies heavily on the temporal ordering in the network. From the data available to us, we extract some quantifying measures of interaction or the interaction patterns and find out more about the temporal dynamics in order to be able to create better strategies for controlling disease spread.

1.1 ABOUT THE THESIS

The project consists of two main parts. In the first part we analyse two datasets of repeated contacts between individuals and we quantify their robustness against an epidemic spread using the SIR model. The initial aim of the thesis is then –

> to find strategies that can influence the robustness of a complex network by modifying the relational ties between individuals.

In the second part, we investigate the temporal dynamics of the network and develop an algorithm and an agent-based model for the same. This added aim can be posed as –

> to analyze the temporal nature of human interactions and probe its 2nd order non-Markovian character.

This manuscript consists of 7 chapters and an appendix. Chapter 2 introduces the basic background needed to follow along with this thesis. Related literature and publications are referred as well for further reading.

In Chapter 3 we cover the first part of the thesis where we run simulations of infection processes on empirical data and try some basic strategic intervention techniques. We discuss the results of the same at the end of the chapter.

In Chapter 4 we develop the Synthetic Timestamp Generator (SynTG) algorithm that can be used to generate temporal data for a static network while preserving its topology. To this effect, we analyze the empirical data for interaction patterns and use them as the base for the algorithm.

Chapter 5 describes the agent-based model (ABM) we developed for the analysis of non-Markovian dynamics of temporal networks. The ABM is a grounds-up model that creates both the topology and the temporal structure of the network.

We discuss the findings regarding non-Markovian dynamics and compare our work with current research in Chapter 6. We also discuss the drawbacks and limitations of our work along with suggestions for future work in this chapter.

Chapter 7 ties the work together and concludes the thesis, following which we offer some useful programming tips and pieces of code in Appendix A.

2 BACKGROUND

This chapter provides a general description and minimum introduction to the various concepts and tools (of both mathematical and computational nature) used in this thesis. The following sections will hopefully be self-contained and will act as a springboard for understanding the results presented in this thesis and their relevance in the broader network research paradigm. Wherever possible references are provided for further consultation.

2.1 DESCRIPTION OF DATA SETS

In this thesis we rely on the open-source data sets provided by SocioPatterns[1] – a collaboration between international researchers and developers. We make use of two data sets – *'High school contact and friendship networks'* (HSI) [1] and *'Contacts in a workplace'* (WPI) [2].

2.1.1 *Interactions In A High School*

The HSI data set corresponds to the contacts and friendship relations recorded between students in a high school in Marseilles, France, in December 2013. From the original research paper [1] –

> "The data collection concerned high school students of specific classes called "classes préparatoires" in Lycée Thiers, Marseilles, France. These classes, specific to the French schooling system, gather students for studies that take place for two years after the end of the usual high school studies. They study in a high school environment but are de facto mostly separated from the "regular" high school students: their classes are located in a different part of the high school building and they typically take their lunches separately. They constitute thus an almost closed population with few contacts with the outside world, at least during workdays. At the end of these two years, students go through competitive exams yielding admission to various higher education colleges."

The students were divided into nine classes based on different specializations – three classes of type "MP" (MP, MP*1, MP*2) which focused on mathematics and physics, two of type "PC" (PC and PC*) that specialized in physics and chemistry, one of type "PSI" (PSI*) for engineering studies, and three of type "BIO" (2BIO1, 2BIO2, 2BIO3) that focused on biology.

Data was collected using RFID sensors developed by SocioPatterns. [3, 4] The human body does not allow the radio frequencies used by the sensors to pass through so the sensors only recorded an interactions between two individuals if they were facing each other at a close range (< 1.5m). The sensors detected 188,508 face-to-face interactions between 327 consenting students over the course of 5 days during school working hours with a time resolution of 20 seconds.

In addition, students were asked to fill out surveys where they reported their friendship circles. Some students even permitted access to their Facebook profiles for collection of friendship data. In both cases, however, the sample size was smaller than the interaction data.

1 SocioPatterns website: www.sociopatterns.org

2.1.2 *Interactions In A Work Place*

This WPI data set corresponds to interactions recorded between individuals working in an office space in Saint Maurice, France. From the original research paper [2] –

> The study took place in one of the two office buildings of the InVS, located in Saint Maurice near Paris, France, and lasted two weeks. The building hosts three scientific departments – the Direction Scientifique et de la Qualité (DISQ, Scientific Direction), the Département des Maladies Chroniques et des Traumatismes (DMCT, Department of Chronic Diseases and Traumatisms) and the Département Santé et Environnement (DSE, Department of Health and Environment) – along with Human Resources (SRH) and Logistics (SFLE). DSE and DMCT are the largest departments, with more than 30 persons each, DISQ and SRH consist of around 15 persons, and finally logistics consists of only 5 persons. DISQ, DMCT and SFLE share the ground floor, while DSE and SRH are located on the first floor.

Data was collected using the same RFID sensor technology as in the HSI data set. 100 individuals wore the trackers for 14 days resulting in 9,827 interactions again with a minimum time discretization of 20 seconds. The only metadata available in this case was the department the individuals belonged to.

2.2 COMPLEX NETWORKS

Networks or graphs are a theoretical construct used to describe relations between entities. A simple example would be to consider a family tree with two generations – the parents and the children. Each individual in the family acts as a *node* and the fact that they are *related to each other* (since they are a family) is represented with a distinct line connecting all the pairs of nodes called an *edge*. Assuming our family consists of four individuals (mother, father, son and daughter) we get 4 nodes and 6 edges as shown in Fig. 2.1.

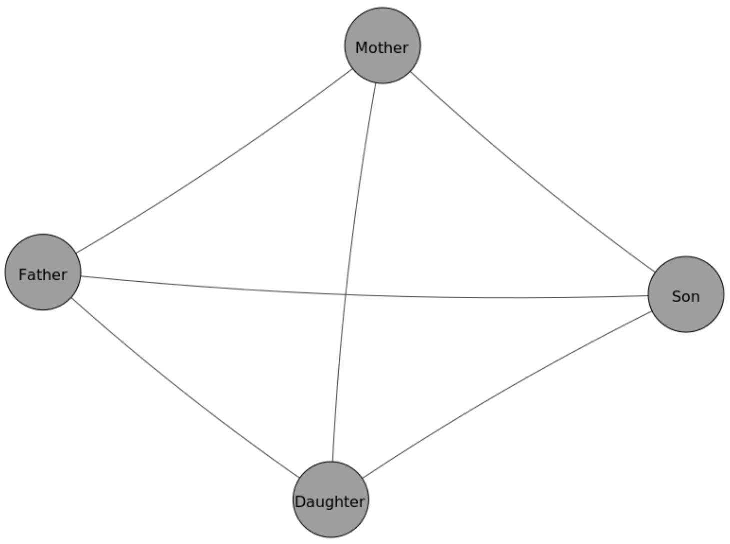


FIGURE 2.1: An example of a small graph.

For our current intents and purposes we will assume that all edges are bi-directional, i.e., if an edge exists between the father and the son then they are both related to each other. This is called an *undirected edge*. The opposite case of a *directed edge* can be understood by considering another network that records the phone call interactions within this family. So if the mother calls her daughter, we get a directed edge from the mother to the daughter since considering it to be an undirected edge would mean that there is no difference between the mother calling or the daughter calling.

Another interesting property to add to this network would be to add a timescale to it. With this not only do we know whether two nodes are connected by an edge but also *when* this connection was formed. So the edge connecting the parents would have a timestamp of the date of their wedding (say in the year 1985), and the edges connecting each parent to their daughter would be time-stamped with her date of birth (say in the year 1990). This is what we call a *temporal network* – a network with temporal data. While on the topic, it is useful to consider a relevant idea. Can an edge between the mother and daughter exist if we are looking at the year 1980? No, it cannot! Because the parents aren't even married yet and thus the daughter hasn't born yet either. From this we can conclude that the *ordering of edges*, i.e., what comes before what, is very important.

Now if we expand our scope from one family to an entire city, we will see thousands of such small networks (See Fig. 2.2). However they will not all be independent of each other since some nodes from different families will be related to each other. For example, the network representing the father's brother's family will be connected to this family's network through the edge between the father and his brother. If we expand our scope further to a country or even to the whole world, we see that things slowly start to get complex – some very interesting patterns start to emerge. There is no strict definition of what constitutes a complex network and what does not apart from the type of features that they exhibit. Complex temporal networks are built on non-trivial topological features such as heavy-tailed degree distribution and inter-event times, bursty nature of edge activation etc. [5–7]. These properties will be discussed in detail in the chapters to follow particularly in Chapter 4.

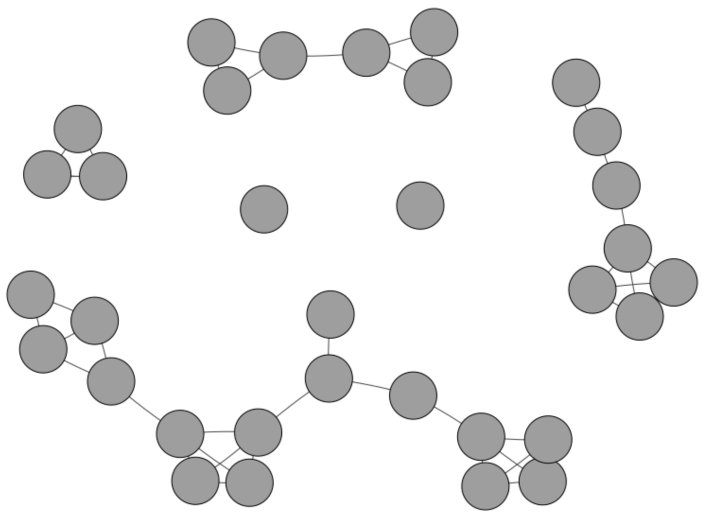


FIGURE 2.2: Small graphs connect together to form larger graphs.

Research around the 'six degrees of separation' phenomenon [8–10] has shown that on an average any two humans in the world are just 6 acquaintances away from knowing each other. Networks showing such behaviour are termed *small-world* where 6 acts as the scale of the network. The opposite class of *scale-free* networks refers to the case where no such defining scale exists. These networks contain some nodes that have very few connections and others that have hundreds or even thousands of connections. The best-selling book by Clay Shirky titled *'Here Comes Everybody'* [11] provides a wonderful introduction to this topic with many well-explained examples from real life data sets such as the number of Wikipedia edits made by users or the number of photos uploaded on Flickr.

Studying social networks (both on-line and off-line) can be very useful in not only gaining a better sociological and psychological understanding of our society but in also accelerating or containing the spread of an information or infection. [12–15] Significant research has been done in the field of epidemic modelling on complex networks [16–19]. Up until a few years ago, most of the research looked at scale-free, static networks. However the static, time-aggregated representation often misses on important properties of the network that only emerge when the temporal nature is put into consideration. Masuda & Holme make important statements in this regard based on a review of research on network epidemiology. [20]

This acts as a starting point for our work in this thesis. We study the temporal nature of complex networks and see how it effects the spread of an infection on the network.

2.3 COMPARTMENTAL MODELS IN EPIDEMIOLOGY

Mathematical epidemiology plays the role of modelling the spread of infections. It is an old and established field with significant success in predicting, preventing and controlling global epidemics. [21–24] The most widely-used method in epidemic studies is that of dividing the population into sub-populations or compartments. Many models have been built on this method and are called compartmental models in epidemiology.

The most basic model is the Susceptible-Infected model or the SI model. In this model the population is divided into two compartments – those who are susceptible to the infection (S) and those who are already infected (I). The susceptible population can get infected with a probability p_1. By considering the dynamics of birth and death the model can be easily represented in terms of differential equations and the solution can be found either analytically [23] or using numerical methods.

However, the SI model does not consider the chance that an individual once infected can recover from the infection. To add that possibility we expand to the Susceptible-Infected-Recovered model or the SIR model where an added compartment for recovered population is considered. In this case an infected individual can recover based on a probability p_2. Fig. 2.3 provides a simple representation of this model.

The SIR model has been applied to a large number of diseases where recovery confers a life-long immunity, most famously to measles as shown by Shulgin, Stone & Agur [25].

Some of the other possible compartmental models are the Susceptible-Infected-Susceptible (SIS) model where the infected individual becomes susceptible again immediately after recovery and the SIRS model where the individual has a temporary immunity after recovery. All these models and others offer their own unique advantages, each fitting a certain class of diseases.

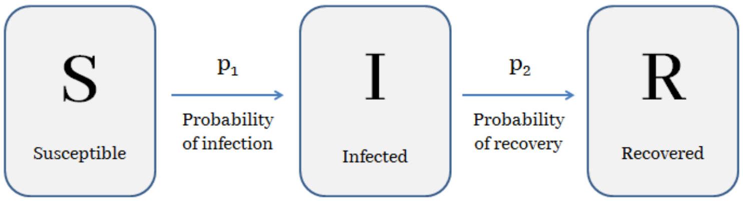


FIGURE 2.3: A visual representation of the SIR epidemic model.

