EPIDEMIC PROCESSES ON COMPLEX NETWORKS

MODELLING, SIMULATION AND ALGORITHMS

EPIDEMIC PROCESSES ON COMPLEX NETWORKS

MODELLING, SIMULATION AND ALGORITHMS

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof. ir. K. C. A. M. Luyben,
voorzitter van het College voor Promoties,
in het openbaar te verdedigen op maandag 12 januari 2015 om 15.00 uur

door

Ruud VAN DE BOVENKAMP

elektrotechnisch ingenieur
geboren te Heemskerk.

Dit proefschrift is goedgekeurd door de promotor:

Prof. dr. ir. P. F. A. Van Mieghem

Copromotor: Dr. ir. F. A. Kuipers

Samenstelling promotiecommissie:

Rector Magnificus,	voorzitter
Prof. dr. ir. P. F. A. Van Mieghem,	Technische Universiteit Delft, promotor
Dr. ir. F. A. Kuipers,	Technische Universiteit Delft, copromotor
Prof. dr. eng. C. Scoglio	Kansas State University
Prof. dr. ir. M. R. van Steen	Vrije Universiteit Amsterdam
Prof. dr. ir. D. H. J. Epema	Technische Universiteit Delft
Prof. dr. ir. M. J. T. Reinders	Technische Universiteit Delft
Prof. dr. R. Pastor-Satorras	Universitat Politecnica de Catalunya
Prof. dr. ir. R. Kooij	Technische Universiteit Delft, reservelid

Title:	Epidemic Processes on Complex Networks: Modelling, Simulation and Algorithms
Front & Back:	An SIS outbreak on a rectangular lattice

ISBN 978-94-6186-411-6

An electronic version of this dissertation is available at
http://repository.tudelft.nl/.

To Natasha

CONTENTS

1

INTRODUCTION

If there is one thing that the human brain excels at, it is making connections. Maybe that is why networks are such popular scientific tools. After all, a network is nothing but a collection of points and connections. Or maybe we see networks everywhere because John Donne was right to observe that "No man is an island"[1], and networks truly are everywhere.

No matter the cause of the ubiquitousness of networks, the study of (complex) networks is very divers. Networks can represent vastly different objects including physical infrastructures such as rail, road and waterways [1, 2], flight routes and shipping lanes [3, 4], but also sewage systems and power grids [5], and, of course, the internet. Networks can be constructed from financial transactions [6], friendship or collaboration relations among individuals [7], sports players or online gamers that have played on the same team [8], and more abstract things such as functional brain networks where the nodes in the network are brain regions that share a link when they show correlated activity [9], or co-purchase networks where nodes are items in a shop that share a link when they were purchased together [10].

The emergence of the current multidisciplinary research field of complex networks is generally attributed to the rise of the computer. The exponential growth in transistor density of integrated circuits has brought cheap computation power and huge memory capacities, enabling the storage and manipulation of large datasets. The existence of large datasets in itself can be attributed to the continued computerisation of our world: machine readable data can be processed much easier than handwritten records. Where it used to take anthropologists many hours to map out the social structure of a community, perhaps using questionnaires or by taking interviews, the immense popularity of online social networking sites and the existence of digital address books of, for ex-

[1] No man is an island, entire of itself; every man is a piece of the continent, a part of the main. If a clod be washed away by the sea, Europe is the less, as well as if a promontory were, as well as if a manor of thy friend's or of thine own were: any man's death diminishes me, because I am involved in mankind, and therefore never send to know for whom the bell tolls; it tolls for thee. (part of Meditation #17 from Devotions upon Emergent Occasions (1623))

ample, email or phone contacts, has made it possible to extract the structure of huge communities in almost no time.

Regardless of what physical or conceptual entity a given network represents, the network representation is always used to gain knowledge about or insight in the behaviour of the entity. For example, it is crucial to know whether a power-grid is capable of delivering the requested energy to all the endpoints in the network. It is therefore necessary to compute, for a given load, the total current flow through each link and whether that flow is below the capacity of the link. And what happens if a link breaks? Is there another path through the network to continue to deliver energy to all endpoints, and do the links on that path have enough capacity to do so. Similar questions can be asked of all kinds of networks: is it still possible to offer a full train service after a points failure, how much congestion can be expected on road network when a main road is closed for engineering work, do species go extinct when the food-web is altered?

The answer to all these questions centre around the structural properties of a network. Nodes are mere abstractions, it is the links that show how a system is connected. A large part of complex network studies is occupied with defining, computing and understanding network metrics such as the maximum and average distance between two nodes, the number of shortest paths between all nodes that traverse a certain link, the number of neighbours that are neighbours amongst themselves and many others. Most metrics aim to indicate or rank which *nodes* are important/central to the network, based on the links between them. Especially in the study of online social networks this has lead to fascinating results (and reaffirmation of previous results) such as the friendship paradox, that states that your friends have, on average, more friends than you, and the six degrees of separation, that states that a random person is only six steps in friends away. In addition to topological metrics, which centre around how nodes are linked together, there is the spectral domain. In the spectral domain, metrics are derived from the eigenvalues of the adjacency matrix or the Laplacian matrix.

The structural properties of a network help to describe and analyse it and make it possible to classify nodes as central or influential, but the network in itself remains a static entity. A roadmap can tells us that there is a path between our home and our holiday destination on some sunny beach in southern France. Navigation equipment might use algorithms to compute what the shortest/quickest/cheapest/most scenic route towards that destination is. It is, however, the dynamic process of thousands of cars, lorries and coaches all trying to reach a myriad of different destinations that determines the performance of the network. Spectral metrics have proven especially useful in relating network properties to dynamic processes.

This interplay of the structure of the network and the dynamic process that runs on top of the network is the over arching topic of this thesis. Rather than characterising the topological properties of networks, we are interested in the influence that the topology has on a certain dynamic process that operates on top of the network. The spread of information or content through an online social network is a good example of a dynamic process on a network. But also the spread of a virus through a population or a computer virus over the internet is a good example of how basic and local interactions lead to a global level of infected nodes. An interesting question in the context of virus spread is how many nodes (people, computers, etc.) will be infected on average, and how long it

takes to reach this number, and how long they will stay infected. In the coming chapters we will answer these and other questions under various circumstances.

1.1. RESEARCH QUESTIONS

The main research focus of this thesis is the interplay of local interactions and global effects. Nature is full of examples of how simple, local rules within a community can lead to complex organised behaviour, such as the construction of termite mounds, the organised building, feeding and defence of ant colonies and the social organisation of bee-hives. We will mainly focus on a much simpler example of a dynamic process on a graph: virus or information spread. More specifically, we focus on the Susceptible-Infected-Susceptible virus model. Much work has been devoted over the past decades to understanding this simple virus model on graphs, and a few approximations to the otherwise intractable model have been proposed. Naturally, it is important to know in which types of graphs these approximations work well, and which one is best.

Approximations to the SIS model give the average fraction of infected nodes in the meta-stable state and a threshold value for the effective infection rate above which the meta-stable state actually exists. However, they do not offer any insight in how long the virus will stay in the meta-stable state or how quickly the meta-stable state is reached. Also, the meta-stable state itself is not very accurately defined. Especially in the context of real outbreaks of (computer) viruses, we need a better understanding of the temporal properties of the SIS model.

The SIS process models the infection and curing processes as Poisson processes: the inter-arrival time of infection and curing events are exponentially distributed. Whereas this makes the process a Markov process, and enables the use of the mathematical tools developed to analyse Markov processes, real-world infection and curing processes are not necessarily Poissonian. The effects of non-Markovian infection and curing should be well understood in order to make the SIS applicable to many of the real-world spreading phenomena.

From a engineering point of view, we can mimic a virus spreading process to use in an algorithmic setting. Purely local interactions between nodes can collectively form a dynamic process that computes global properties of the network in a completely decentralised fashion. Just as the topology dictates the success of an epidemic outbreak, it also dictates the performance of epidemic algorithms. A good understanding of the influence of topological features on, for example, the convergence time of an algorithm is crucial in designing successful epidemic algorithms. On the other hand, we can adapt the network to make it less vulnerable to virus threats, which conversely means that epidemic algorithms and content propagation is hindered. One approach is to remove links from the network to reduce or increase network metrics that are known to influence the process. This thesis contributes to a better understanding of the research questions mentioned in this section.

1.2. OUTLINE OF THIS THESIS

The remainder of this thesis is organised as follows. In chapter 2 we introduce the Susceptible-Infected-Susceptible (SIS) process on a graph together with two mean-field ap-

proximations. We benchmark these two approximations against a slightly modified SIS model, the ε-SIS model, where nodes also have a small self-infection rate in order to remove the absorbing state of the SIS model. Chapter 3 discusses the survival time of the SIS process on a graph. The worst-case survival time is the time to reach the absorbing state starting from the all-infected state. In chapter 4 the dynamic behaviour of two SIS processes competing for the same healthy nodes in a network is discussed. The key focus is on the domination time, a concept very much related to the survival time of a single outbreak. In chapter 5 we move away from the classical SIS process to understand the influence of a non-Poissonian infection and curing process on the epidemic threshold, the fraction of infected nodes in the meta-stable state, and the survival time of an epidemic. In a move even further away from the SIS epidemic, we develop an epidemic algorithm to count the number of nodes in a large dynamic and distributed network in chapter 6. Finally, we discuss how to extract networks from empirical datasets in chapter 7. We also discuss how to remove links from a network in such a way that the spreading of an epidemic is suppressed the most. In chapter 8 we give an overview of the contributions of this thesis to the general field of complex networks and discuss future research in the area of SIS epidemics and algorithmics. Appendix A gives an overview of all the tools we have developed to extract networks from datasets and to compute network properties and simulate epidemic algorithms.