Barabási provides a great introduction on how to use epidemic models in a network representation in *Network Science*. [26]

2.4 GHYPE - MULTIPLEX NETWORK REGRESSION

Generalized hypergeometric ensembles or gHypE is a recently developed approach for model selection and statistical hypothesis testing of data on complex networks. [27] As shown in [28] the `HSI` data set can be easily analyzed for judging the influence of relational layers on the interaction. Essentially it tells how likely is an interaction between two students from the same class (or gender, topic, etc.) to occur as compared to those from different classes. With that we can find which layers have a significant effect on the interactions and which do not.

gHypE was available to us as a package in the R programming language. We utilized it mainly for its ability to create synthetic realizations of the original network (created from the empirical data set). Realizations are almost identical copies of the original network in the sense that they preserve all its topological properties. One can also add certain control parameters in order to create a modified realization. For example, we removed the effect of certain layers from the network and the realization we then created was based solely on the effect of the rest of the layers.

2.5 PATHPY

PATHPY [29] is a software package available in the Python programming language.[2] It allows a unified approach to the analysis of temporal networks and pathways. Its theoretical foundations are based on the higher-order network abstractions and temporal centrality measures developed in [30–32].

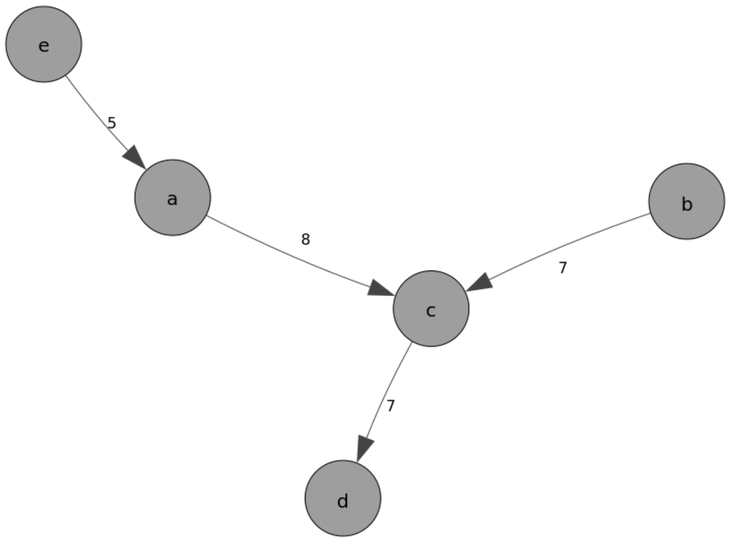


FIGURE 2.4: An example of a first-order network.

What constitutes a path in a network? Consider a simple network as shown in Fig. 2.4. Here the edges $(a \to c)$ and $(c \to d)$ make a path of path length 1. That means that all edges in a network are paths of path length 1.[3] The numbers written on the edges represent the edge-weight or how many edges exist between two nodes. As can be seen, $(a \to c)$ has an edge-weight of 8 meaning that a interacted with c eight times. Now if we consider the path $(a \to c \to d)$ then it is a path of path length 2. Longer path lengths can thus be simply deduced for static networks.

2 A detailed tutorial is available at – https://ingoscholtes.github.io/pathpy/tutorial.html
3 Considering a first-order network representation.

However, now we invoke the little aside mentioned in Section 2.2 when talking about temporal networks. If we consider the ordering of the edges the story begins to change. Suppose the network shown in Fig. 2.4 represents an interaction-infection network. The path $(a \rightarrow c \rightarrow d)$ can be considered to be of path length 2 if and only if $(a \rightarrow c)$ is immediately followed by $(c \rightarrow d)$. If a gives c the common cold, the infection will only reach d if c interacts with it within a few days. Otherwise c will have already recovered and any future interaction will not lead to d getting infected. In that case $(a \rightarrow c \rightarrow d)$ will be reduced to two distinct paths of path length 1 each.

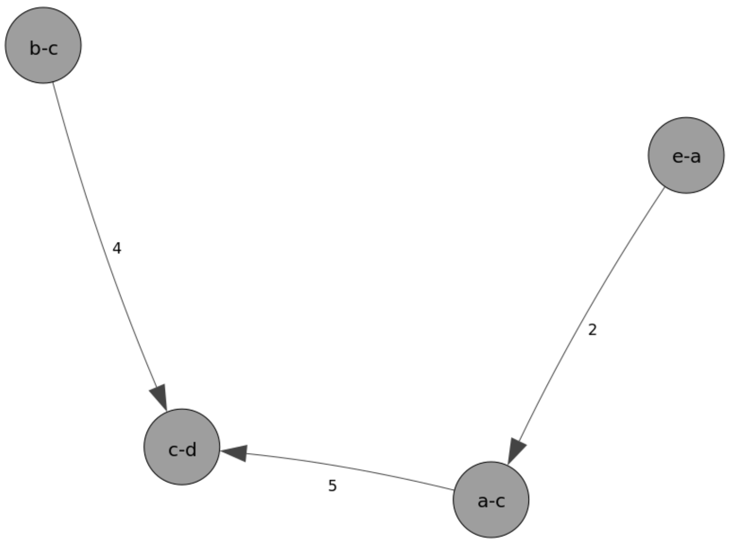


FIGURE 2.5: An example of a second-order network.

To effectively study this we use what is called a 2nd order network representation. In this each edge (say $(a \rightarrow c)$ and $(c \rightarrow d)$) becomes a node and an edge connects these nodes if and only if one path leads to the other. This is shown in Fig. 2.5. On comparing with Fig. 2.4 it can be seen that even though a interacted with c 8 times and c with d 7 times, only 5 interactions of the former were immediately followed by the latter. Thus only 5 cases have a path length of 2 and the rest are just distinct paths of unit path length.

2.5.1 *Markov Dynamics*

Without going into a detailed theoretical background of the subject we will provide a limited working description in relation to our use-case scenario instead. Consider the interaction network shown in Fig. 2.4. If this network portrays first-order Markov dynamics then it means that the fact that c chose to talk to d is a choice it exerted based solely on its present state. In other words, it is a random process – the history of a node has no effect on its future interactions. For such a network a first-order representation is usually good enough for temporal analysis.

However, if the past plays a role in determining the future, i.e., there exists a certain *memory* in the network, then the network exhibits second-order Markov dynamics. This can be interpreted in the following way. The fact that c spoke with d is a result of the fact that it had spoken to a (and b, e ...) before it.

The existence of memory in a network can completely change its temporal dynamics [33] as we find out in this thesis. (Spoiler alert!)

2.6 GRAPH TOOLS

We use the `igraph` package for the bulk of the network analysis. However, we use the `graph_tools` package for creating visual simulations of the infection process. The `graph_tool` documentation has a detailed guide for creating animations that was incredibly helpful in this regard.[4] We also make use of Gephi which is an open source software for graph visualization. The network image on the front cover of this thesis was created using Gephi.

2.7 PROGRAMMING LANGUAGES

The bulk of the work in this thesis was done in Python 2.7[5]. Apart from the core pre-installed modules, the essential trio of scientific programming – `SciPy`, `NumPy` and `matplotlib`, were used. The `igraph`, `graph_tool` and PATHPY packages mentioned in the previous sections were also run in Python. The pseudocode provided in the later chapters are also based on Python syntax.

The gHypE package was used in R[6]. The typesetting of this manuscript was done using LaTeX.

4 'SIRS epidemics' animation tutorial: https://graph-tool.skewed.de/static/doc/demos/animation/animation.html#sirs-epidemics
5 Python programming language: www.python.org
6 R Project: www.r-project.org

3

EPIDEMIC MODELLING

In this chapter we will work with the data sets listed in Section 2.1. We will provide a general statistical and graphical analysis of the data, following which we will cover the epidemic modelling algorithms used for studying the spread of infection on the networks generated from the data. Finally we will present the results of the infection process on the empirical and synthetic networks.

3.1 DATA ANALYSIS

3.1.1 *Statistical Analysis*

Both the `HSI` and `WPI` datasets are statistically very similar to each other. Before we dig into the network properties we would like to superficially differentiate between the two.

`HSI` consists of contacts between students in a high school. The students classify themselves under 3 genders and are divided into 9 classes in the school which fall under 4 major topics. The related numbers are provided in Table 3.1.

Furthermore, we have data on the reported friendships and Facebook connectivity albeit this data is only available for a limited number of students. While 135 students filled in the friendship survey, only 17 students provided access to their Facebook local network. This already casts doubt on the validity of this data, particularly the Facebook friendship relations. Casiraghi's multiplex network regression analysis of this dataset confirms this hypothesis [28] by showing that the Friendship network layer is superfluous or non-significant when studying the effect of relational layers on interactions. Thus, we choose to work without this layer.

On the other hand, the `WPI` data set, though very similar in its origins, lacks additional metadata about the individuals involved in the study beyond their departmental classification. The details are shown in Table 3.2.

There is not much to deduce from this amount of information so we move to the next step, which is to plot the networks, whereupon we will be able to see how the interactions play out in the networks.

Label	Classification and Number of students									Total
Class	MP	MP*1	MP*2	PC	PC*	PSI*	2BIO1	2BIO2	2BIO3	327
	33	29	38	44	39	34	36	34	40	
Topic	MP	PC	PSI	BIO						
	100	83	34	110						
Gender	M	F	Unknown							
	175	145	7							

FIGURE 3.1: `HSI` data set.

Label	Classification	Number of workers
Department	DISQ	15
	DMCT*1	30
	DSE*2	38
	SRH	13
	SFLE	4
	Total	100

FIGURE 3.2: WPI data set.

3.1.2 *Network Topology*

At this point it would be interesting to see how individuals interact between different labels. For example, do students of `class` 2BIO1 interact mostly with students from the same `class` or `topic` (BIO)? How significant is the cross-label interaction?

We first plot the HSI interaction graph using Gephi. We choose a circular layout for the nodes arranged by `class`. A gray-scale map provides an easier view of the interaction edges as seen in Figure 3.3.

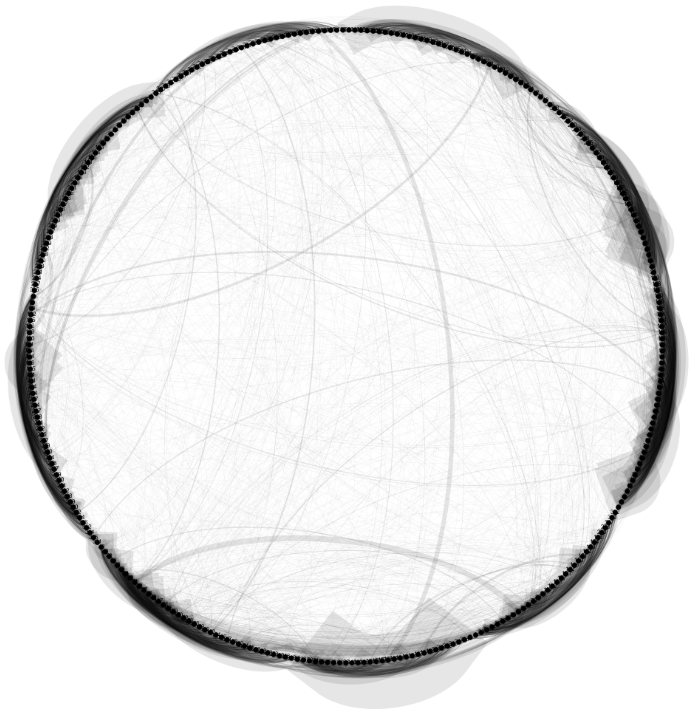


FIGURE 3.3: HSI interaction network graphed in a circular layout arranged by `class`.

The class boundaries can be easily deduced from the thick intra-connectivity at the circumference of the circle. However there is also a fairly strong representation of inter-class connectivity as shown by the long edges inside the circle.

Similar behaviour is witnessed for the other two layers of `topic` and `gender` as well. From this we can deduce that though intra-layer interaction is strong, it is not an opaque boundary.

If we now look at the Friendship network (Figure 3.4), we see a clear existence of cliques, some even isolated entirely from the main network. Again, despite the cliques, there is connectivity between them as seen before.

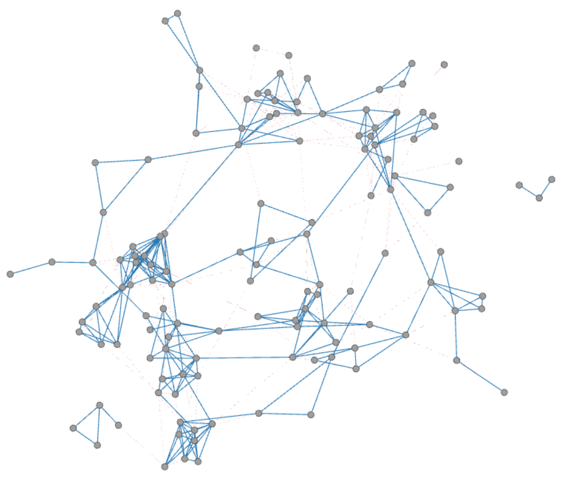


FIGURE 3.4: Friendship network from the HSI data set.

3.2 MODELLING ALGORITHM

In this section we will describe the algorithms used for running epidemic models on the networks. We primarily make use of a Susceptible-Infected-Recovered (SIR) model, however, two other models – namely, Susceptible-Infected-Susceptible (SIS) and Susceptible-Infected-Recovered-Susceptible (SIRS), are made use of as well. The general idea of all three models is the same and is described briefly in the previous chapter under Section 2.3.

We start with the description of the SIR model and will then generalize it to SIS and SIRS. The algorithm we developed for this is similar to the Temporal Gillespie algorithm presented by Vestergaard & Génois [34], however, we deem it as a mere coincidence as the knowledge of the existence of this algorithm was gained after we had developed our own.

The outline of the SIR process is as follows. At each timestep –

1. Add new edges representing the interactions that take place in this timestep.
2. Manually infect nodes. (Only during the seeding process)
3. Run the infection process wherein the infected nodes infect susceptible nodes with a probability p_1. Each infected node stays infected for *at least* X time-steps.
4. If an infected node has been infected for over X time-steps, run the recovery process wherein the infected node can recover with a probability p_2.
5. Delete all edges to get an empty network for the next iteration.

In the SIS process, instead of recovering in Step 4 the node becomes susceptible again. The SIRS process has an extra penultimate step where the recovered node becomes susceptible again with a probability p_3.

The epidemic process stops once there are no more susceptible nodes left in the system ($N_{susceptible} = 0$) or if there are no more infected nodes left ($N_{infected} = 0$), or if the temporal data runs out, whichever occurs earlier. The percentage of network that gets infected in the process ($Infectiousness$) can be calculated as follows –

$$N_{susceptible} + N_{infected} + N_{recovered} = N_{total} \tag{3.1}$$

$$Infectiousness = \frac{N_{infected} + N_{recovered}}{N_{total}} * 100 = (1 - \frac{N_{susceptible}}{N_{total}}) * 100 \tag{3.2}$$

An important note to be made here: though the algorithm is seemingly very easy to understand, many complications may arise due to incorrect accounting of nodes. To ease this process we make use of the `vertex attributes` functionality of `igraph`. The `graph_tool` equivalent is called `vertex properties`. This functionality treats each node as an object (in the OOP sense), allowing us to add custom properties with values of type `integer`, `float`, `bool` or `string` to it. This way we can keep track of which nodes are susceptible, infected or recovered, how many time-steps has a node been infected for etc., apart from the basic metadata that the data set provides us with.

As mentioned before in the previous chapter we create two scripts – one in `igraph` that we use to run our simulations, and an equivalent translation of the code in `graph_tool` that runs visual simulations allowing easier debugging (and cool visualizations!). The visual simulation also helped gain an intuition for the way the spreading process runs on the network.

Our aim here is three fold –

1. Run the SIR simulations on the `HSI` network for different control parameters and fix certain benchmark values (of the control parameters) for the system.
2. Create a synthetic realization network using gHypE and SynTG, and compare with the original network for the same benchmark values.
3. Create modified realization networks (that is, apply strategies) and compare.

There are a lot of possible ways to tweak the controls of this algorithm. Apart from the probabilities p_1, p_2 and p_3, we can change the number of manually infected nodes in the seeding process, and/or the minimum duration that a node remains infected. An interesting modification would be to change the way we iterate our temporal data. The usual (physically relevant) way would be to traverse linearly (that is, $1, 2, 3...$), but we could get interesting results if we instead iterated randomly (for example, $15, 62, 97, 38...$). And then there is also the choice between SIR, SIS and SIRS models.

We provide the values of the control parameters that we used as a benchmark in Table 3.5. There is little rigorous reasoning behind why these values were chosen apart from intuition.

Parameter	Value
Probability of getting infected, p_1	0.2
Probability of recovering, p_2	0.15
Probability of becoming susceptible after recovering, p_3	0.1
Minimum infection time	5 time-steps
Number of seed nodes	2 per time-step
Length of seeding process	10 time-steps

FIGURE 3.5: Benchmark values of control parameters for the epidemic process.

3.3 RESULTS

A number of metrics can be collected for the infection process, such as, the evolution of the percentage of individuals in different compartments (susceptible S, infected I, recovered R), the basic reproduction number (R_0) [35], the time taken for infecting a certain percentage of the population etc. However, we focus on a single metric for quantifying the *robustness* of the system – the percentage of network infected at the end of the infection process, or the *infectiousness* (I_0). If a network showcases a lower value of I_0 as compared to another then it is said to be more robust.

The control parameters that we modified were – the number of interactions per timestep; the mode of iteration of temporal data (linear and random); the infection probabilities (p_1, p_2, p_3); the time scale, placement and strength of the seeding process; and the type of epidemic model (SIR, SIS, SIRS).

Unless explicitly stated, we run an SIR epidemic model.

3.3.1 *Empirical Network: Static Analysis*

As should be expected, the static network is less robust to the infection process in comparison to the temporal network as can be seen in Figure 3.6. This is because the infection process completely disregards temporal correlations in a static network. What that means is that an infected node i will try to infect *all* its connected nodes at *each* timestep in a static network whereas in a temporal network a node j can get infected if and only if it was in contact with an infected node (say i) in the given timestep.

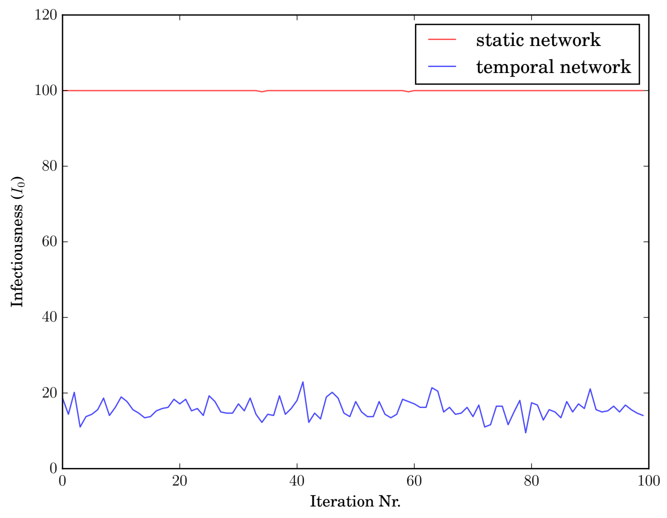


FIGURE 3.6: The percentage of network infected at the end of the SIR infection process (I_0) is considerably higher for a static network as compared to a temporal network. Here $p_1 = 0.2$ and $p_2 = 0.15$.

The static network shows very little deviation in its behaviour when the control parameters are changed. The system's robustness is so low that even when the recovery probability p_2 is as high as 0.9, I_0 is still close to 100%. Similarly, all other parameters have little to no effect on I_0.

If we move on to the Friendship network, we see similar behaviour albeit with lower values of I_0 (̃90%) owing to the lesser density of the network. Another interesting result that we notice in this network is that at times the infectiousness is as low as 1%. This happens when the seed node is placed in one of the independent cliques in the network (seen before in Figure 3.4).

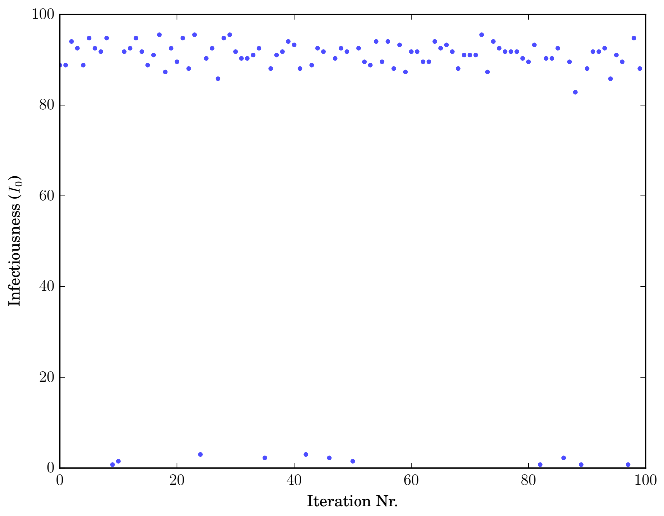


FIGURE 3.7: Infectiousness over 100 iterations in the Friendship network subject to an SIR process with $p_1 = 0.2$ and $p_2 = 0.15$.

Thus we conclude that network topology has a role to play when estimating the robustness of a static network.[1]

3.3.2 *Empirical Network: Temporal Analysis* [2]

The first major consequence of adding the temporal nature of the interaction process to the infection process is that the network never reaches values of I_0 as high as 100%. For our benchmark values of $p_1 = 0.2$ and $p_2 = 0.15$ we get an average 15.8% infectiousness for the network (as seen in Figure 3.6). This owes to the fact that chances of a node getting infected are limited by whether it comes into contact with an infected node or not.

With respect to a change in the control parameters, the results are mostly intuitive. Parameters that can be intuitively thought to have an enhancing effect on the infectiousness, like increasing p_1 or decreasing p_2, or increasing the number of seed nodes, have that same effect.

The most interesting behaviour though is seen when the mode of iteration of temporal data is switched from linear to random. As seen in Figure 3.8 we see that the random mode shows considerably higher infectiousness than the linear mode. Physically, this is irrelevant since time can only run linearly (anything otherwise would break so many laws of physics it won't even be funny), however, it confirms a result about the Markov dynamics that we will elucidate on in the next section.

The runtime of the temporal analysis averages around 40 time-steps, which though only about 15 minutes in real time, tells us that it takes less than a quarter of an hour for an infection (with reasonable spreading probabilities) to spread to 15% of the population! The consequences are far-reaching from a modelling perspective since this leaves us very little room to try intervention-based strategies.

1 We do not invoke any further analysis in the static network case since our target is to modify the robustness of a temporal network. The static network analysis did provide a useful starting point nonetheless.

2 It should be noted that all the analysis of temporal nature is done with the interaction network only. The Friendship network does not have a temporal attribute to it.

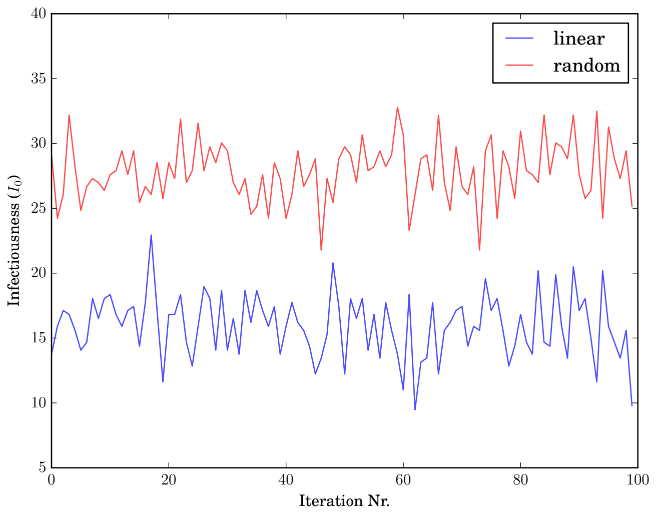


FIGURE 3.8: Iterating temporal data in a random order shows an increased value of I_0 in comparison to the original linear order.

Figure 3.9 provides a visual representation of how the infection process develops in the system.

3.3.3 *Synthetic Network: Temporal Analysis*

We create synthetic realizations of the temporal network using gHypE and generate temporal data for the synthetic networks using the SynTG algorithm. The results from Section 3.3.2 are re-created entirely with the difference that I_0 has a lower value than before (~10% instead of 15.8%).

We tried two types of strategies for modulating the infectiousness of the system. One affected the entire network topology and the other had a more focused effect. The first strategy involved removing the control for a particular layer(s), i.e., creating a network where the interactions occur only based on the effects of the class and sex layers and not the topic layer. We tried this for different layers and by removing multiple layers at once or one after the other, and each time the infectiousness of the system was either higher or lower than the original. This randomness in the results made it very difficult to rank the effects. We could not reliably say that removing a particular layer will have the greatest effect on the infectiousness because each new iteration of the process gave new results.

In the other strategy we tried to create hubs in the network by selecting a node and artificially increasing its propensity to talk to all other nodes to 1 (maximum). This way every node interacts with this node thereby making it a hub. We tried it for every single node but the results were too random to reliably to pin down any meaningful hypothesis.

The randomness in the results arises from the randomness implicit in our SynTG algorithm. Each time we generate the data for a particular network topology we get a new data set. We tried mitigating this random effect by averaging the results from thousands of iterations and running a t-test to check for statistical significance. But since the network space in which the same topology can have different temporal order is practically infinite we were unable to make any headway.