2

THE SIS EPIDEMIC MODEL

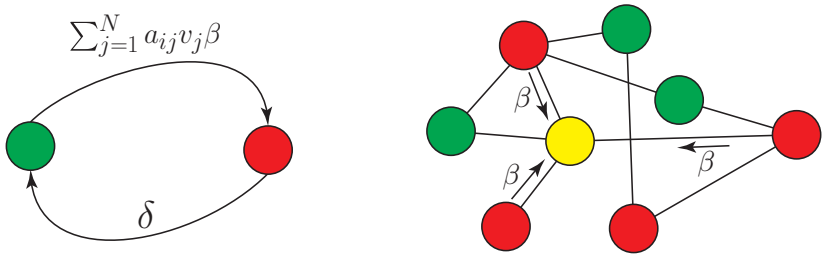
We introduce the SIS virus spread model and explain why it is generally infeasible to derive exact results. We use a modified SIS model, the ε–SIS model, as a benchmark for the comparison between the N-intertwined Mean-Field Approximation (NIMFA) and the Pastor-Satorras & Vespignani HMF approximation of the SIS model. NIMFA, the HMF approximation, and the ε–SIS spreading model are compared in different graph types. We focus on the epidemic threshold and the steady-state fraction of infected nodes in networks with different degree distributions. Overall, the NIMFA approximation is superior to the HMF approximation. In regular graphs, NIMFA and the HMF approximation are exactly the same. However, for some special graph types, such as the square lattice and path graph, the two mean-field approximations are both far away from the ε–SIS model, especially around the epidemic threshold.

2.1. THE SIS MODEL IN A NUTSHELL

The describing properties of the Susceptible-Infected-Susceptible [11] (SIS) model are so deceptively simple that a nutshell is indeed sufficient to contain all there is to know about the SIS model. A node can be in one of two states: healthy or infected. Nodes in the healthy state move to the infected state, while nodes in the infected state move back to the healthy state. The infection process describes nodes moving from the healthy state to the infected state, and the curing process describes nodes moving from the infected state to the healthy state.

In the classic SIS model, both the curing and infection processes are Poisson processes. Figure 2.1a shows the curing and infection processes from a nodal perspective. The curing process has a rate δ and is a nodal process, i.e. it is not influenced by the viral state of the neighbours of the infected node. The infection process, however, is a per link process with a rate β *per link* between a healthy and an infected node. The total rate of change from the healthy state to the infected state for a node i is given by β times the number of infected neighbours.

Figure 2.1b shows the SIS process from a network perspective. The red nodes are infected nodes, whereas the green nodes are healthy nodes. If the yellow node i is infected,



(a) The SIS process as seen from a node perspective. The red node denotes the infected state, whereas the green node denotes the healthy state.

(b) The SIS process as seen from a network perspective. Red nodes are infected nodes, whereas green nodes are healthy nodes. The yellow node is also healthy, but coloured differently for easy reference.

Figure 2.1

it will cure (move to the healthy state) with a rate δ irrespective of the state of the rest of the network. When it is healthy, however, the three infected neighbours will spread the infection with a combined rate of 3β. The combined or total infection rate that node i experiences is not constant over the healthy period of the node as infected neighbours can cure. Even though all Poisson processes are independent, the viral state of the nodes are not independent since clearly the rate of change from the healthy to the infected state of the yellow node i in Figure 2.1b depends on the number of infected neighbours.

Since the infection and curing processes in the SIS model are Poisson processes, the entire infectious state of the network is fully described by a continuous-time Markov chain. With the Markov description of the network state, all properties of the SIS process are known, that is, theoretically. The one complicating factor in the Markov analysis of the SIS model is the exploding state space. Figure 2.2 shows the Markov graph of the SIS process on K_5, a complete graph of 5 nodes. The Markov graph in Figure 2.2 illustrates two fundamental properties of the SIS process. First, every ordered combination of healthy and infected nodes is a state in the Markov chain, and second, whereas almost all transitions in the Markov graph are bidirectional, the ones to the all-healthy are unidirectional. The former tells us that the state space scales with 2^N in the number of nodes N, the latter that the all-healthy state is an absorbing state.

The exponentially growing state space of the Markov chain makes an exact analysis of the SIS process using Markov chains infeasible for graphs larger than 10 to 20 nodes in most cases. For a few graphs, such as the complete graph and the star graph, it is possible to reduce the number of states in the Markov chain, but this is not generally the case. The absorbing state in the Markov chain tells us that the infection will eventually die out, leaving the network in the all-healthy state. The natural questions to ask are then, how long will it take until the infection dies out, and how many nodes will be infected, on average, during an outbreak of the infection. These two questions are the motivation for substantial parts of this thesis and will pop-up frequently.

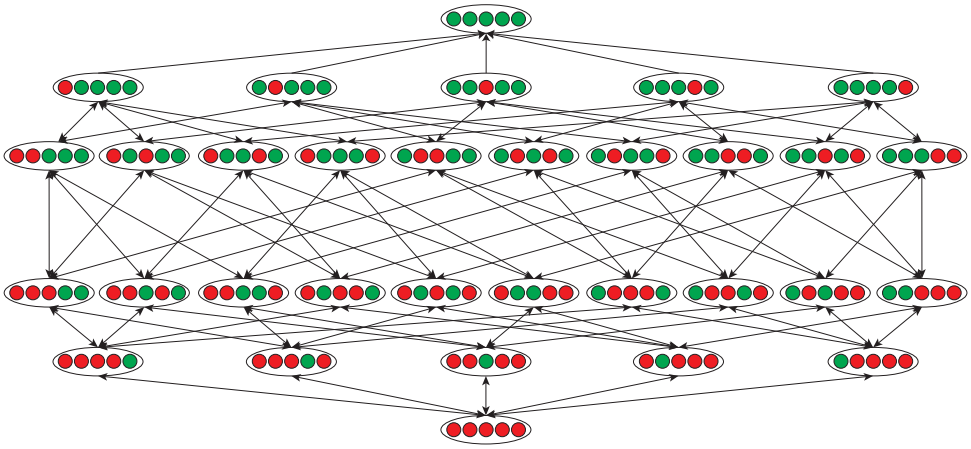


Figure 2.2: SIS state space for a complete graph of 5 nodes (K_5).

With the focus on the duration of an infection outbreak and the number of infected nodes during such an outbreak, we move out of the nutshell and into the complex world of dynamic processes on networks. Because of the huge state space in the general case, Markov methods are of limited use and approximations of the SIS process are needed. In this chapter we evaluate two different mean-field approximations of the SIS process: the N-intertwined Mean-Field Approximation (NIMFA) [12, 13] and the Pastor-Satorras & Vespignani heterogeneous mean field (HMF) approximation [14].

We consider the spread of an infection or virus in an undirected graph $G(N, L)$, characterised by a symmetric adjacency matrix A. The effective infection rate is defined as $\tau = \frac{\beta}{\delta}$, that is, the infection rate divided by the curing rate. The viral state of a node i at time t is specified by a Bernoulli random variable $X_i(t) \in \{0, 1\}$, where $X_i(t) = 0$ refers to a healthy node and $X_i(t) = 1$ to an infected node. Every node i at time t is either infected, with probability $v_i(t) = \text{Prob}[X_i(t) = 1]$ or healthy (but susceptible) with probability $1 - v_i(t)$.

A fundamental question very much related to the duration of an outbreak, is whether a virus will spread through the entire network, or will die out. It turns out that for viruses with an effective infection rate that is high enough, the existence of the absorbing state is of little practical consequence, and the epidemic will stay in the network for a very long time. It has long been observed (see [14–21]) that an epidemic threshold τ_c exists that separates two different regimes in the dynamic infection process on a network: for an effective infection rate τ above the threshold, the infection spreads and becomes persistent in time; for $\tau < \tau_c$, the infection dies out exponentially fast.

A first order mean-field epidemic threshold $\tau_c^{(1)} = \frac{1}{\lambda_1(A)}$, where $\lambda_1(A)$ is the largest eigenvalue of the adjacency matrix A, was first proposed by Wang *et al.* [21], and rigorously proved by Van Mieghem *et al.* in [12, 13] and later appeared in the physics community [22]. Van Mieghem *et al.* [12] also showed that this mean-field threshold lower bounds the "in reality observed" epidemic threshold, $\tau_c^{(1)} = \frac{1}{\lambda_1(A)} \leq \tau_c$. A more accurate lower bound (the second order mean-field threshold) $\tau_c \geq \tau_c^{(2)} \geq \tau_c^{(1)}$ has been derived in [23].

The HMF epidemic threshold [14, 17] is given by $\tau_c^{HMF} = E[D]/E[D^2]$, where D is the degree of a randomly chosen node in G.

In this chapter, we present a detailed comparison of the two mean-field approximations. Usually, the quality of an approximation is assessed by two criteria: 1) which approximation is closer to the exact SIS model in terms of average fraction of infected nodes, and 2) which approximation's epidemic threshold is nearer to the epidemic threshold of the exact SIS model? A direct comparison to the SIS model is, however, not possible, because the steady-state of the exact SIS model in a finite network is, as explained earlier, the absorbing all-healthy state. The presence of an absorbing state is a major complication in the analysis of the SIS model.

The steady-state of both mean-field approximations corresponds, in fact, to what is called the *meta-stable state* in the SIS model, which is not accurately defined for finite networks [12]. Therefore, we define the meta-stable state of the SIS model via the steady-state of the ε–SIS model for a prescribed value of ε. The ε–SIS process generalises the SIS model by adding a nodal component to the infection. We assume that each node i can be infected spontaneously. The spontaneous infection process is a Poisson process with rate ε. Hence, besides receiving the infection over links from infected neighbours with rate β, the node i can also itself produce a virus with rate ε. As a consequence of the self-infection, the unidirectional state transitions in Figure 2.2 to the all-healthy state become bidirectional for $\varepsilon > 0$, removing the absorbing state, and Markov theory guarantees a unique steady-state. Ideally, ε is small enough not to influence the other state transitions too much. When $\varepsilon = 0$, the ε–SIS model reduces to the "classical" SIS model. Hence, for small values of $\varepsilon > 0$, the ε–SIS spreading model can be used to approximate the exact SIS model. Here, the ε–SIS spreading model with a small value for ε is used as a benchmark to compare the steady-state of NIMFA and the HMF approximation on different network types.