We also tried a different approach to the problem wherein we tried to increase the runtime of the infection process so that we could develop some intervention strategies. To this effect we made use of SIRS and SIS epidemic models as well as modifications such as ramped infection

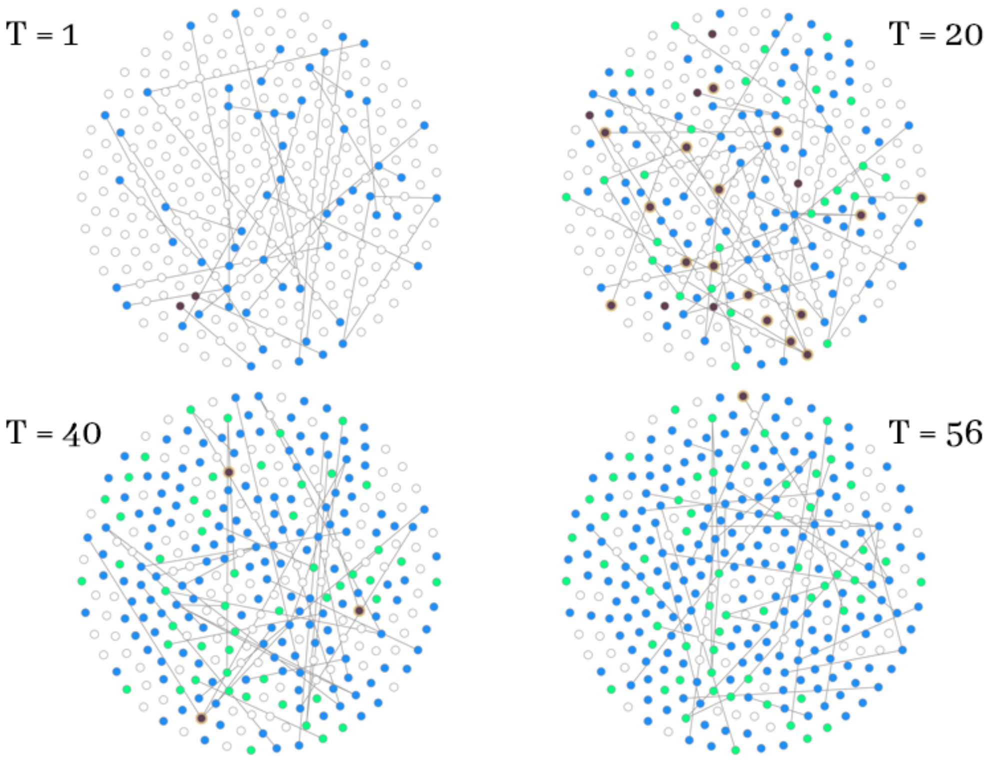


FIGURE 3.9: A visual representation of the infection process. This particular simulation ran for 57 time-steps. The nodes coloured red are infected, blue are susceptible and green are recovered. The nodes in white have not entered the system, i.e., engaged in an interaction, yet.

probability p_1, increasing p_3 to an unnatural 0.99 or making nodes stay infected for at least 20 time-steps, there is little change in the infectiousness and the infection process did not last longer than 100 time-steps. You only get infected if you are at the right place at the right time.

This convinced us that we needed to investigate the the network dynamics more rigorously in order to be able create more effective strategies.

4 SYNTHETIC TIMESTAMP GENERATOR (SynTG)

When we create synthetic realizations in gHypE, it only provides us with static networks, or more precisely, just adjacency matrices. We need temporal data for each realization in order to effectively study the dynamics of infection spread. One might näively imagine that the temporal data from the original `HSI` dataset can be used for this purpose. However, that is not the case since not only do the number of links between nodes change, the links themselves also change in every realization that we draw. Thus, we need to create an algorithm that uses the original temporal data as a truth value to create synthetic temporal data for a given realization network.

Before we proceed with thinking about the logic of the algorithm, we must first analyze the `HSI` dataset and look for certain interaction patterns that we might feel important for recreating the temporal data.

4.1 FINDING INTERACTION PATTERNS

Instead of a node-level approach we choose to focus on the patterns created by pairs of nodes (i, j) (the edges), i.e., we adopt a higher order perspective of the network.

To begin with, we find out that though there are a total $53,301$ pairs of interaction possible among 327 nodes[1], only $5,818$ pairs actually interact, of which $3,324$ are of the same sex, $4,977$ are of the same topic and $4,035$ of the same class.

We will now try to answer the following three intuitive questions –

1. How many times does a given pair interact?
2. How often does it interact?
3. How long does it interact for?

We first look at the distribution of the degree of edges.

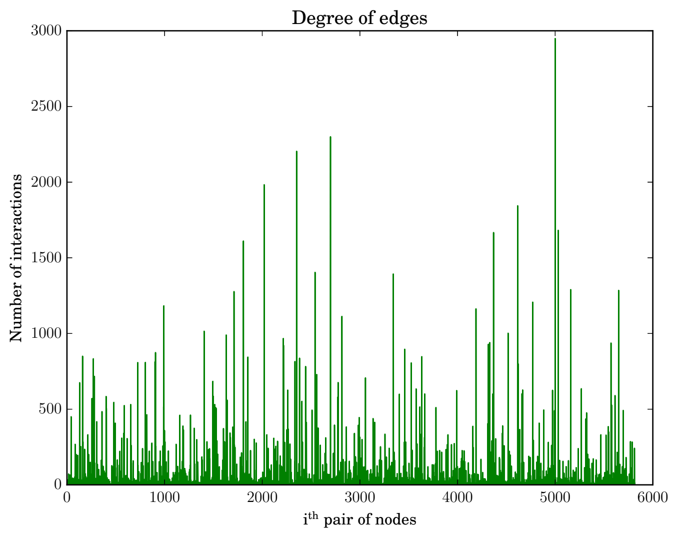
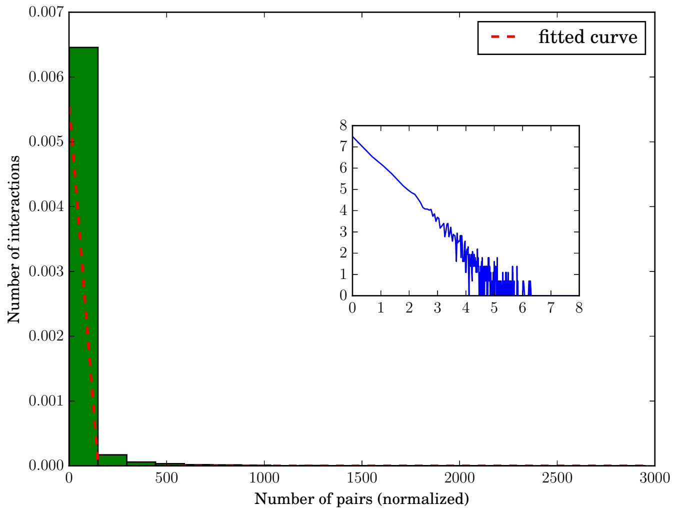


FIGURE 4.1: Number of interactions for each of the 5818 pairs of nodes.

1 Number of pairs = $^{327}C_2 = \frac{327\times326}{2} = 53301$

Number of interactions	Number of pairs
1	1787
2	694
3	442
42	13
100	1
1983	1
Maximum number of interactions	2949 (1 pair)
Mean number of interactions	32.40
Median number of interactions	3

(a) Statistics on the degree of edges.



(b) Histogram of the degree of edges with a best fit curve.

FIGURE 4.2: The distribution follows a heavy-tailed power law as seen in 4.2b inset.

As is common for most human interactions, there are some pairs that interact a lot (greater than 1000 times even) but most pairs interact only a handful of times. Some statistics are listed in Table 4.2a. It is interesting to note that while the average number of interactions is 32.40, the median value is a meager 3, a clear indication that the distribution is far from Gaussian.

Upon plotting the histogram it can be seen that the data is skewed to be bottom heavy. The fitted curve (in red in Fig. 4.2b) appears roughly exponential but on plotting a log-log plot the heavy-tailed power law nature of the distribution becomes apparent. Such heavy tails are a common occurrence in human interaction related datasets. [2]

Now that we know how many times the pairs interact, we move on to the final questions – how often do they interact and for how long does a typical interaction last?

To do so we run the code snippet presented in Listing 4.1 on the data. The logic is simple – it calculates the inter-event activation times (what we call `sporadicity`) of each interacting pair and stores them in a dict object with each pair tuple acting as a key.

If we run the same analysis as before, we will see that the sporadicity of interaction shows a power law distribution as well.

2 The Zipf Mystery - https://youtu.be/fCn8zs912OE

```python
# db_pairs is a dict containing all the timestamps
# of interaction of each interacting pair
for pair in db_pairs.keys():
    # List of inter-event activation times
    sporadicity = []
    sorted_timestamps = np.sort(db_pairs[pair])
    for i in range(len(db_pairs[pair]) - 1):
        sporadicity.append(sorted_timestamps[i+1] - sorted_timestamps[i])
    # db_sporadicity is a dict that will store the sporadicity
    # lists for each pair
    db_sporadicity[pair] = sporadicity
```

LISTING 4.1: Code snippet for calculating inter-event activation times.

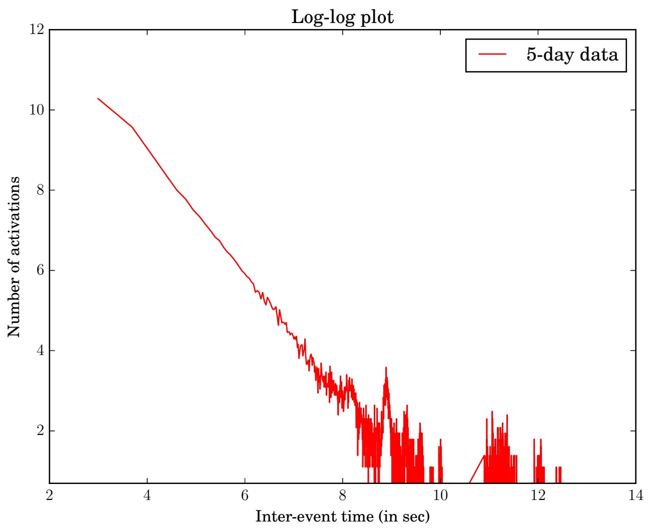


FIGURE 4.3: Log-log plot of the number of activations for inter-event times (sporadicity) for edge activation.

In fact the power law here has an oddly heavy tail. That is due to the fact that the data is spread over 5 working days. Thus there exist long duration spaces between edge activation times that correspond to the 4 nights. We can compress the temporal data by removing these night gaps and not lose any of the temporal or topological properties of the network. This is reasonable because nothing changes during the night as there are no interactions between the students. Any interaction that may have occurred can be assumed to be out of the bounds of the system and unrelated to our concern. Such a compression of data would not be reasonable if, for example, 24-hour interaction data was available for the nodes.

Thus, instead of considering the original data spread over 5 working days, we consider one really long day.

As can be seen in Fig. 4.4, the tail of the power law, though still fairly heavy, looks more reasonable after the data is compressed.

If we now plot a distribution of this data, we see that it is visually exponential with a very sharp decay rate.

An important inference to draw from this is that most pairs interact very often, with the minimum time discretization available (20 seconds) showing the most number of activations.

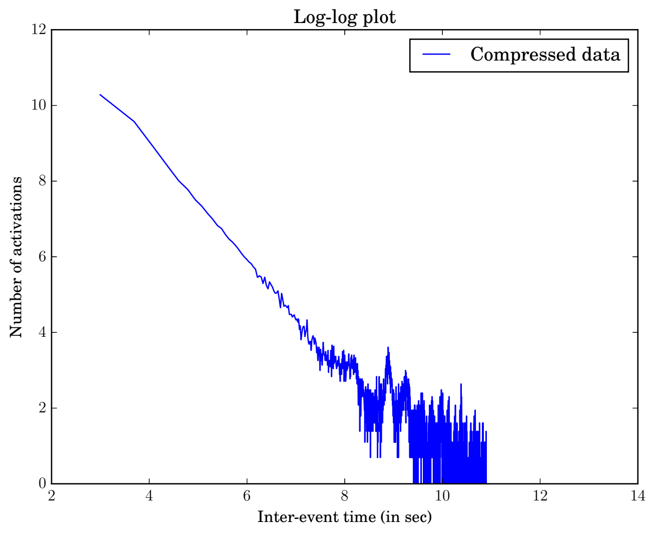


FIGURE 4.4: Log-log plot of the number of activations for inter-event times (sporadicity) for edge activation after temporal data is compressed.

This hints that there is a certain bursty character in the interaction pattern, i.e., when people interact they tend to do so for a long duration at a time.

Let us pick a pair at random. Say, 194 and 440 (which are their anonymized ID tags). The sporadicity of their interactions looks like this –

```
[20, 20, 20, 20, 20, 40, 60, 160, 20, 80, 40, 120, 120, 60, 380, 40, 20, 520, 20,
     20, 20, 20, 20, 20, 40, 60, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
    20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 60, 20,
    20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 120]
```

The occurrence of the five repeated `20`s in the beginning indicates that the pair interacted for $5 \times 20 = 100$ seconds at a stretch. Such patterns occur throughout the data.

A remark about the edge-based approach mentioned in the beginning of the section. If we looked at the inter-event times of all the nodes put together, we would have a global representation that gives us only the heavy-tailed sporadicity curve and nothing more. Since all we know is that a node is interacting and not who it is interacting with we cannot say whether the burstiness that occurs in this representation is due to the node interacting a lot with different people or if it is engaged in certain interaction pairs. The other approach, the one that we employ here, is an edge-based representation where we look at the activation times of individual edges. This gives us the sporadicity curve and highlights the bursty activation of edges.

A possible concern with the compression of temporal data above could be that it might lead to artificial amplification of the bursty character of interactions. However that is not the case as by considering only a single day of interactions one still gets the same results. This can be verified by running only Day 1's interactions through PATHPY and checking that the network still shows 2nd order properties.

If we revisit the questions posed at the beginning of this section, we now have simple one line answers if we consider a typical pair.

1. How many times does it interact? – Most pairs interact once or twice; a handful of pairs interact a lot.

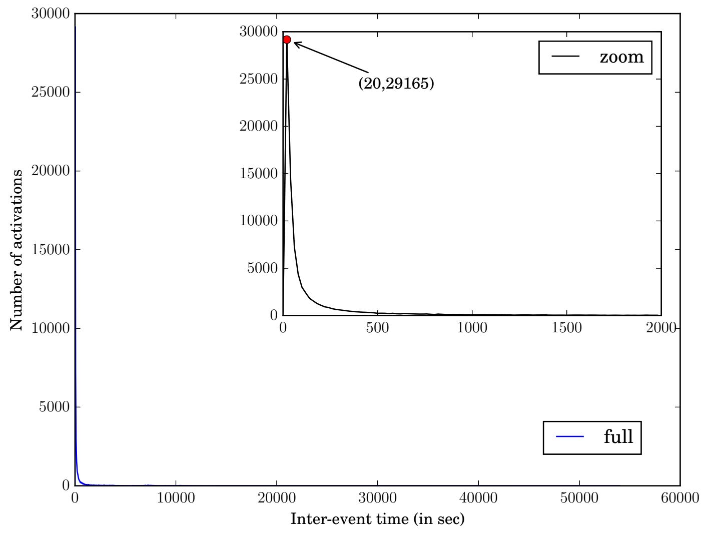


FIGURE 4.5: Distribution of edge activation times.

2. How often does it interact? – Most pairs interact quite often.
3. How long does it interact for? – Typically the interactions occur in bursts.

These are some basic interaction patterns that have emerged from the data. We can find some more if we shift our focus once again. Previously we looked at the edges instead of nodes, now we look at the timestamps.

First, we find out how many interactions occur at each timestamp. On plotting the distribution (Fig. 4.6) we see that the values range from 1 to 99 with an average of 25.56 interactions per timestep and a median 24 interactions per timestep.

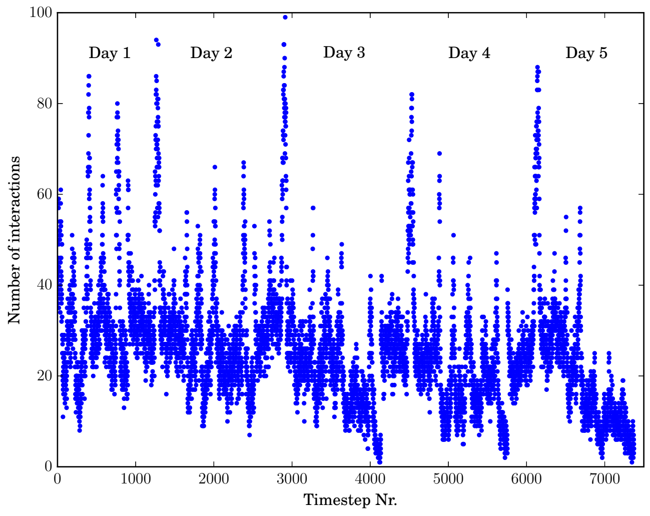


FIGURE 4.6: Plot of number of interactions for each timestamp.

The relatively close mean and median values hint at a possible Gaussian approximation. We plot the histogram and fit a Gaussian curve to it to see if such an approximation is reasonable.

Fig 4.7 shows that a Gaussian curve can be approximated on the distribution of number of interactions per timestep.

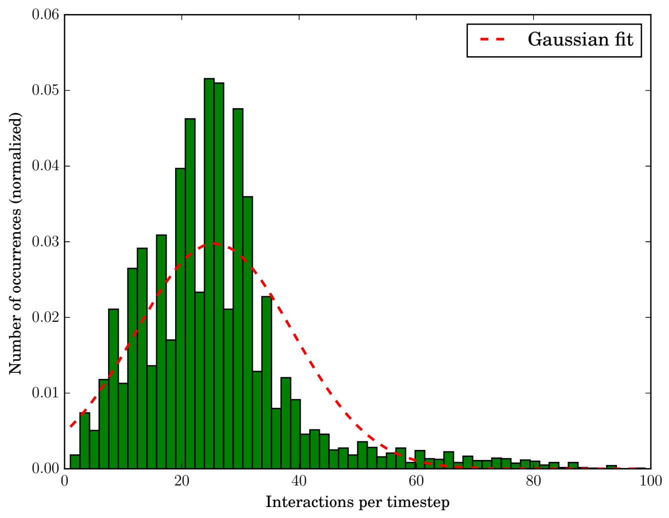


FIGURE 4.7: Histogram of number of interactions for each timestamp with a fitted Gaussian curve.

Further, in Fig. 4.6 it can be seen that there is a fairly strong separation between the different days that the data was collected. This can be assumed to be due to an increased chatter in the morning when the school reopens. While it may intuitively seem important to retain this feature in the algorithm, however, in collusion with our previous reasoning for compressing temporal data, it will become apparent that it is not necessary for recreating the temporal data. For our current target of generating synthetic temporal data it can be treated as a cosmetic feature that can be added to the data at a later point. But it will be important when finding resilience strategies, as discussed in the Chapter 6.

4.2 DEVELOPING THE ALGORITHM

Now that we have found some useful interaction patterns, we set out to develop the algorithm.

The gHypE package provides us with an adjacency matrix for each realization we draw from it. This solves the first question we posed in the previous section, i.e., we know how many interactions occur between any given pair of nodes. Or we know the number of times a given edge gets activated.

As for the sporadicity of interactions, we make a crude approximation that the distribution seen in Fig. 4.5 is exponential. Thus we fit an exponential distribution and using the rate parameter $\lambda(=\frac{1}{mean})$ we draw sporadicity values.

The other main ingredient we need is the number of interactions per timestep. To this effect we create a Python dict with each timestamp as key and a value drawn from the fitted Gaussian distribution (Refer: Fig. 4.7) as the number of interactions that will occur in that timestep. It should be noted that we use the same timestamps as the original data for our algorithm.

Our main objective is to fit all the interactions for each pair in the available timestamps with the added constraint that each timestamp can hold a previously determined fixed number of interactions. This can be easily understood with an analogy to the Urn problem[3].

We have a fixed number of urns (timestamps), each with a fixed capacity (number of interactions per timestamp). We have balls of different colours, each colour

[3] Wikipedia – Urn problem

representing an edge and the number of balls of the same colour represent the number of activations of that particular edge. Our problem now is to fit all the balls in all the urns in a way that maintains the non-Markovian temporal nature of interactions. (‽)

The principle we follow in creating this algorithm is that we will fit one interaction at a time for each pair sequentially. The intuitive reasoning for doing so is that that is how interactions actually occur in real life. Another reason for following this principle is based on trial and error. (Refer: Appendix A for other unsuccessful strategies that were tried.)

The pseudocode for our algorithm is provided in Listing 4.2. We will now clarify a few things about the code.

```
1  while AT LEAST ONE PAIR HAS AN INTERACTION TO BE ADDED:
2      for EACH PAIR:
3          if THE PAIR STILL HAS AN INTERACTION TO BE ADDED:
4              # Get the sporadicity of next interaction
5              next_sporadicity = PICK FROM EXPONENTIAL DISTRIBUTION
6              FOR EVERY TENTH ITERATION:
7                  next_sporadicity = PICK A RANDOM VALUE FROM ORIGINAL DATA
8                  CORRECT FOR NIGHTFALLS
9              # Add next timestamp
10             next_timestamp = previous_timestamp + next_sporadicity
11
12             # If a picked timestamp lies outside the range of available
13             # timestamps; Or if the picked timestamp cannot accomodate more
14             # interactions; Or if at least one of the nodes in the given pair
15             # is already interacting in the picked timestamp
16             CHECK:
17                 REPEAT LINES 5 - 10
18
19             # If after adding the timestamp just picked there are more
20             # interactions to be added and there are not enough timestamps
21             # left available
22             FALLBACK:
23                 DELETE THE TIMESTAMP LAST ADDED
24                 for NUMBER OF INTERACTIONS LEFT TO BE ADDED:
25                     # Add consecutive burst of interactions
26                     next_sporadicity = 20
27                     next_timestamp = previous_timestamp + next_sporadicity
28
29             UPDATE ALL THE DATABASES WITH THIS INTERACTION
```

LISTING 4.2: Pseudocode for SynTG algorithm.

While we pick most of our sporadicity values from the approximated exponential distribution, on every tenth run however, we pick a value from the original data instead. This is a correcting mechanism we added to the code. Given the nature of the exponential distribution it is practically impossible to draw values greatly than roughly 10 standard deviations. Which in our use case translates into – the largest possible sporadicity value that can be drawn from the exponential is 300 seconds. We need values far larger than that in order to accurately represent the interactions which is why we pick a value at random from the original pool of sporadicity values on every tenth iteration. Although it wasn't tested in this thesis, one can vary the parameters here to recreate the data with greater accuracy, however as with our discussion in the last paragraph of Section 4.1, it appears that doing so will not have an effect on the larger nature of interactions and will only act as a cosmetic add-on.

The check introduced in lines `12 - 17` of the pseudocode is crucial to the sanity of algorithm. A remark to be made about the final `if` condition in the check is that though it ensures that all nodes interact only once in a given timestamp, the following fallback condition circumvents this. This is not a huge problem though because even in the original data there are 7,034 cases where a node interacts with more than one other node in the same timestamp. Despite our inconsistency we are able to recreate this facet in our algorithm with great accuracy as our synthetic data shows generally 6,500 such interactions involving multiple nodes.

The fallback in lines `19 - 27` though initially introduced in order to boost the runtime of the algorithm turns out to be the most important part of the algorithm as it recreates the bursty nature of interactions as was seen in the previous section.

If we now compare the histogram of number of interactions per timestep in the original data v/s the final number of interactions in our synthetically generated data, and the sporadicity distributions in the original v/s the synthetic data, we see that they are in excellent agreement.

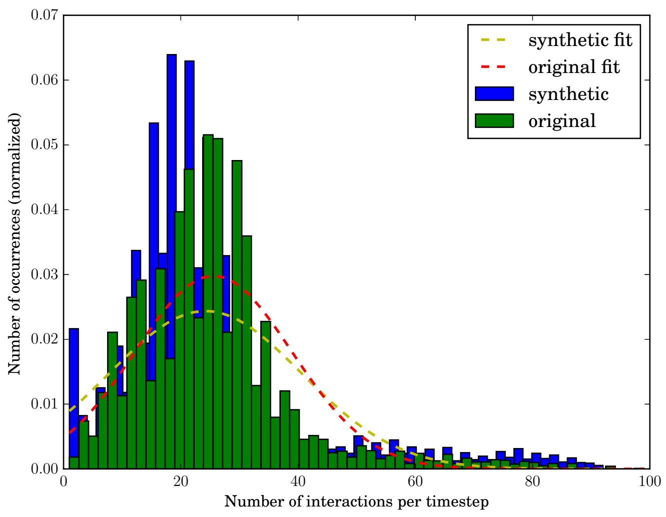


FIGURE 4.8: A comparison plot of number of interactions per timestamp for the original data v/s the synthetically generated data.

With that we come to the end of this chapter. We now have an algorithm that can be used to generate synthetic timestamp data given just the adjacency matrix of a static network in a way that the interaction patterns of the original data are preserved, i.e., the network topology is maintained.

More details about unsuccessful algorithm logic and some not-so-easily discernible bugs are provided in Appendix A.

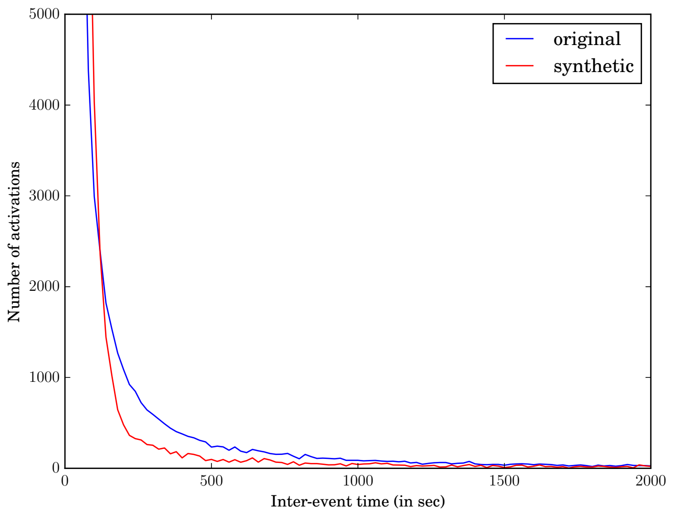


FIGURE 4.9: A comparison plot of sporadicity distribution in the original data v/s the synthetically generated data.

5 AGENT BASED MODEL

The final task of the thesis is to develop an agent-based model (ABM) that reproduces the repeated interactions of individuals according to the relations found in the data. This ABM would allow us to not only verify any control strategies found in the preceding analysis but also validate our SynTG algorithm.

As with the SynTG algorithm in the previous chapter, we start with the basic logic of the ABM. We can break the model down into the following steps –

1. Define an `Agent` class with all necessary properties.
2. Create a static network of agents.
3. Make the agents interact in a manner that preserves the interaction patterns of the original data.

It should be intuitive at this point that the three steps listed above are in an increasing order of difficulty. Nonetheless, we tackle them one at a time.

5.1 DEFINING AGENT PROPERTIES

First we create a class that will define our agents, for which we declare the properties listed in Table 5.1.