2.2. SIMULATION METHOD & MEAN-FIELD APPROXIMATIONS

We will first introduce the ε–SIS model simulation techniques we have used and give a detailed description of the two mean-field approximations, before turning to the comparison of the two approximations in various graph types.

2.2.1. SIMULATING THE ε–SIS SPREADING MODEL

The ε–SIS spreading model was proposed recently by Hill et al. [24] in their analysis of emotions as a form of infection in a social contact network and earlier in [25] where ε is defined as the driving field conjugate to the density of infected nodes. Here, we will explain the simulation process, but defer to [26] for an analysis of the ε–SIS model.

We will briefly describe the ideas behind our simulator here. See Appendix B for a more detailed description of the simulator. In our simulations we take a nodal central, event driven approach. An event can either be the curing of a node or the spreading of the infection from one node to another. Events are stored in a timeline as tickets. A ticket contains, besides the time and the event type (spreading or curing), the owner of the ticket. The ticket owner is usually a node, but can also be the system to allow for scheduling of administrative tasks. Tickets are continuously taken from the timeline and

passed on to the owner.

If the ticket owner is a node, the ticket either indicates a curing or spreading event. In case of a curing event, the node simply changes its state from infected to healthy; in case of a spreading event, it will spread the infection to the neighbour mentioned in the ticket. If the neighbour was not already infected, it will now become infected and create one or more tickets.

A newly infected node will always create a ticket for its own curing event. According to continuous-time Markov theory (see [27]), the time between infection and curing is exponentially distributed with rate δ and is stored by the node for future reference. An infected node also generates spreading times at which it will spread the infection to its neighbours. The spreading times are again exponentially distributed but now with rate β. If the spreading time does not exceed the node's curing time, a ticket is created for the spreading event. All newly created tickets are stored in the timeline. Finally, the owner of the original ticket generates a new spreading time, which, if not exceeding its own curing time, creates a new spreading ticket for the same neighbour.

If the ticket is not owned by a node, it is a system ticket. System tickets are used to cause the spontaneous infections in nodes. Every node becomes infected spontaneously at a rate ε, but to minimise the number of tickets in the timeline, the system creates one spontaneous infection ticket at the time. The time between spontaneous infection tickets is exponentially distributed with rate $N\varepsilon$. When the system receives a spontaneous infection ticket, it selects a random node and tries to infect it. If the node is already infected, nothing will change, whereas a healthy node will become infected and create the tickets described above.

During the simulation, for each possible number of infected nodes (0 to N) it is recorded how long the network was in a state with that many nodes infected. The average number of infected nodes during the simulation can be determined by multiplying the number of infected nodes by the fraction of time spent in that state, and sum over all the states.

2.2.2. THE PASTOR-SATORRAS & VESPIGNANI HMF APPROXIMATION

Pastor-Satorras & Vespignani [14] studied the Susceptible-Infected-Susceptible (SIS) epidemic on networks and proposed the heterogeneous mean-field (HMF) approximation, in which the degree distribution plays an important role. In the notation of the original HMF work, the fraction (or density) of infected nodes in a network is denoted by ρ, and the relative density of infected nodes with degree k, i.e., the probability that a node with k links is infected, by $\rho_k(t)$. The dynamical mean-field reaction rate equation can be written as

$$\partial_t \rho_k(t) = -\delta \rho_k(t) + \beta k[1-\rho_k(t)]\Theta\big(\rho(t)\big),$$

where $\Theta\big(\rho(t)\big)$ is the probability that any given link points to an infected node. In steady-state, $y_\infty = \lim_{t\to\infty}\rho(t)$ is only a function of the infection rate β and curing rate δ, and as consequence, so is $\Theta\big(\rho(t)\big)$. By imposing stationarity $[\partial_t \rho_k(t) = 0]$, when $t \to \infty$, the relative density reduces to

$$\rho_k(\tau) = \frac{\tau k \Theta(\tau)}{1 + k\tau\Theta(\tau)} \tag{2.1}$$

where $\tau = \frac{\beta}{\delta}$ is the effective infection rate, and

$$\Theta(\tau) = \frac{1}{E[D]} \sum_{k=1}^{N-1} k \text{Prob}[D = k] \rho_k(\tau) \tag{2.2}$$

where D is the degree of a randomly selected node in the graph. Clearly, if $\tau = 0$, then $\Theta(0) = 0$. Substituting (2.1) into (2.2) leads to a self-consistent relation, from which $\Theta(\tau)$ can be determined as

$$\Theta(\tau) = \frac{\tau\Theta(\tau)}{E[D]} \sum_{k=1}^{N-1} \frac{k^2 \text{Prob}[D = k]}{1 + k\tau\Theta(\tau)} \tag{2.3}$$

(2.3) has a trivial solution $\Theta(\tau) = 0$. For a non-trivial solution $\Theta(\tau) > 0$ to exist, (2.3) must satisfy the following condition:

$$\frac{E[D]}{\tau} = \sum_{k=1}^{N-1} \frac{k^2 \text{Prob}[D = k]}{1 + k\tau\Theta(\tau)} \tag{2.4}$$

Next, we introduce the following expansion,

$$\frac{1}{1 + k\tau\Theta(\tau)} = \sum_{j=0}^{\infty} (-1)^j (k\tau\Theta(\tau))^j$$

which is valid when $k\tau\Theta(\tau) < 1$ for all k,

$$\begin{aligned} \frac{E[D]}{\tau} &= \sum_{j=0}^{\infty} (-1)^j \left\{ \sum_{k=1}^{N-1} \text{Prob}[D = k] k^{j+2} \right\} \tau^j \Theta^j(\tau) \\ &= \sum_{j=0}^{\infty} (-1)^j E\left[D^{j+2}\right] \tau^j \Theta^j(\tau) \end{aligned}$$

where the latter series converges for $\Theta(\tau) < 1/(D_{max}\tau)$. Since $\tau = 0$ leads to $\Theta(0) = 0$, the non-trivial solution $\Theta(\tau) > 0$ exists when $\tau > \tau_c^{\text{HMF}} \geq 0$ by the definition of the epidemic threshold. When $\Theta(\tau)$ is sufficiently small, we can write the expansion up to first order as

$$\frac{E[D]}{\tau} = E[D^2] - \tau\Theta(\tau)E[D^3] + O(\Theta(\tau)^2) \tag{2.5}$$

in which $\tau\Theta(\tau)E[D^3] > 0$. Hence, when $\tau > \tau_c^{\text{HMF}}$, but $\Theta(\tau)$ is small enough to ignore the second order terms $O(\Theta(\tau)^2)$, we have from (2.5),

$$\frac{E[D]}{\tau} < E[D^2]$$

implying that for all $\tau > \tau_c^{\text{HMF}}$, it holds that $\tau > \frac{E[D]}{E[D^2]}$. Thus, the epidemic threshold of the HMF approximation is

$$\tau_c^{\text{HMF}} = \frac{E[D]}{E[D^2]}$$

The same result was also deduced differently in [28]. For a regular graph with degree r, it holds that $E[D^2] = E[D]^2 = r^2$, and the epidemic threshold is given by $\tau_c^{\text{HMF}} = \frac{1}{r} = \frac{1}{\lambda_1}$ [14].

Finally, we can evaluate the steady-state fraction $y_\infty(\tau)$ of infected nodes using the relation

$$y_\infty(\tau) = \sum_{k=1}^{N-1} \text{Prob}[D=k]\rho_k(\tau) \tag{2.6}$$

2.2.3. THE NIMFA APPROXIMATION

The HMF approximation considers the relative density $\rho_k(t)$ of infected nodes with given degree k during the epidemic process. However, the state of each node is not taken into account. The NIMFA approximation [12, 29] is derived by separately observing each node. Every node i at time t in the network is in one of two states: infected, with probability $\text{Prob}[X_i(t)=1]$ or healthy, with probability $\text{Prob}[X_i(t)=0]$. Since a node can only be in one of two states, $\text{Prob}[X_i(t)=0]+\text{Prob}[X_i(t)=1]=1$. As explained in Section 2.1, the curing and infection processes are Poisson processes, and the whole epidemic process is a Markov process. A straightforward application of Markov theory gives the infinitesimal generator $Q_i(t)$ of the two-state nodal continuous-time Markov chain (see also Figure 2.1a) as

$$Q_i(t) = \begin{bmatrix} -q_{1;i} & q_{1;i} \\ q_{2;i} & -q_{2;i} \end{bmatrix}$$

with $q_{2;i}=\delta$. Markov theory requires that the infinitesimal generator is a matrix whose elements are not random variables. However, this is not the case in our simple model: $q_{1;i}(t) = \beta\sum_{k=1}^{N} a_{ij}1_{\{X_k(t)=1\}}$. Using a mean-field approximation [12] so that $E[q_{1;i}] = \beta\sum_{j=1}^{N} a_{ij}\text{Prob}[X_j(t)=1]$, the effective infinitesimal generator becomes

$$Q_i(t) = \begin{bmatrix} -E[q_{1;i}] & E[q_{1;i}] \\ \delta & -\delta \end{bmatrix}$$

Then, in accordance with Markov theory in [27, eq. (10. 11), pp. 182], denoting $v_i(t)$ =$\text{Prob}[X_i(t)=1]$ and $\text{Prob}[X_i(t)=0] = 1-v_i(t)$, the set of nodes obey the differential equations

$$\begin{cases} \frac{dv_1(t)}{dt} = \beta\sum_{j=1}^{N} a_{1j}v_j(t) - v_1(t)(\beta\sum_{j=1}^{N} a_{1j}v_j(t)+\delta) \\ \frac{dv_2(t)}{dt} = \beta\sum_{j=1}^{N} a_{2j}v_j(t) - v_2(t)(\beta\sum_{j=1}^{N} a_{2j}v_j(t)+\delta) \\ \vdots \\ \frac{dv_N(t)}{dt} = \beta\sum_{j=1}^{N} a_{Nj}v_j(t) - v_N(t)(\beta\sum_{j=1}^{N} a_{Nj}v_j(t)+\delta) \end{cases}$$

written in matrix form,

$$\frac{dV(t)}{dt} = \beta AV(t) - diag(v_i(t))(\beta AV(t)+\delta u) \tag{2.7}$$

where the vector $V(t) = [v_1(t)\ v_2(t) \cdots\ v_N(t)]^T$. The average number of infected nodes in G is equal to $y(t) = u^T V(t)$, where u is the all-one vector.