Agent property	Description	Range of values
sex	Sex of the agent	{M, F, Unknown}
classroom	Class that the agent is in	{MP, MP*1, MP*2, PSI*, PC, PC*, 2BIO1, 2BIO2, 2BIO3}
topic	Topic of agent's class	{MP, PSI, PC, BIO}
id_tag	Anonymized ID equivalent	$[0, \text{NUMBER\_OF\_AGENTS}] \in \mathbb{Z}$
degree	Agent's degree	$[1,\ 4562] \in \mathbb{Z}$
activity	Measure of how likely the agent is to initiate an interaction	$(0,\ 1) \in \mathbb{R}$
in_a_club	Toggle that tells if the agent is member of a club	$\{0,\ 1\}$
clubmates	IDs of agents that are in the same club; club size $\in \{5, 7, 10, 12, 15\}$	$[0, \text{NUMBER\_OF\_AGENTS}] \in \mathbb{Z}$
friends	IDs of agents that are in the same friend group; group size $\in [1,\ 25]$	$[0, \text{NUMBER\_OF\_AGENTS}] \in \mathbb{Z}$
person_history	ID of last agent interacted with	$[0, \text{NUMBER\_OF\_AGENTS}] \in \mathbb{Z}$
label_history	Interaction label of last agent interacted with	{class, topic, sex, friendship, club}

FIGURE 5.1: Properties of `Agent` class.

While the first four properties are the same as those available in the metadata of the original dataset, we define some new properties as well. The `degree` and `activity` properties will act as our supposed pseudo that we hope will recreate the sporadicity of interactions. The `clubmates` and `friends` properties will help in creating new labels for interactions beyond the trivial trio of *sex*, *class* and *topic*. Lastly, we keep a tab on the *ID* and *label* of the agent last interacted with because our previous analysis hints at the existence of a one-step memory in the network. We will elaborate on why these properties are needed in the following sections.

5.2 INITIALIZING AGENTS

Having defined the `Agent` class, we now initialize the agents and create a minimal network with nodes and node properties only and no edges.

To assign each agent to a `classroom` and `sex` we make use of the original distribution as seen in Table 3.1. The `topic` is automatically assigned based on the `classroom`. For `id_tag` we assign integer values ranging from 0 to the number of agents, which is 327 in our case. This allows for easy iteration and debugging.

In order to assign a `degree` to the agent we look at the degree distribution in the original data. When we plot the sorted degree distribution we see that it follows a polynomial distribution (Refer: Fig. 5.2). After a bit of trial and error we find that a fifth-degree polynomial fits best and that is what we use to draw `degree` values for our agents. We partition the degrees of all the agents into four quartiles, using which we then assign them `activity` values. So if an agent's `degree` lies in the third quartile, it will be assigned an `activity` value that also lies in the third quartile of all activities.

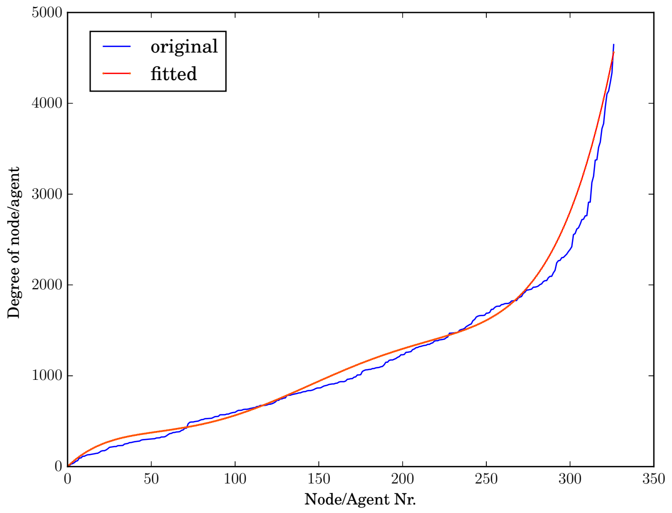


FIGURE 5.2: Degree distribution of nodes in the original dataset and a fitted fifth-degree polynomial curve. The distribution is sorted from lowest to highest.

Next we create groups of friends. The logic that we employ ensures that all agents except one have at least one friend. The solitary agent can be added to a friend group manually, however, we choose to keep it as an edge case. The friend groups range from a minimum of 2 members to a maximum of 25. The latter value is a consequence of the number of the total agents in the system. We also ensure that there are some agents that are members of multiple friend groups, as is seen in the original data, but the number of such agents is kept at around 10% of the total number of agents. This way we fill the `friends` property of the agents. The exact logic can be easily understood by viewing the code.

We also created 11 clubs with membership ranging from 5 to 15 members. Since it is unrealistic to assume that all students are involved in some club or the other we select 105 agents at random and assign them to different clubs. Unlike friend groups, there are no overlapping members in clubs.

We create a `populate_agents` function that initializes the agents and adds the aforementioned attributes to them. We also create a `create_interaction_network` function that creates an igraph `Graph` object with each node representing an agent with its attributes added as vertex properties.

5.3 MAKING AGENTS INTERACT

Finally we tackle the most involving task of the ABM – making the agents interact. We need to consider a few things that have become apparent in our preceding analysis. The interactions patterns found in Section 4.1 have to be brought forward to the ABM with the added constraint that the ABM should still show emergent behaviour, i.e., we shouldn't explicitly code all the patterns into the agents. The main patterns found were –

1. Sporadicity of interactions follows a heavy-tailed power law which implies that most interactions have very small inter-event times
2. There exist bursts of interactions
3. The degree distribution (of both nodes and edges) follow a power law
4. The number of interactions per timestep is loosely governed by a Gaussian distribution

Keeping these in mind we have to develop an interaction strategy from the perspective of an agent. That is our clue. We need to think from the point of view of a student in the school. This leads to some straightforward questions –

1. When a student initiates an interaction does she pick a student at random? Or based on a certain label (like *class*, *topic* or *sex*)?
2. Are the labels picked with equal probability or is there a governing likelihood ratio?
3. Are there more labels than just those available in the metadata? (Like *friendship* and *clubs*)
4. Does the person the student last interacted with have an effect on the person she chooses to interact with next? If so is it at the node level or at a label level?

We look at these above-mentioned questions as standards of complexity and so we add them to our model one at a time beginning with the very simple completely stochastic mode of interactions.

We follow the same basic principle as in SynTG – interactions occur sequentially. We add interactions to a timestamp till its quota of number of interactions is fulfilled and then move to the next occurring timestamp and so on. The reasoning is also the same as before – this is how interactions actually occur in real life. Another possible method would be to add all interactions for a single agent first and then move to the next but that is highly unrealistic and would very likely give misleading results.

The number of timestamps and number of agents remains the same as the original network. To define the number of interactions for each timestamp we make use of the same Gaussian distribution as described in Section 4.2 of the previous chapter.

The high-level pseudocode for interaction is presented in Listing 5.1. It doesn't look like much but it gives us an idea of how the interaction process should be structured.

```
def interact():
    for EACH TIMESTAMP:
        while THERE IS STILL AN INTERACTION TO BE ADDED:
            # Pick an agent who will initiate the interaction
            # and an agent who he will interact with
            interacting_pair = interaction_process()

            # Add interaction to the database
            add_interaction(CURRENT TIMESTAMP, interacting_pair)
```

Listing 5.1: Pseudocode for the interaction function.

Now we define the `interaction_process` function used in the pseudocode above. This function returns a tuple which represents the `interacting_pair`. We will now define this function various ways each time increasing the complexity of the process using the questions we listed earlier as a reference guide.

Note that the `activity_agents` variable is a Python `dict` we use to segment the agents into quartiles based on their individual `activity`.

Listing 5.2 illustrates the code behind the ABM based on stochastic interactions.

```
def interaction_process():
    # Pick a quartile based on the given probability bias
    quartile = np.random.choice(activity_agents.keys(), p = [0.1, 0.2, 0.3, 0.4])

    # Pick an agent who will initiate interaction from that quartile
    primary_agent = np.random.choice(activity_agents[quartile])
    # Pick an agent who will be interacted with
    secondary_agent = np.random.choice(ALL AGENTS)

    interaction_pair = (primary_agent, secondary_agent)
    return interaction_pair
```

Listing 5.2: Completely stochastic interaction.

Next we introduce label-based interaction in the ABM (Refer: Listing 5.3). In this case, instead of picking the `secondary_agent` completely at random the agent that initiates the interaction (`primary_agent`) first chooses a label and then picks an agent from that label at random.

```python
def interaction_process():
    # Pick a quartile based on the given probability bias
    quartile = np.random.choice(activity_agents.keys(), p = [0.1, 0.2, 0.3, 0.4])

    # Pick an agent who will initiate interaction from that quartile
    primary_agent = np.random.choice(activity_agents[quartile])

    # Pick a label from which the interacting agent will be picked
    label, label_members = pick_label(primary_agent)

    # Pick an agent who will be interacted with
    secondary_agent = np.random.choice(label_members)

    interaction_pair = (primary_agent, secondary_agent)
    return interaction_pair
```

Listing 5.3: Label-based interaction.

The algorithm for picking labels is shown in Listing 5.4. A remark needs to be made here about the probability biases used when picking labels. These values are actually the inter-layer odds ratio. We make use of a very crude definition for this –

$$\text{Odds}(class : sex) = \frac{\text{Number of interactions within same class/Total number of interactions}}{\text{Number of interactions within same sex/Total number of interactions}}$$

Looking at the original dataset we know that there are 124,191 interactions between two students of the same sex, 176,244 between those of the same class and 185,312 of the same topic. If we analyze the friendship network as well we find that there are 48,752 interactions between students who have reported each other as friends. However, since the friendship data is available for only a third of the total nodes, we choose to neglect it. We only use these odds for the case when the label has to be chosen from among *class*, *topic* and *sex*. *Friendship* data being incomplete and *club* label being made-up, we make use of uniform picking probabilities whenever either or both of these labels are in the sample space.

Finally we add the memory preservation control to the ABM. We introduce both one-step memory at the node-level and at the label-level. To do this we keep a tab on the `id_tag` and label of the person last interacted with for each agent. Listing 5.5 shows how we incorporate this into our code.

With this we conclude this chapter. Starting with a basic intuition we progressively added complexity to our model. The implications and results of the same will be discussed in the following chapters.

```python
def pick_label(choice, broad_label=None):
    # Pick a label

    # LOGIC
    # If you are not in a club
    if AGENT.in_a_club == 0:
        # And you don’t have friends
        if len(AGENT.friends) == 0:
            label = np.random.choice(['class', 'topic', 'sex'],
                                     p=[0.36283086, 0.38149901, 0.25567013])
        # But still have friends
        else:
            label = np.random.choice(['class', 'topic', 'sex', 'friendship'],
                                     p=[0.25,0.25,0.25,0.25])
    # If you don’t have friends
    elif len(AGENT.friends) == 0:
        # And are not in a club
        if AGENT.in_a_club == 0:
            label = np.random.choice(['class', 'topic', 'sex'],
                                     p=[0.36283086, 0.38149901, 0.25567013])
        # But are still in a club
        else:
            label = np.random.choice(['class', 'topic', 'sex', 'club'],
                                     p=[0.25,0.25,0.25,0.25])
    # If you have friends and are in a club
    else:
        label = np.random.choice(['class', 'topic', 'sex', 'club', 'friendship'],
                                 p=[0.2,0.2,0.2,0.2,0.2])

    # Create a sample space of agents in the broad label
    label_members = ALL AGENTS WITHIN THE SAME LABEL

    return label, label_members
```

Listing 5.4: Code for picking a label.

```python
def interaction_process():
    # Pick a quartile based on the given probability bias
    quartile = np.random.choice(activity_agents.keys(), p = [0.1, 0.2, 0.3, 0.4])

    # Pick an agent who will initiate interaction from that quartile
    primary_agent = np.random.choice(activity_agents[quartile])

    # Label memory
    if np.random.uniform() < 0.5:
        # Pick label from memory
        label, label_members = pick_label(primary_agent,
                                          memory=Agent.label_history)
    else:
        # Pick label from all labels
        label, label_members = pick_label(primary_agent, memory=False)

    # In-group memory
    if np.random.uniform() < 0.87:
        # Pick agent from memory
        secondary_agent = Agent.person_history
    else:
        # Pick agent from label
        secondary_agent = np.random.choice(label_members)

    interaction_pair = (primary_agent, secondary_agent)
    return interaction_pair
```

Listing 5.5: Interaction based on one-step memory preservation.

6 DISCUSSION

6.1 NON-MARKOVIAN DYNAMICS

The empirical interaction network displays non-Markovian dynamics in the way the interactions occur temporally which renders our previous methods of strategic intervention for controlling the infection spread inadequate. We investigate this temporal behaviour in detail in the hope that it will allow us to model better strategies.

We previously looked at the interaction patterns implicit in the data when creating the SynTG algorithm. These patterns allow us to investigate the non-Markovian dynamics precisely in order to be able to properly describe the temporal structure of the network. The two main patterns that we coded in our algorithm were –

- a heavy-tailed inter-event time distribution (sporadicity) and,
- a bursty character of interactions.

The sporadicity and bursty character both are artefacts of a heavy-tailed inter-event time distribution, and almost always occur simultaneously in empirical data. However, there is a subtle yet important distinction to draw between the two.

Consider again the inter-event time data for a pair of individuals as mentioned in Chapter 4 before –

```
[20, 20, 20, 20, 20, 40, 60, 160, 20, 80, 40, 120, 120, 60, 380, 40, 20, 520, 20,
    20, 20, 20, 20, 20, 40, 60, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20,
   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 60, 20,
   20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 20, 120]
```

Clearly a majority of interactions occur with a time gap of `20` seconds (which is the minimum time resolution available) followed by `40` seconds and `60` seconds and so on, as should be expected from a heavy-tailed distribution. The occurrence of 5 consecutive `20`s could either imply that it was one 100 seconds long interaction that got chopped into 5 discrete data points, or that these are actually 5 (or less) distinct interactions. The former, naturally, seems more likely. So we get what is called a bursty character in the interactions, i.e., individuals tend to interact for long duration spaces (bursts).