An alternative approach to find the NIMFA epidemic threshold starts with the expected value of the nodal infection probability. Since X_i is a Bernoulli random variable, which has the nice property that $E[X_i] = \Pr[X_i = 1]$, the exact SIS governing equation [23] for node i equals

$$\begin{aligned}\frac{dE[X_i(t)]}{dt} &= E\left[-\delta X_i(t) + \beta(1 - X_i(t))\sum_{k=1}^{N} a_{ki}X_k(t)\right] \\ &= E\left[-\delta X_i(t) + \beta\sum_{k=1}^{N} a_{ki}X_k(t) - \beta\sum_{k=1}^{N} a_{ki}X_i(t)X_k(t)\right] \qquad (2.8)\end{aligned}$$

where (2.8) also holds for asymmetric adjacency matrices. Directly from (2.8), we deduce that

$$\frac{dE[X_i(t)]}{dt} \leq -\delta E[X_i(t)] + \beta\sum_{k=1}^{N} a_{ki}E[X_k(t)]$$

When written for all nodes i, with $w_i = E[X_i(t)]$ and the vector $W = (w_1, w_2, \cdots, w_N)$, we obtain the matrix inequality

$$\frac{dW(t)}{dt} \leq \left(\beta A^T - \delta I\right)W(t) \qquad (2.9)$$

from which follows that

$$W(t) \leq e^{(\beta A^T - \delta I)t}W(0) = e^{(\tau A^T - I)t^*}W(0)$$

where the effective infection rate $\tau = \frac{\beta}{\delta}$ and the normalised time $t^* = \delta t$ is measured in units of the curing rate δ. The upper bound is dominated by the fastest growth in t^*, which is due to the largest eigenvalue of $\tau A^T - I$. The exponential factor is dominated by $\tau\lambda_1(A) - 1$, where $\lambda_1(A)$ is the real, largest eigenvalue of the non-negative matrix A (by Perron-Frobenius Theorem, see [30]). When $\tau\lambda_1(A) - 1 \leq 0$ or $\tau \leq \frac{1}{\lambda_1(A)} = \tau_c^{(1)}$, where $\tau_c^{(1)}$ is the first-order mean-field epidemic threshold [31], $w_i = E[X_i(t)]$ decreases exponentially in t^*. Hence, the epidemic will die out fast. By definition of the epidemic threshold τ_c as the border between exponential die-out and a non-zero fraction of infected nodes in the meta–stable state, we conclude that the exact epidemic threshold $\tau_c \geq \tau_c^{(1)}$ in any finite sized network. A major property, proved in [12] as well as in [23], of the NIMFA approximation is that $v_i(t) \geq v_i(t)|_{exact}$. Hence, the NIMFA approximation upper bounds the SIS epidemics and, consequently, $\tau_c^{(1)} < \tau_c$. The *lower* bound $\tau_c^{(1)} = \frac{1}{\lambda_1(A)}$ is of great practical use: if the effective infection rate τ can be controlled such that $\tau \leq \tau_c^{(1)}$, then the network is safeguarded from long-term, massive infection. The approximation $\tau_c^{(1)} = \frac{1}{\lambda_1(A)}$ can be refined to get a second order epidemic threshold $\tau_c^{(2)} \geq \tau_c^{(1)}$ as shown in [23]. Successive higher order approximations are possible but will increase the complexity with each added term.

2.3. THE STEADY-STATE FRACTION OF INFECTED NODES

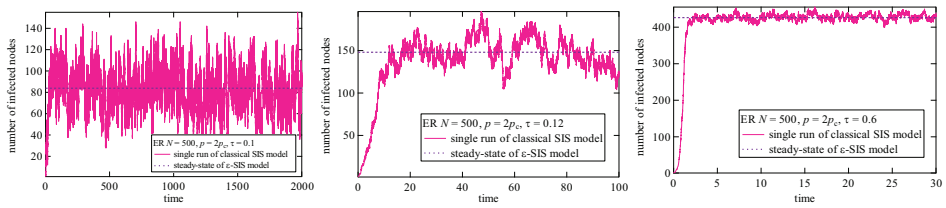
Before turning to the comparison results in the various graph types, we discuss the steady-state fraction of infected nodes in the ε–SIS model and the two mean-field approximations.

2.3.1. THE ε–SIS MODEL

In this chapter, we use the ε–SIS model as a benchmark to compare both mean-field approximations. Whereas the classical SIS model has an absorbing state, the ε–SIS model does not for $\varepsilon > 0$. The non-zero steady-state of the ε–SIS model is reached as time progresses. We believe that the steady-state fraction of infected nodes in the ε–SIS model is the simplest way to determine the number of infected nodes in the meta-stable state of the SIS model. Another approach would be to remove the absorbing state by not allowing the last infected node to cure by reinfecting it. Alternatively, the simulation process can be reset to a previous state with a probability that depends on the state distribution of the simulation up to the point of extinction [32, 33]. The meta-stable state of the classical SIS model, although easily recognised, is difficult to define precisely. One approach would be to run many independent instances of the virus spreading process and calculate the average number of infected nodes at sampled points in time and look for a platau. This will, however, lead to too low an average number of infected nodes as a function of time, as for smaller values of the effective infection rate, many instances of the virus spreading process die out very quickly. These died-out instances have a large impact on the average number of infected nodes as a function of time. Since instances of the virus that die out quickly do not reach a meta-stable state, they have to be filtered out, but that would require an assessment of how long a "reasonable" outbreak lasts. Such a reasonable outbreak will be dependent on the effective infection rate and on the network topology which makes it infeasible as a simulation method.

As the ε–SIS model has a well defined steady-state, the steady-state number of infected nodes can be computed precisely. We start our simulations with no nodes infected and continue to run for a specified warm-up period. After the warm-up period, the measurement period starts during which we record the average number of infected nodes. For all simulations we have taken the warm-up and measurement period to be 10^7 time units and self-infection rate to be $\varepsilon = 10^{-3}$. We have chosen for a duration of 10^7 time units after careful experimentations. The accuracy of the ε–SIS simulations have been compared to the exact ε–SIS Markov chain (see [26]) for small ($N \leq 10$) networks, where more than 3 digits were accurate for all the considered τ - ranges.

The steady-state number of infected nodes of the ε–SIS model will be close to the average number of infected nodes in the meta-stable state of the SIS model for small values of ε. In Figure 2.3, we show a "reasonable" instance of a virus outbreak together with the steady-state number of infected nodes of the ε–SIS model. These examples illustrate that steady-state average number of infected nodes of the ε–SIS model is precisely the line around which the number of infected nodes in the SIS model varies.



(a) (b) (c)

Figure 2.3: The meta-stable state of the classical SIS model (solid line) and the steady-state of the ε-SIS model (dashed line) in ER graphs with $N = 500$ nodes, and a link probability of twice the connectivity threshold $p = 2p_c$, for $\tau = 0.1$ **(a)**, $\tau = 0.12$ **(b)**, and $\tau = 0.6$ **(c)**. For all graphs ($\varepsilon = 10^{-3}$).

2.3.2. PASTOR-SATORRAS & VESPIGNANI HMF APPROXIMATION

From (2.1) and (2.2), we obtain the set of nonlinear equations

$$\begin{cases} \dfrac{\tau \sum_{k=1}^{N-1} k\text{Prob}[D=k]\rho_k}{E[D]+\tau \sum_{k=1}^{N-1} k\text{Prob}[D=k]\rho_k} - \rho_1 = 0 \\ \dfrac{2\tau \sum_{k=1}^{N-1} k\text{Prob}[D=k]\rho_k}{E[D]+2\tau \sum_{k=1}^{N-1} k\text{Prob}[D=k]\rho_k} - \rho_2 = 0 \\ \vdots \\ \dfrac{(N-1)\tau \sum_{k=1}^{N-1} k\text{Prob}[D=k]\rho_k}{E[D]+(N-1)\tau \sum_{k=1}^{N-1} k\text{Prob}[D=k]\rho_k} - \rho_{N-1} = 0 \end{cases} \tag{2.10}$$

From the nonlinear set (2.10), the densities ρ_1, ρ_2,$\cdots \rho_{N-1}$ can be calculated, and after using (2.6), we obtain the steady-state fraction $y_\infty(\tau)$ of infected nodes.

2.3.3. NIMFA APPROXIMATION

The steady-state of the NIMFA approximation is obtained from (2.7), after letting $t \to \infty$ and $\lim_{t\to\infty} \frac{dv_j(t)}{dt} = 0$, as

$$\beta A V(t) - diag(v_i(t))(\beta A V(t) + \delta u) = 0 \tag{2.11}$$

Written as a nonlinear equation for a single node i, leads to

$$\tau(1 - v_i) \sum_{j=1}^{N} a_{ij} v_j = v_i \tag{2.12}$$

or, alternatively

$$v_{i\infty} = \frac{\beta \sum_{j=1}^{N} a_{ij} v_{j\infty}}{\beta \sum_{j=1}^{N} a_{ij} v_{j\infty} + \delta} = 1 - \frac{1}{1 + \tau \sum_{j=1}^{N} a_{ij} v_{j\infty}} \tag{2.13}$$

The steady-state fraction $y_\infty(\tau)$ of infected nodes can be calculated using (2.13) by solving the system and summing over all v_i.

For example, for the complete graph K_N, when $t \to \infty$, $v_{i\infty} = y_\infty$, from which the fraction of infected nodes (2.13) reduces to

$$y_\infty = 1 - \frac{1}{1 + \tau(N-1) y_\infty}$$

or

$$y_\infty = 1 - \frac{1}{(N-1)\tau} \tag{2.14}$$

which is exactly the same as for the HMF approximation in (2.10) when $\rho_k = \rho_{N-1} = \rho = y_\infty$, as also illustrated in Fig 2.7. Obviously, 2.14 only holds for $\tau > \frac{1}{(N-1)}$

2.4. COMPARISON OF $y_\infty(\tau)$ VERSUS τ

In this section, we compare the ε–SIS model and two approximations in different graph types. We take the following topologies into account: the bipartite graph, the star graph, the complete graph, the lattice graph, the path graph, the Erdős-Rényi random graph, the Bárabasi-Albert scale-free graph and the small-world graph. The steady-state fraction $y_\infty(\tau)$ of infected nodes is calculated for increasing effective infection rates τ and $\varepsilon = 10^{-3}$. The values of the NIMFA approximation, the HMF approximation and the simulations of the ε–SIS spreading model are shown in blue, red and green lines respectively. The different markers indicate the size of the graphs, e.g. circles in Figure 2.4 indicate the results for graphs with $N = 10$ nodes.

2.4.1. COMPLETE BIPARTITE GRAPHS

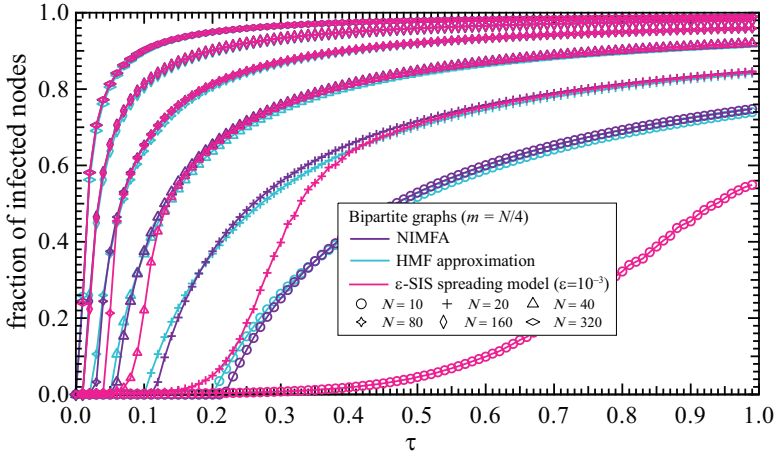


Figure 2.4: Comparison in bipartite graphs.

A complete bipartite graph K_{M_1,M_2} consists of two disjoint sets S_1 and S_2 containing respectively M_1 and M_2 nodes. All nodes in S_1 are connected to all nodes in S_2, while nodes in the same set are not connected. We take $M_1 = N/4$ nodes, and $M_2 = 3N/4$ nodes. The steady-state fraction $y_\infty(\tau)$ of infected nodes as a function of τ is computed in bipartite graphs with $N = 10$, 20, 40, 80, 160 and 320 nodes. Figure 2.4 shows that the epidemic thresholds for the HMF approximation and the NIMFA approximation are close to that of the ε–SIS spreading model ($\varepsilon = 10^{-3}$) in complete bipartite graphs for $N > 40$. As $\tau_c^{(1)}$ of the NIMFA approximation is nearer to τ_c than τ_c^{HMF} of the HMF approximation, $\tau_c^{(1)}$ provides the better epidemic prediction for the SIS model in the complete

bipartite graph K_{M_1,M_2}. Moreover, in [23] it is proved that $\tau_c \geq \tau_c^{(2)} \geq \tau_c^{(1)}$, which means that the second order NIMFA approximation is closest to the ε–SIS spreading model, and therefore the best in bipartite graphs.

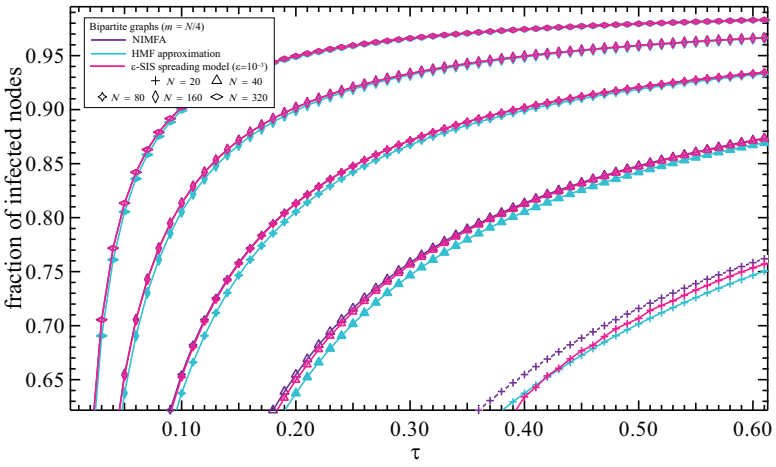


Figure 2.5: Zoom in of the comparison in the bipartite graphs.

Three interesting results can be observed by zooming in on Figure 2.4 as shown in Figure 2.5. First, the NIMFA approximation is an upper bound of the ε–SIS spreading model. Second, the difference between the NIMFA approximation and the ε–SIS spreading model decreases with N. We observe that the NIMFA approximation almost overlays the ε–SIS spreading model, when $N = 320$. Third, the HMF approximation is lower than the ε–SIS spreading model, illustrating that the HMF approximation is not upper bounding the SIS model.

2.4.2. STAR GRAPHS

The star graph $K_{1,N-1}$ is a special bipartite graph where one of the disjoint sets contains only one node while the other set contains all the other nodes.

The epidemic threshold for the first order NIMFA approximation is given by $\tau_c^{(1)} = \frac{1}{\lambda_1}$. For any connected graph, the spectral radius is upper bounded [30] by $\lambda_1 \leq \sqrt{2L-N+1}$, and equality is reached for the complete graph K_N, and the star $K_{1,N-1}$. As a star graph contains $L = N-1$ links, we obtain

$$\tau_c^{(1)} = \frac{1}{\sqrt{2L-N+1}} = \frac{1}{\sqrt{N-1}}. \tag{2.15}$$

The second-order mean-field threshold for the star was estimated in [23] to be $\tau_c^{(2)} = \frac{1}{\sqrt{0.53N-1.3}}$, while exact computations indicate that $\tau_c = \frac{1}{\sqrt{N}}\sqrt{\frac{1}{2}\log N + \log\log N + O(1)}$ for large N. Also, using a definition of the epidemic threshold expressed as a function of the survival time of the SIS process, in Chapter 3.3.3 the epidemic threshold $\tau_c^{(ST)}$ is shown to be very close to the exact value.

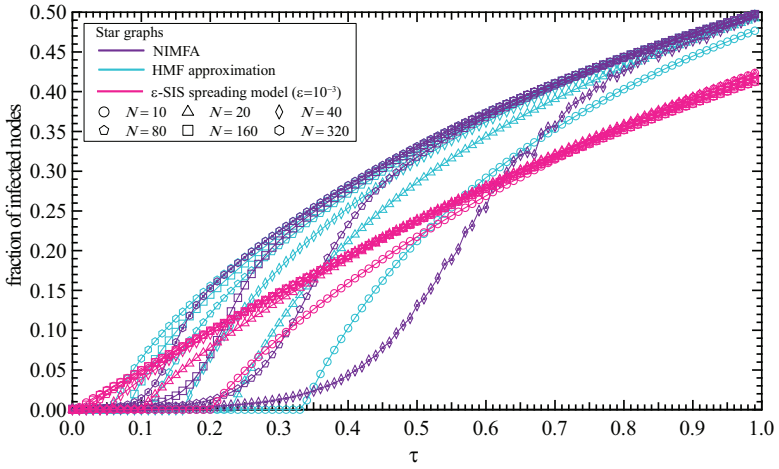


Figure 2.6: Comparison in star graphs.

Recall that the epidemic threshold of the HMF approximation is given by $\tau_c^{\text{HMF}} = \frac{E[D]}{E[D^2]}$. For star graphs it holds that $E[D^2] = \frac{N^2-N}{N}$ and $E[D] = \frac{2(N-1)}{N}$, reducing the HMF threshold to

$$\tau_c^{\text{HMF}} = \frac{2}{N}. \tag{2.16}$$

Equalities (2.15) and (2.16) indicate that, for $N > 2$, the epidemic threshold of the NIMFA approximation is larger than that of the HMF approximation in star graphs. Figure 2.6 shows the superiority of the NIMFA approximation, especially when N is large. Nevertheless, the two epidemic thresholds are both quite far from the threshold of the ε–SIS spreading model ($\varepsilon = 10^{-3}$) in star graphs.

2.4.3. COMPLETE GRAPHS

The complete graph K_N is a graph in which every node is connected to every other node. For a complete graph $\tau_c^{\text{HMF}} = \frac{E[D]}{E[D^2]} = \frac{N-1}{N(N-1)^2/N} = \frac{1}{N-1}$, at the same time $\lambda_1 = N-1$. Hence, the epidemic threshold of the NIMFA approximation $\tau_c^{(1)} = \frac{1}{\lambda_1}$ is equal to the threshold of HMF approximation $\tau_c^{HMF} = \frac{E[D]}{E[D^2]}$. For K_N, both approximations are very close to the ε–SIS spreading model ($\varepsilon = 10^{-3}$), as is shown in Figure 2.7. This is to be expected, since the mean-field approximation in the NIMFA approximation is best for dense graphs, as explained in [12]. Moreover, for K_N, the steady-state equations (see Sections 2.3.3 and 2.3.2) in the NIMFA and HMF approximation are the same: $y_\infty = 1 - \frac{1}{(N-1)\tau}$.