However, consider the inter-event time data for another pair of individuals drawn from the same empirical data set –

```
[40, 320, 20, 460, 1880, 20, 140, 20, 60, 20, 40, 20, 40, 20, 3840]
```

While the heavy-tailed power law character is apparent here as well, there is no bursty character at all! These two individuals interacted for short duration spaces and the interactions are separated by long duration spaces of time. A hypothetical situation where such an interaction "type" could occur is say between an office worker and the security guard. The worker meets the guard every time he enters or leaves the building and has a short interaction each time. Another hypothetical situation could be between two students from different classes who only interact when they meet at the water fountain or in the restroom.

The point being made here is that there are situations in real life where the inter-event time distribution can display a heavy-tailed power law but there is no bursty character in the interactions. However, if the bursty character exists then the inter-event distribution must be heavy-tailed. In other words –

> You can have a heavy-tailed inter-event distribution without burstiness but you cannot have burstiness without a heavy-tailed inter-event distribution.

Research so far has made little to no clear distinction in this regard and we believe it is an important property of interaction networks that should be kept in mind when dealing with such temporal networks.

What we know at this point is that the network displays non-Markovian dynamics. Using PATHPY we find that the network displays 2nd order dynamics, i.e., there exists a certain memory preservation in the network. We do not, however, know what exactly leads to this non-Markovian nature and where this memory comes from. The agent-based model we built helps us to pin that down.

To this effect we encode a number of different interaction behaviours into our agents and validate the temporal correlations using PATHPY to see which behaviours are necessary for creating memory in the network. These behaviours are –

- Activity-based interaction. Agents with a high value for the activity parameter interact more often others.
- Distribution-based agent degree. Agents with high activity get assigned a high value of degree. The degree distribution is sampled from the empirical data.
- Label-based interaction. When deciding who to interact with, agents pick based on different labels (class, topic, sex, friendship etc.). There exists a certain pre-defined empirically modeled odds ratio for picking labels.
- Label-based memory. Agents have a disposition to talk to someone from the same label as the last time.
- Memory based on in-groups. Agents have greater disposition to talk to someone who they have already interacted with thereby creating an in-group.

Apart from this there was an added constraint on the number of interactions per timestep which was also modeled empirically.

Using a process of elimination and by trying various combinations of the above-mentioned behaviours and constraints, we find out that the only requirement for creating a 2nd order network is the existence of an in-group memory in the agent behaviour. Everything else that we encoded acts as supplementary behaviour. This realization is in line with the findings of previous research on non-Markov dynamics of temporal networks. [32] We further find that roughly 87% (minimum) of interactions have to occur within the in-group. This result also corroborates with research in social psychology. [36]

Another interesting behaviour exhibited by our ABM is that the heavy-tailed inter-event time distribution and burstiness of interactions gets implicitly encoded into the system on the introduction of in-group memory. This further solidifies our claim that in-group memory acts as the source of non-Markovian dynamics.

Now that we know about the non-Markovian dynamics of our network, we shift focus to the temporal correlations. Temporal ordering plays a significant role in determining the speed of the contagion process. Research has shown that diffusion processes can be both sped up and slowed down by modifying the ordering of the temporal connections. [32] Other researchers has found that the existence of memory in a network inhibits the spreading process in SIR models. [37]

The idea is that by modifying the temporal ordering, an erstwhile 2nd order network with memory can be turned into a 1st order memory less network, and these 2nd order networks display lower values of spreading than the memoryless ones. We see the same effect in our epidemic modelling analysis when we find that iterating the temporal data randomly shows higher infectiousness than when iterated linearly. (Refer: Fig. 3.8) Random iteration destroys all temporal correlations in the network and turns it into a Markov memoryless network instead.

6.2 COMPARISON TO OTHER MODELLING FRAMEWORKS

At the time of writing this thesis the activity-driven model presented by Perra *et al.* appears to be the commonly used framework for modelling temporal networks. [38–40] We would like to offer a comparison between the activity-driven model (ADM) and the agent-based model (ABM) that we built in this thesis.

The activity-driven model offers an excellent analytically tractable solution for modelling temporal networks. The main externally defined parameter in the model is the activity which is defined as "the probability per unit time to create new contacts or interactions with other individuals". The activity values are "assigned according to a given probability distribution $F(x)$ that may be chosen arbitrarily or given by empirical data". At each timestep a node gets activated based on the probability given by its activity value. It can then choose to interact either randomly [38] or with a disposition to interact with someone from its in-group [39]. In the latter case, the probability to interact with someone from the in-group is given by $\frac{n}{n+c}$ where n is the number of existing ties and c is an offset constant chosen based on the degree class.

The ADM defines all the temporal characteristics based on the probability distribution $F(x)$. This compact formulation makes mathematical analysis very convenient. The resulting degree distribution depends on the form of $F(x)$, i.e., a power law distributed activity produces a power law degree distribution. The average number of interactions per timestep are also set based on $F(x)$. This is particularly useful when dealing with social networks (and many others) as they are known to exhibit distributions of this kind both for the degree and for the activity.

However, this leaves no room for any further adjustment. To use the ADM one has to always use a fixed degree distribution and average interactions per timestep value. There is only one parameter that can be adjusted (the probability distribution) and changing it modifies every other aspect of the network. Also, this minimal representation comes at the cost that one has to fix new values for the parameters for every new empirical data set. And though the authors say that ADM "allows a quantitative discussion of the biases induced by the time-aggregated representations in the analysis of dynamical processes", what the model is actually doing is that it is modifying the topology of the network when creating the temporal data. So despite its analytic usefulness it is pretty convoluted in its usage.

We mitigate these issues by providing a significant level of abstraction in our ABM which allows us to modify every aspect of the network however the use-case demands. As an abstract example, if the use-case demands a normally distributed degree, power law based activity, and log-normal distribution for interactions per timestep – our ABM can be used to model it. Since our ABM is built based on interaction patterns observed in empirical data it accurately represents real-life interactions. Plus, these interaction patterns can be easily switched on or off and new ones can be added without affecting the rest of the system. In addition, as shown in the previous section, the activity parameter is unnecessary for creating 2nd order networks; this cannot, however, be avoided in the activity-based model since the activity ties together all the constraints on the model.

Another issue that arises in the usage of ADM for generating networks with memory is that the fraction $\frac{n}{n+c} \rightarrow 1$ very quickly. The authors use $c = 1$ which sets a very heavy reinforcement condition. After about a dozen ties, the agent almost exclusively talks within its in-group, which is highly non-realistic. Our agent-based model avoids this problem entirely by using a fixed value for the probability of interacting with someone from within the in-group. Our non-rigorous trials show that any value above 0.87 is sufficient. One can of course modify this probability to even higher values if need be.

As mentioned earlier, breaking the temporal correlations of a 2nd order non-Markovian network (by shuffling the edges of the network) converts it into a 1st order Markovian memoryless network. This is verified for a number of empirical data sets using PATHPY. The temporal data we generate using the SynTG algorithm and with the ABM as well showcase this property. However, the data generated using the activity-based model does not display this property and even a shuffled temporal data appears as 2nd order in PATHPY. This leads us to believe that the activity-based model encodes something more than just in-group memory. We cannot comment what without a detailed analysis.

Finally, as noted in Chapter 4, only $5,818$ unique interaction pairs exist in the empirical data (HSI) out of a total possible $53,301$, which is about 10.9%. Even though it is not explicitly coded, the percentage of unique pairs in the temporal data generated using our ABM lies in the range of 10% $\pm$1%. However, the activity-driven model, thanks to its heavy reinforcement for in-group memory, always displays about 1.5% of unique pairs which again is highly non-realistic.

6.3 DRAWBACKS AND LIMITATIONS

The SIR analysis performed in this thesis is lacking in many respects. One, using just the infectiousness as a metric is not enough to solve the problem of defining a robustness parameter. The time it takes to infect X percentage of nodes, the effect of different seeding methods, cascade effects etc. are a few possible alternatives/additions in this direction. Secondly, for creating intervention and prevention strategies we must take into account individual node behaviour and not just the aggregated metrics. A more involved analysis is needed so we can find if the infected node talks a lot and infects a lot, or if it talks very economically at the right point of time. So we need to run targeted SIR models, i.e., we need to keep track of much more data at the node level such as finding out "super-spreaders" in the network. Lastly, if some "super-spreader" nodes are found, a regression analysis can be done to find exactly which metrics (such as centrality measures) qualify a node to become a super-spreader.

While working with the SynTG algorithm to generate temporal data for synthetic networks supplied by gHypE we some times were unable to create 2nd order networks. After some investigation we found that the issue lay in the way gHypE distributed the edges to the network. It only takes care of the total number of edges and not the number of unique edges. As noted in Chapter 4 and in the previous section as well, only 10.9% unique interaction pairs exist in the empirical data. Whereas since gHypE completely disregards this, the networks tend to show upwards of 20% unique pairs. The effect of which is seen when generating temporal data using SynTG since we do not explicitly code in-group memory into it. There are more people to talk to and so the in-group effect is lost. This can be easily mitigated by modifying the probability vector for the *rmultinom* function that generates multinomially distributed random number vectors. (Refer: Appendix A)

One can try to encode memory explicitly in the SynTG algorithm but that is a much harder problem because it involves finding out the best possible ordering of edges that leads to a 2nd order network. However, such a network space is large and one could potentially get a 2nd order network with a network topology different than the network in question. So eventually the problem boils down to ordering the edges based on the network topology which is basically what we do in SynTG by considering the interaction patterns. There may of course exist better approaches than the one that we have adopted.

One problem that may arise in our agent-based model is that when generating data for longer duration spaces of time (a few thousand time-steps), the algorithm shows a dramatic slow down in speed due to the fact that the agents in one's in-group might have exhausted their degree quota. A workaround for this can be coded wherein when picking the agents from one's in-group (which is currently completely random), those who have exhausted their degree quota are no longer in the sample size. The bias for in-group can also be reduced when such a situation arises.

6.4 SUGGESTIONS FOR FUTURE WORK

It will be worthwhile to explore whether investigating non-Markovian temporal networks in such detail has any real consequences when it comes to developing strategies for modifying the robustness of complex networks or specifically when creating intervention and prevention strategies for epidemics. The work done by Gemmetto, Barrat & Cattuto suggests that a certain level of time aggregation is reasonable. [41] Gauvin *et al.* suggest that individual-based control strategies are often impractical and ignore topological and temporal perspectives, and so they have developed a tensor-based mesoscopic analysis technique that can identify community structures and components that can be targeted to achieve resilience in the network. [42] Lee *et al.* exploit the network structure for creating effective infection stopping strategies. [43]

Based on our current analysis it appears that the non-Markovian nature occurs like a step function, i.e., it appears soon as there is in-group memory in the interaction process. It will be interesting to analyze, both using data and mathematics, if this in-group memory can be substituted with other interaction patterns. We have done something similar in the SynTG algorithm where we generated non-Markovian behaviour by not explicitly encoding memory, however, we can't precisely tell how this memory got encoded implicitly. It might also be possible that some networks that display in-group memory do not qualify as 2nd order networks as some other interaction pattern might counteract the effect of memory.

The ABM and SynTG algorithm developed in this thesis can be used to run all types of analysis on temporal networks. The models can be extended to all types of networks exhibiting non-Markovian dynamics such as communication networks, proximity networks, opportunistic P2P networks, neural networks etc. A possible work flow diagram is shown in Fig. 6.1.

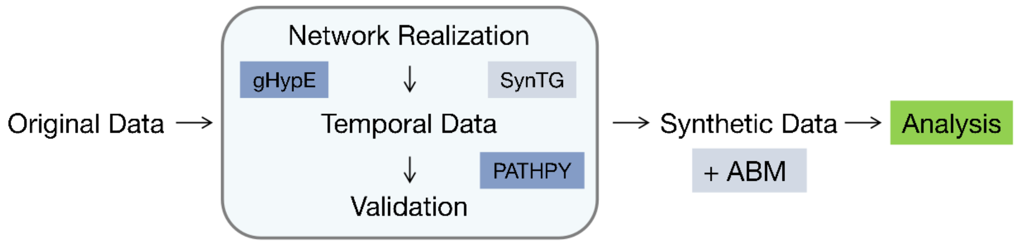


FIGURE 6.1: Suggested work flow for temporal analysis.

The ABM can be extended for different types of analysis. As an example, Pozzana, Sun & Perra extend the activity-based model by adding an attractiveness parameter that governs a node's propensity to attract an edge the same way the activity parameter defines its propensity to create an edge. The ABM can also be used for analyzing some social psychological research questions such as typical group sizes exhibited by humans [44] or the threshold probability for in-group behaviour.

7 CONCLUSION

I find that a great part of the information I have was acquired by looking up something and finding something else on the way.
— Franklin Pierce Adams

To put our work into context with the current trajectory of network science research, we build on the fact that time-aggregated perspective of temporal networks, though convenient and useful, is not perfectly applicable in all scenarios and particularly so in non-Markovian networks.

We confirm the results about heavy-tailed inter-event activation times and bursty character of interactions acting as underlying facets of human interaction networks and go on to further draw a subtle yet important distinction between the two – something that has so far been ignored by the research community.