2.4.4. SQUARE LATTICE GRAPHS

The square lattice graph is a two-dimensional grid. Ignoring the boundary nodes, the square lattice can be regarded as a regular graph, where all nodes have the same degree ($k = 4$). In this case, the equations of the NIMFA approximation and the HMF approx-

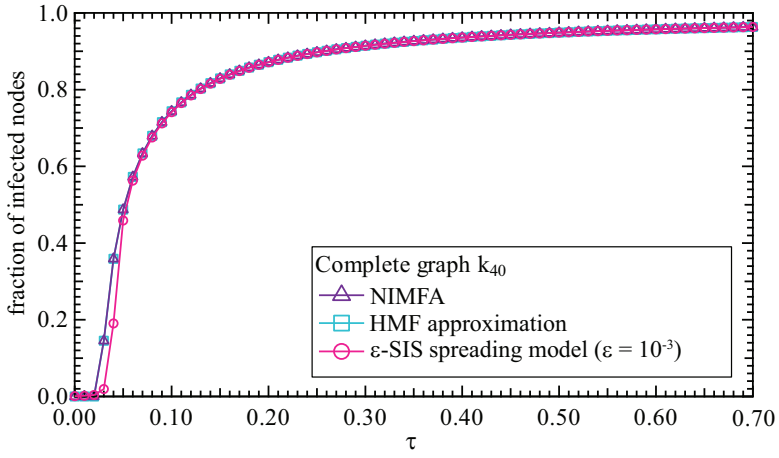


Figure 2.7: Comparison in complete graphs.

imation are almost the same, as can also be seen from the simulations. Figure 2.8 shows

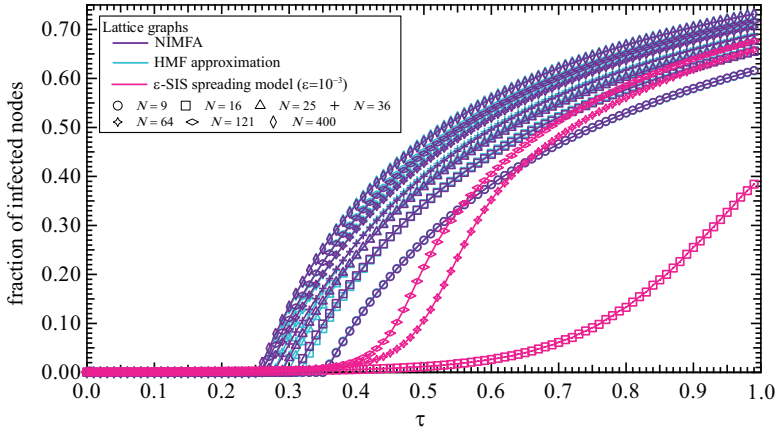


Figure 2.8: Comparison in square lattice graphs.

that the epidemic threshold of the ε–SIS spreading model ($\varepsilon = 10^{-3}$) decreases with the size N of the network. The HMF approximation performs a bit better than the NIMFA approximation in approaching the ε–SIS spreading model in lattice graphs. The simulation illustrates that both the NIMFA approximation and the HMF approximation do not predict the epidemic threshold for epidemic processes in lattices. In Chapter 3.5 it is shown that the time behaviour of the SIS process in the square lattice is also not accurately described by NIMFA. We remark that, in the related process of percolation, the critical probability [34–36] on the square lattice is equal to 1/2.

2.4.5. PATH GRAPHS

The path graph is an example of a tree graph in which every root node has only one branch, and only the last root node is not branched at all. As shown in Figure 2.9, the steady-state fraction $y_\infty(\tau)$ of infected nodes of the NIMFA approximation and the HMF approximation are far from that of the ε- SIS spreading model ($\varepsilon = 10^{-3}$). The epidemic thresholds in the NIMFA approximation and the HMF approximation are both near 0.5, since the average degree of the path graph is 2, ignoring boundary nodes. However, the steady-state fraction $y_\infty(\tau)$ of infected nodes of the ε–SIS spreading model increases very slowly with τ between $0 \leq \tau \leq 1$, and seems to always be around 10^{-3} in the range of network sizes that we considered. Clearly, the true epidemic threshold is much larger than 0.5. In chapter 3.3.3 it is shown that for the ring graph the true epidemic threshold is around 2.5 - 3 times $\tau_c^{(1)}$, which suggests that for the path graph τ_c is around 1.25-1.5.

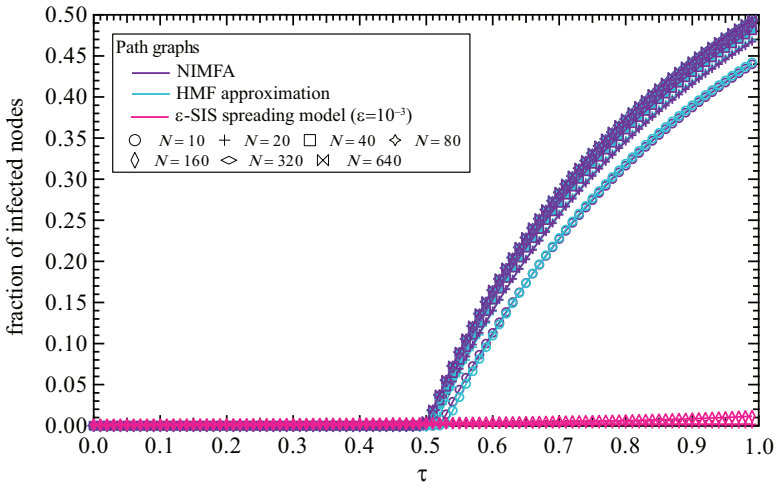


Figure 2.9: Comparison in path graphs.

2.4.6. ERDŐS-RÉNYI RANDOM GRAPHS

In this section we investigate the thresholds in Erdős-Rényi random graphs[1] (ER), which have a binomial degree distribution [37]. An Erdős-Rényi random graph is connected with high probability, if $p > p_c \approx \frac{\ln N}{N}$ for large N; p_c is called the connectivity threshold. All the graphs in the simulations are generated with $p = 2p_c$, and checked for connectivity. Figure 2.10 shows that the steady-state fraction $y_\infty(\tau)$ of infected nodes in the NIMFA approximation and the HMF approximation for ER graphs with N = 10, 20, 40 and 80, are extremely close. However, they both differ from the epidemic threshold of the ε–SIS spreading model, especially when N is small. When N is large, the two approximations are close to the ε–SIS spreading model ($\varepsilon = 10^{-3}$), as is shown in Figure 2.11.

[1] An Erdős-Rényi random graph can be generated from a set of N nodes by randomly assigning a link with probability p to each pair of nodes.

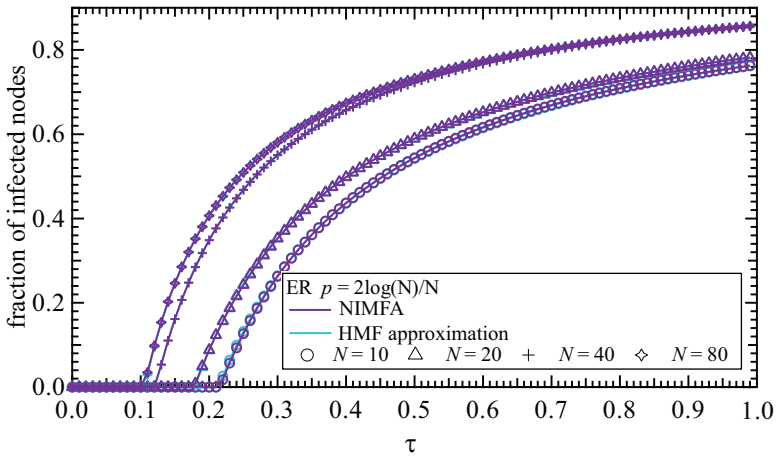


Figure 2.10: Comparison in Erdős-Rényi random graphs.

2.4.7. BÁRABASI-ALBERT SCALE-FREE GRAPHS

The Bárabasi-Albert graph[2] (BA) [38] is a graph model that captures the power-law degree distribution often seen (or approximately seen) in real-world networks. The steady-state fraction of infected nodes as a function of the effective infection rate $y_\infty(\tau)$ is computed in a BA graph with $N = 1000$ and $m = 4$ and shown in Figure 2.12. The NIMFA approximation is close to the HMF approximation, but a little superior. This is to be expected, since the NIMFA approximation is better than the HMF approximation in star graphs as explained in Section 2.4.2, and the BA model can be regarded as a set of hubs with star graph features.

2.4.8. WATTS-STROGATZ SMALL-WORLD GRAPHS

Watts-Strogatz small-world graphs[3] (WS) [39] have two main properties: a small average hopcount $E[H]$, similar to Erdős-Rényi random graphs, and a high clustering coefficient C_G, similar to a ring lattice. The structural properties of small-world graphs have been found in various real-world networks, including social networks [40], neural networks [41] and biological oscillators [42]. In this chapter, the WS graphs are generated with $N = 40$ and 80, $k_s = 6$ and $p = 0.1$ and 1. In Figure 2.13 the steady-state fraction $y_\infty(\tau)$ of infected nodes, as predicted by the two approximations are shown together with the ε–SIS simulations. The NIMFA approximation and the HMF approximation are quite close to each other, but far away from the ε–SIS spreading model. The epidemic thresholds $\tau_c^{(1)} = \frac{1}{\lambda_1}$ and $\tau_c^{\text{HMF}} = \frac{E[D]}{E[D^2]}$ in small-world graphs are close to each other, no matter what N and p are. This can be explained by observing that most nodes have the same degree

[2] A Bárabasi-Albert graph starts with m nodes. At every time step, we add a new node with m links that connect the new node to m different nodes already present in the graph. The probability that a new node will be connected to node i in step t is proportional to the degree $d_i(t)$ of that node. This is referred to as preferential attachment.

[3] A Watts-Strogatz small-world graph can be generated from a ring lattice with N nodes and k edges per node, by rewiring each link at random with probability p.