On the 2nd order nature of non-Markovian temporal networks we show that this nature indeed stems from the existence of a node-level memory which points directly at the existence of in-groups. Our experimental results hint that roughly 87% of human interactions occur within our in-groups, though a rigorous formal analysis is required for validation of this claim.

Furthermore we are able to recreate this memory by using interaction patterns found in empirical data. In other words, we are able to show where exactly this memory comes from in the network. In doing so we have built an algorithm that preserves the network topology when adding temporal characteristics to it.

We also confirm that memory can have an effect on the speed of infection process in the network as has been shown in recent works on epidemic modelling and diffusion dynamics.

Lastly, the agent-based model developed in this thesis may appear to be at loggerheads with the activity-driven model presented by N. Perra et al, but we are able to show that the latter, though with its own merits of simplicity and analytical tractability, comes with a lack of generalization and a possible deviation from reality. The fact that our model was built based on the interaction patterns of empirical data not only fixes both of these problems but it also offers significant room for adjustment as required on a case-by-case basis.

We can't say just yet if our findings cause a significant deviation when it comes to estimation of epidemic control strategies. Perhaps a reasonable level of time aggregation and temporal deconstruction can be allowed. To find a concrete answer to this question we need to apply what has been done on time-aggregated networks to the temporal networks with the non-Markovian properties in mind, and then compare the two.

In the end, we would like to stray away from the tower of science into the highlands of society. It is important to remember, in conjunction to all the analysis on human-interaction datasets, that people are people — they are living, breathing human beings and not just data points with anonymous IDs. While we inch towards expanding our knowledge of our own species we must take a moment to revel in the mesmerizing complexity of humans as individuals and as a society. It is also important that all scientific progress must keep the lowest common denominator in its scope of view, for though a miscalculation may seem to lie within our error margins, it can have dire consequences for many real humans. Our past is dotted with examples of people being treated as objects, and our present is culprit to treating them as abstract data points. We have done a shoddy job as a sentient species to treat each of us as equals, let’s hope to do better.

BIBLIOGRAPHY

1. Mastrandrea, R., Fournet, J. & Barrat, A. Contact patterns in a high school: a comparison between data collected using wearable sensors, contact diaries and friendship surveys. *PloS one* **10**, e0136497 (2015).
2. Génois, M., Vestergaard, C. L., Fournet, J., Panisson, A., Bonmarin, I. & Barrat, A. Data on face-to-face contacts in an office building suggest a low-cost vaccination strategy based on community linkers. *Network Science* **3**, 326 (2015).
3. Cattuto, C., Van den Broeck, W., Barrat, A., Colizza, V., Pinton, J.-F. & Vespignani, A. Dynamics of person-to-person interactions from distributed RFID sensor networks. *PloS one* **5**, e11596 (2010).
4. Barrat, A., Cattuto, C, Tozzi, A., Vanhems, P. & Voirin, N. Measuring contact patterns with wearable sensors: methods, data characteristics and applications to data-driven simulations of infectious diseases. *Clinical Microbiology and Infection* **20**, 10 (2014).
5. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M. & Hwang, D.-U. Complex networks: Structure and dynamics. *Physics reports* **424**, 175 (2006).
6. Strogatz, S. H. Exploring complex networks. *nature* **410**, 268 (2001).
7. Holme, P. & Saramäki, J. Temporal networks. *Physics reports* **519**, 97 (2012).
8. Travers, J. & Milgram, S. The small world problem. *Phychology Today* **1**, 61 (1967).
9. Bakhshandeh, R., Samadi, M., Azimifar, Z. & Schaeffer, J. *Degrees of separation in social networks* in *Fourth Annual Symposium on Combinatorial Search* (2011).
10. Watts, D. J. & Strogatz, S. H. Collective dynamics of'small-world'networks. *nature* **393**, 440 (1998).
11. Shirky, C. *Here comes everybody: The power of organizing without organizations* (Penguin, 2008).
12. Barabási, A.-L. *Linked: The new science of networks* 2003.
13. Giesecke, J. *Modern infectious disease epidemiology* (CRC Press, 2017).
14. Keeling, M. J. & Eames, K. T. Networks and epidemic models. *Journal of the Royal Society Interface* **2**, 295 (2005).
15. Keeling, M. J. & Rohani, P. *Modeling infectious diseases in humans and animals* (Princeton University Press, 2008).
16. Pastor-Satorras, R. & Vespignani, A. Epidemic spreading in scale-free networks. *Physical review letters* **86**, 3200 (2001).
17. Pastor-Satorras, R. & Vespignani, A. Epidemic dynamics and endemic states in complex networks. *Physical Review E* **63**, 066117 (2001).
18. Moreno, Y., Pastor-Satorras, R. & Vespignani, A. Epidemic outbreaks in complex heterogeneous networks. *The European Physical Journal B-Condensed Matter and Complex Systems* **26**, 521 (2002).
19. Boguná, M. & Pastor-Satorras, R. Epidemic spreading in correlated complex networks. *Physical Review E* **66**, 047104 (2002).
20. Masuda, N. & Holme, P. Predicting and controlling infectious disease epidemics using temporal networks. *F1000prime reports* **5** (2013).

21. Kermack, W. O. & McKendrick, A. G. *A contribution to the mathematical theory of epidemics* in *Proceedings of the Royal Society of London A: mathematical, physical and engineering sciences* **115** (1927), 700.

22. Anderson, R. M., May, R. M. & Anderson, B. *Infectious diseases of humans: dynamics and control* (Wiley Online Library, 1992).

23. Hethcote, H. W. The mathematics of infectious diseases. *SIAM review* **42**, 599 (2000).

24. Van den Driessche, P. & Watmough, J. Reproduction numbers and sub-threshold endemic equilibria for compartmental models of disease transmission. *Mathematical biosciences* **180**, 29 (2002).

25. Shulgin, B., Stone, L. & Agur, Z. Pulse vaccination strategy in the SIR epidemic model. *Bulletin of mathematical biology* **60**, 1123 (1998).

26. Barabási, A.-L. *Network Science* (Cambridge University Press, 2016).

27. Casiraghi, G., Nanumyan, V., Scholtes, I. & Schweitzer, F. Generalized hypergeometric ensembles: Statistical hypothesis testing in complex networks. *arXiv preprint arXiv:1607.02441* (2016).

28. Casiraghi, G. Multiplex Network Regression: How do relations drive interactions? *arXiv preprint arXiv:1702.02048* (2017).

29. Scholtes, I. When is a network a network? multi-order graphical model selection in pathways and temporal networks. *arXiv preprint arXiv:1702.05499* (2017).

30. Scholtes, I., Wider, N. & Garas, A. Higher-order aggregate networks in the analysis of temporal networks: path structures and centralities. *The European Physical Journal B* **89**, 61 (2016).

31. Pfitzner, R., Scholtes, I., Garas, A., Tessone, C. J. & Schweitzer, F. Betweenness preference: Quantifying correlations in the topological dynamics of temporal networks. *Physical review letters* **110**, 198701 (2013).

32. Scholtes, I., Wider, N., Pfitzner, R., Garas, A., Tessone, C. J. & Schweitzer, F. Causality-driven slow-down and speed-up of diffusion in non-Markovian temporal networks. *arXiv preprint arXiv:1307.4030* (2013).

33. Rosvall, M., Esquivel, A. V., Lancichinetti, A., West, J. D. & Lambiotte, R. Memory in network flows and its effects on spreading dynamics and community detection. *Nature communications* **5** (2014).

34. Vestergaard, C. L. & Génois, M. Temporal gillespie algorithm: Fast simulation of contagion processes on time-varying networks. *PLoS computational biology* **11**, e1004579 (2015).

35. Diekmann, O., Heesterbeek, J. A. P. & Metz, J. A. On the definition and the computation of the basic reproduction ratio Ro in models for infectious diseases in heterogeneous populations. *Journal of mathematical biology* **28**, 365 (1990).

36. Efferson, C., Lalive, R. & Fehr, E. The coevolution of cultural groups and ingroup favoritism. *Science* **321**, 1844 (2008).

37. Sun, K., Baronchelli, A. & Perra, N. Contrasting effects of strong ties on SIR and SIS processes in temporal networks. *The European Physical Journal B* **88**, 326 (2015).

38. Perra, N., Gonçalves, B., Pastor-Satorras, R. & Vespignani, A. Activity driven modeling of time varying networks. *Scientific reports* **2** (2012).

39. Karsai, M., Perra, N. & Vespignani, A. Time varying networks and the weakness of strong ties. *Scientific reports* **4**, srep04001 (2014).

40. Pozzana, I., Sun, K. & Perra, N. Epidemic Spreading on Activity-Driven Networks with Attractiveness. *arXiv preprint arXiv:1703.02482* (2017).

41. Gemmetto, V., Barrat, A. & Cattuto, C. Mitigation of infectious disease at school: targeted class closure vs school closure. *BMC infectious diseases* **14**, 695 (2014).

42. Gauvin, L., Panisson, A., Barrat, A. & Cattuto, C. Revealing latent factors of temporal networks for mesoscale intervention in epidemic spread. *arXiv preprint arXiv:1501.02758* (2015).

43. Lee, S., Rocha, L. E., Liljeros, F. & Holme, P. Exploiting temporal network structures of human interaction to effectively immunize populations. *PloS one* **7**, e36439 (2012).

44. Burgess, J. W. Do humans show a "species-typical" group size?: Age, sex, and environmental differences in the size and composition of naturally-occurring casual groups. *Ethology and Sociobiology* **5**, 51 (1984).

A
PROGRAMMING INSIGHTS

Here we will offer some insights into how a cross-language compatibility was maintained alongwith some algorithm ideas that failed to work. We will also provide the code for creating synthetic realizations using gHypE and for using PATHPY for finding the optimal order of the network.

Even though the empirical data sets provide anonymized ID tags that can be used to label the nodes in the network, problems arise when loading networks using `igraph` in Python and R. A node with the same ID can have different node indices when the graph is loaded in R or Python. This causes problems if not carefully assessed. In particular problems were faced when node statistics were or other centrality measures were calculated in Python and realization graphs based on those indices were created in R. The same index represented some other node in R!

Another major problem occurs when dealing with Python dictionaries (and we deal with them a lot). Since it is not possible to extract keys from values we often had to maintain two parallel dictionaries one pointing from key to value and the other from value to key. Plus, Python built-in dicts do not maintain the order in which items are added so we chose to use the `OrderedDict` type from the `collections` library.

When working on the logic for the SynTG algorithm, we tried a bunch of different ideas that failed to work. Our first approach was to assign timestamps one interaction pair at a time. But there were way too many edge cases and the algorithm didn't always run error-free. Plus the runtime of the algorithm was unreasonably long as at each step it had to recalculate a list of "available timestamps" which was constrained by the sporadicity and interaction per timestep values. The approach we tried after this was the polar opposite – we looked at one timestamp at a time and assigned all possible pairs to it. But this made maintaining the sporadicity distribution very difficult. A few other ideas which were slight deviations based on these two were tried to no avail. Finally we landed on the idea to assign one timestamp at a time to each pair and it worked.

The code we used for creating realization networks using gHypE is shown in Listing A.1

For finding the optimal maximum order of the network we used the following code in Python using PATHPY –

```python
import pathpy as pp
hsi = pp.TemporalNetwork.readFile('dataset.csv', sep=',')
#hsi = hsi_t.ShuffleEdges()
print(hsi)

hsi_paths = pp.Paths.fromTemporalNetwork(hsi, delta=180)
print(hsi_paths)

hsi_model = pp.MultiOrderModel(hsi_paths, maxOrder=3)
print('Optimal maximum order = ', hsi_model.estimateOrder(hsi_paths))
```

```
b_new = b4
## b4 = [sex class topic friendship]
# If we want to remove the control for topic, we use -
# b_new <- c(b4[1:2],b4[4])

adj <- contact.adj
ens <- EnsembleFromAdjMatrix(adj,selfLoops = F, directed = F,isotropic = F)
omega <- matrix(apply(sapply(X = 1:length(b_new),FUN = function(i){as.vector(w4[[
    i]])^b4[i]}),MARGIN = 1,FUN = prod),nrow(adj))
omega[lower.tri(adj,T)] <- NA

omega[126,(126+1):327] <- 1.0
omega[1:(126-1),126] <- 1.0

ix <- !is.na(omega)
pp <- sum(ens@Xi[ix]*omega[ix])
p <- as.vector(ens@Xi[ix]*omega[ix])/pp

# Modify the probability vector to limit the number
# of unique edges in the network
p_new <- p
for (i in 1:53301){ if (runif(1,0,1) < 0.7){ p_new[i] <- 0}}

adj.random <- matrix(0,nrow(adj),ncol(adj))
adj.random[ix] <- rmultinom(n = 1, size = ens@M, prob = p_new)
colnames(adj.random) <- colnames(contact.adj)
rownames(adj.random) <- rownames(contact.adj)

# Save the graph
temp_graph <- graph_from_adjacency_matrix(adj.random, mode = "upper", weighted =
    NULL, diag = FALSE,
                                add.colnames = NA, add.rownames = NA)
V(temp_graph)$idtag <- V(contact.g)$name
write_graph(temp_graph, "./realization_network.gml", format = "gml")
```

Listing A.1: Creating synthetic networks using gHypE.

LIST OF FIGURES

LISTINGS

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.