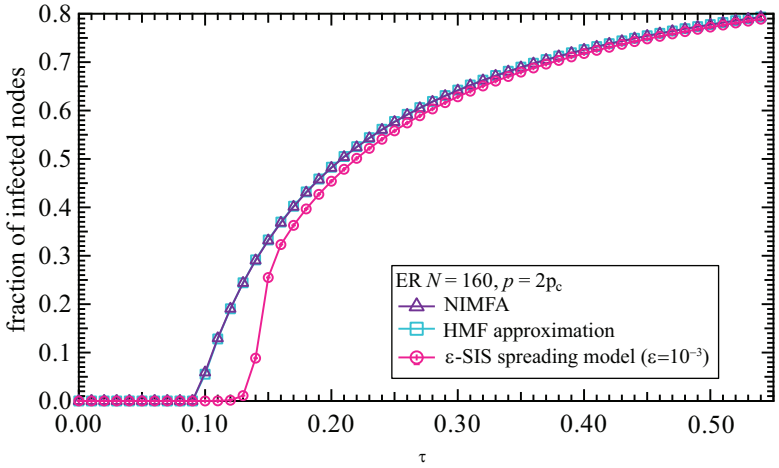


Figure 2.11: Comparison among the NIMFA approximation, Pastor-Satorras approximation and the ε- SIS model in ER network (N = 160).

in WS graphs, justifying the approximation of $E[D^2]$ by $E[D]^2$ and $\tau_c^{\mathrm{HMF}} = \frac{E[D]}{E[D^2]}$ by $\frac{1}{E[D]}$. Another consequence of the similar node degrees in WS graphs is that $E[D]$ is close to $D_{\max}$. Since λ_1 is bounded from below and above as $E[D] \leq \lambda_1 \leq D_{\max}$ [30, art. 43, pp. 46 and art. 48, pp.52], we can approximate λ_1 by $E[D]$, and $\tau_c^{(1)}$ by $\frac{1}{E[D]}$, just as τ_c^{HMF}.

2.5. ANALYTIC COMPARISON OF $\tau_c^{(1)}$ AND τ_c^{HMF}

In this section, we analyse, after [43] the relation between the first order epidemic threshold of NIMFA approximation $\tau_c^{(1)} = \frac{1}{\lambda_1}$ and the epidemic threshold of the HMF approximation $\tau_c^{\mathrm{HMF}} = \frac{E[D]}{E[D^2]}$. From the comparison in Section 2.4, we find that the relation between the two epidemic thresholds strongly depends on the graph type. The two epidemic thresholds are identical in regular graphs where each node has degree r increasing with N. Indeed, since $\lambda_1 = E[D] = r$ (see [30, art. 43, pp. 46]), and $\tau_c^{\mathrm{HMF}} = \frac{1}{r}$, we find that $\tau_c^{(1)} = \tau_c^{\mathrm{HMF}}$. There are graphs for which $\tau_c^{(1)} < \tau_c^{\mathrm{HMF}}$, while in most cases, our simulations in Figures 2.4, 2.6, 2.10 and 2.12 demonstrate that $\tau_c^{(1)} > \tau_c^{\mathrm{HMF}}$.

Case $\tau_c^{(1)} < \tau_c^{\mathrm{HMF}}$*:* The epidemic threshold τ_c^{HMF} is larger than the first order threshold $\tau_c^{(1)} = \frac{1}{\lambda_1}$, when the assortativity[4] ρ_D is zero. Van Mieghem *et al.* [30, 45] have reformulated the assortativity as follows

$$\rho_D = \frac{N_1 N_3 - N_2^2}{N_1 \sum_{i=1}^{N} d_i^3 - N_2^2} \tag{2.17}$$

where, $N_k = u^T A^k u$ is the total number of walks with k hops. In [46], it is proved that $\lambda_1 \geq \frac{N_2}{N_1} = \frac{E[D^2]}{E[D]} = \frac{1}{\tau_c^{\mathrm{HMF}}}$, when $\rho_D = 0$.

[4]The degree correlation, also called the assortativity ρ_D is computed as the linear correlation coefficient of the degree of nodes connected by a link [44]. It describes the tendency of network nodes to connect preferentially to other nodes with either similar (when $\rho_D > 0$) or opposite (when $\rho_D < 0$) properties i.e., degree.

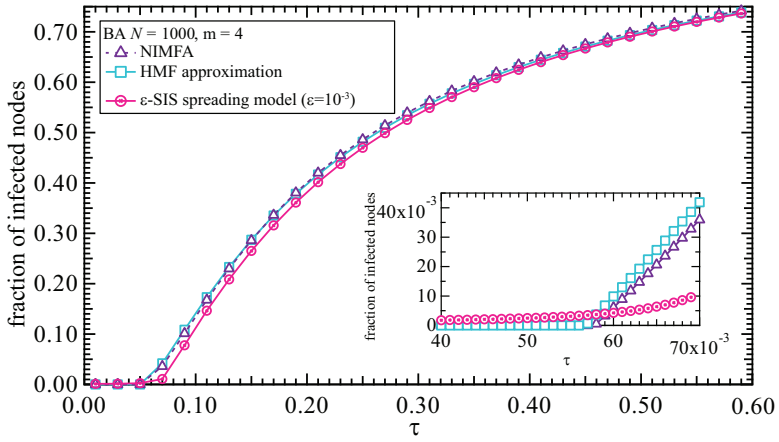


Figure 2.12: Comparison in Bárabasi-Albert Scale-free graphs.

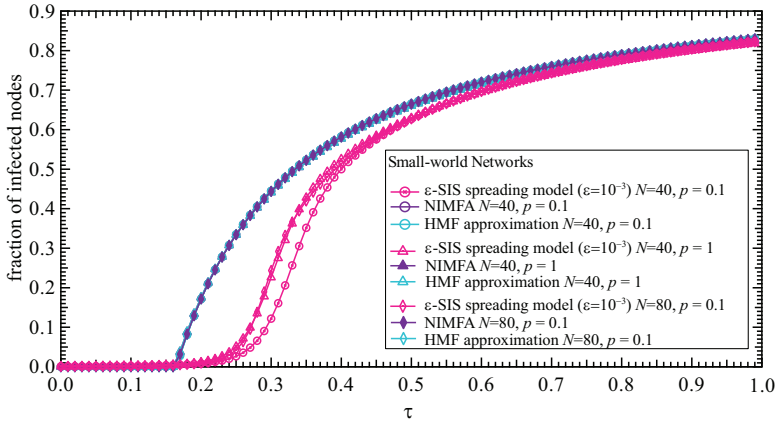


Figure 2.13: Comparison in WS Small-world graphs.

Case $\tau_c^{(1)} > \tau_c^{\text{HMF}}$*:* Newman [44] pointed out that the assortativity ρ_D of the ER graph and the BA graph is zero when N is large. However, in most ER and BA graphs with finite size, the assortativity is only *approximately* zero. Our simulations in Figures 2.10 and 2.12 show that $\tau_c^{\text{HMF}} \leq \tau_c^{(1)}$ in ER and BA graphs, demonstrating that the precise $\rho_D = 0$ condition in (2.17) that led to $N_1 N_3 = N_2^2$ is not valid. Moreover, we have already proved $\tau_c^{\text{HMF}} \leq \tau_c^{(1)}$ in star graphs (see Section 2.4.2).

It would be interesting to find all or the most prominent graph classes in which $\tau_c^{(1)} > \tau_c^{\text{HMF}}$ and in which $\tau_c^{(1)} < \tau_c^{\text{HMF}}$.

2.6. CHAPTER SUMMARY

Many approximations of the SIS model have been proposed to understand SIS epidemics. In this chapter, we study which mean-field approximation, the NIMFA or the HMF, is

better in approaching the SIS epidemic model. A direct comparison to the SIS model is, however, not possible, because the steady-state of the exact SIS model in a finite network is the overall-healthy state. Although an infection in the SIS model will eventually die out, for high enough effective infection rates the fraction of infected nodes as a function of time is meta-stable. We propose to define the number of infected nodes in the meta-stable state of the SIS model via the number of infected nodes in the steady-state of the ε–SIS model for a prescribed small value of ε. From the comparison between the NIMFA and HMF approximations with the ε–SIS spreading model, we conclude that, overall, the NIMFA approximation is better than the HMF approximation, except in square lattice graphs and path graphs. We show that the NIMFA approximation can approach the ε–SIS epidemic model well in most graph types. The simulations show that the NIMFA approximation almost overlaps with the ε–SIS spreading model when the size of network is large enough. While the HMF approximation is better than the NIMFA approximation in the square lattice and path graphs, the difference between the two is small. Moreover, they are both far away from the ε–SIS spreading model. We also show that the NIMFA approximation and the HMF approximation are exactly the same in regular graphs with the degree of nodes increasing with N, such as complete graphs, and are similar in small-world graphs. In addition to the simulation results, we show analytically the conditions under which the epidemic threshold of the NIMFA approximation is larger than, smaller than or equal to that of the HMF approximation.

3

SURVIVAL TIME OF AN SIS EPIDEMIC

Using the hitting time of the absorbing state in a uniformised embedded Markov chain of the continuous-time Markov chain describing the SIS process, we derive an exact expression for the average worst-case survival time of a virus in the complete graph K_N and the star graph $K_{1,N}$. Our results for the complete graph show that the approximation of the average survival time by the first term in the Lagrange series of the infinitesimal generator in [47] is in fact the real average worst-case survival time. Also via the hitting time, we derive a sharper expression for the epidemic threshold in K_N and $K_{1,N}$ than the reciprocal of the largest eigenvalue.

3.1. INTRODUCTION

An interesting question in the context of SIS virus spread on networks is how many nodes, on average, will be infected during an outbreak. As shown in the previous chapter, it has long been observed that an epidemic threshold [17, 31, 48] separates the regime where an infection dies out quickly, from the regime where the expected number of infected nodes is stable over a long time, called the meta-stable or quasi-stationary state. The main concern of this chapter is the question: "how long will the virus stay in this meta-stable state".

Building on the work in [47], we derive a simpler summation over recursive terms as an expression for the average survival time of an SIS process in the complete graph. Based on the hitting time of the absorbing state in the uniformised embedded Markov chain, we derive an exact solution for the average survival time in the complete graph and the star graph. The solution for the complete graph turns out to be exactly the same expression as the result in [47]. We use the complete solution to the hitting time of the absorbing state in both the complete graph and the star graph to give a definition of the epidemic threshold that scales in the number of nodes identical to the "true" epidemic threshold [49]. The effects of non-Poissonian infection and curing processes on the sur-

vival time is treated in Chapter 5.4.

3.2. THE AVERAGE SURVIVAL TIME IN K_N

Let T denote the time for the SIS epidemic process to reach the overall-healthy state, which we call the survival time, and is also known as the time to absorption or the extinction time. The time to extinction starting from the quasi-stationary distribution in a continuous-time Markov chain containing an absorbing state is exponentially distributed as $Pr[T \leq t] = 1 - \exp(-t\zeta)$, where ζ is the second largest eigenvalue of the infinitesimal generator Q of the Markov chain [50]. For an irreducible continuous-time birth-and-death process (SIS on the complete graph reduces to a birth-and-death process) the hitting time of the absorbing state starting with n individuals is distributed as the sum of n independent exponential random variables with parameters equal to the eigenvalues of Q [51, 52]. A recent result [47] for SIS epidemics on K_N is the accurate expression, for large N, of the decay rate $\zeta < 0$ of the survival time for effective infection rates $\tau > \tau_c$. The average survival time $E[T]$ of the SIS process in K_N is approximately equal to $|\zeta^{-1}|$, with

$$-\zeta = \frac{\delta}{F(\tau)} + O\left(\frac{N^2 \log N}{(\tau N)^{2N-1}}\right), \tag{3.1}$$

where

$$F(\tau) = \sum_{j=1}^{N} \sum_{r=0}^{j-1} \frac{(N-j+r)!}{j(N-j)!} \tau^r. \tag{3.2}$$

We show that (3.2) can be rewritten as

$$F(\tau) = \sum_{j=1}^{N} \frac{x_j}{j}, \tag{3.3}$$

where x_j obeys the recursive relation

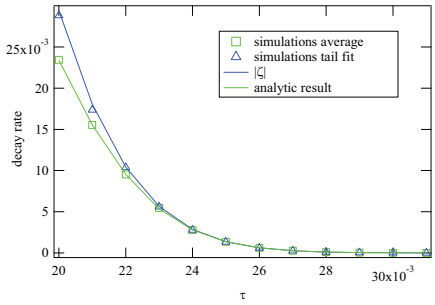
$$x_{j+1} = x_j(N-j)\tau + 1, \tag{3.4}$$

with $x_j = \sum_{r=0}^{j-1} \frac{(N-j+r)!}{(N-j)!} \tau^r$ and initial conditions $x_1 = 1$, $x_j = 0$ for $j < 0$. Indeed,

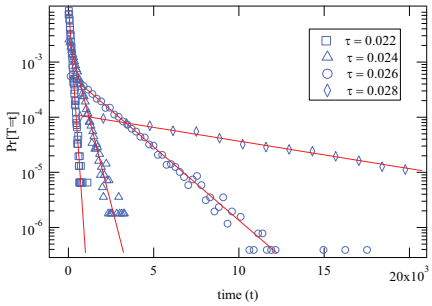
$$\begin{aligned} x_{j+1} &= \sum_{r=0}^{j} \frac{(N-j-1+r)!}{(N-j-1)!} \tau^r = \tau(N-j) \sum_{r=-1}^{j-1} \frac{(N-j+r)!}{(N-j)!} \tau^r \\ &= x_j(N-j)\tau + \tau(N-j)\frac{(N-j-1)!}{(N-j)!}\tau^{-1} = x_j(N-j)\tau + 1 \end{aligned}$$

The summation over recursive terms (3.3) is more efficient to compute numerically, as it contains only N terms instead of $\frac{N^2}{2}$ in the double sum in (3.2) and does not contain factorials.

Equation (3.2) is the first term in the Lagrange series of the second largest eigenvalue ζ of the infinitesimal generator of the continuous-time Markov chain describing the SIS process (see Section 3.3.1), and as such is expected to be close to the real ζ. To verify this, we determine the decay rate of the survival time in a complete graph of 64 nodes in



(a) The decay rate $|\zeta|$ as a function of the effective infection rate τ in a complete graph of 64 nodes determined in four different ways: average survival time (squares), fitting the histogram of the survival time (triangles), ζ (green line), and eq (3.3) (blue line).



(b) Histograms of the survival time in units of $\delta = 1$ in a complete graph of 64 nodes for effective infection rate τ ranging from 0.022 to 0.028. The epidemic threshold in K_N is close to $\frac{1}{N-1} = 0.016$.

Figure 3.1: Decay rate in a complete graph of 64 nodes (a) and histograms of the survival time (b).

four different ways: (i) via the second largest eigenvalue of the infinitesimal generator ζ, (ii) via an exponential fit to the histogram of simulated survival times, (iii) via the reciprocal of the average of the simulated survival times, (iv) and as computed from (3.2). All simulations start in the all-infected state and the time till extinction is measured.

Figure 3.1a shows that the decay rate found by the four different methods all converge to the same value as the effective infection rate τ increases. For relatively small values of τ, however, the four methods split in two groups. The second largest eigenvalue ζ and the fit of the survival times give the same decay rate, and the average survival time and (3.3) give the same decay rate, which leads to two conclusions. First, the second largest eigenvalue is indeed the most dominant eigenvalue and a good approximation for the *tail* of the survival time distribution for the investigated range of τ. For large τ, so much of the weight of the distribution is in the tail that the average of the distribution of T is well approximated by ζ^{-1}, but not for small τ. As the average survival time starting from the quasi-stationary or meta-stable state is given by $|\zeta|^{-1}$, it is to be expected that for small values of τ starting from the all-infected state and the quasi-stationary state leads to increasingly different survival times. Second, and more interesting, (3.3), that was derived as an approximation of ζ, precisely follows the real average survival time. In the derivation of the approximation to the second largest eigenvalue of the infinitesimal generator in [47], an expression for the average survival time was found. In Section 3.3.1 it is shown using the average hitting time of the absorbing state, that (3.3) is the exact average survival time in the complete graph.

Figure 3.1b shows the histograms and fitted exponential functions for four values of the effective infection rate. These results confirm that the survival time of an SIS virus in the complete graph is exponentially distributed and that the decay rate in the complete graph is indeed given by (3.1;3.3).

3.3. SURVIVAL TIME VIA HITTING TIME

An alternative to finding the survival time of the SIS process on a complete graph via the second largest eigenvalue of the infinitesimal generator, is the hitting time of the absorbing state in the continuous-time Markov chain. As discussed in Chapter 2.1, SIS process on the general graph can be written as a 2^N state continuous-time Markov chain. Computing the hitting time for a Markov chain with such a large state space is only feasible for very small graphs. For graphs where the infectious state shows some form of symmetry, however, the number of states can be dramatically reduced. Unfortunately, symmetry in the graph structure or node degree alone is not enough to reduced the size of the state space. In this section, we exploit the symmetry in the infectious state of the complete graph and the star graph, who posses a simplified Markov Chain with $O(N)$ states, to compute the hitting time of the absorbing state. Using the hitting time, we derive an alternative expression for the epidemic threshold, which we compare to the exact results on the complete and star graph in [49].

We use an embedded Markov chain to transform the continuous-time Markov chain to a discrete-time one. The embedded Markov chain of a continuous-time process contains the transition probabilities at the time of a transition, but no longer contains the precise timing of the events (see [53]). The average hitting time of the absorbing state in the embedded chain, starting from the all-infected state, gives the average number of transitions between the initial state and reaching the absorbing state. However, the time between transitions is unknown. By introducing self-loops in the embedded Markov chain, the average transition rate from state i to j ($i \neq j$) in the embedded chain is made identical to the transition rate in the original continuous-time Markov chain. The transition rate, including self-loops, is uniform, which enables relating the number of transitions to time.

The transition matrix of the uniformised embedded Markov chain in units of ϕ of a continuous-time Markov chain is given by $S(\phi) = I + \frac{Q}{\phi}$, where Q is the infinitesimal generator of the continuous-time Markov chain [53]. Transitions in the uniformised Markov chain all occur with the same rate $\phi \geq \max_i q_i$, where q_i is the total outgoing rate of state i: $q_i = \sum_{j=1, j\neq i} q_{ij}$, with q_{ij} the transition rate from state i to state j.

The average hitting time of the absorbing state in a discrete Markov chain is given by the minimal non-negative solution of the following system [53, 54]:

$$\begin{cases} w_1 = 0 \\ w_i = 1 + \sum s_{ij}(\phi) w_j \quad \text{for } i \neq 1 \end{cases} \tag{3.5}$$

where w_i is the average hitting time of the absorbing state starting from state i, and s_{ij} is the transition probability from state i to j in the uniformised embedded Markov chain.

The system of equations (3.5) can be written as $Kw = b$, where $K = I - S(\phi)$. The vector b is defined as $b = \boldsymbol{u} - e_1$ where $\boldsymbol{u}$ is the all-one vector and e_1 is the first standard basis vector. The minimal non-negative solution of system (3.5) for w_i will give the average hitting time of the absorbing state when starting in state i in units of ϕ^{-1}, where as multiplying K by ϕ will give the average hitting time in units of 1. Therefore, we solve the system

$$\phi K w = b, \tag{3.6}$$

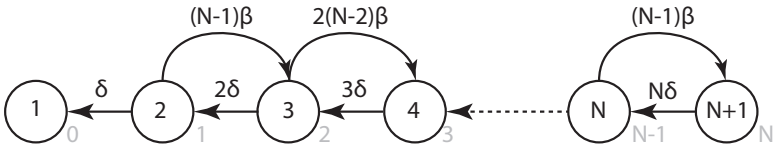


Figure 3.2: State diagram and transitions of the SIS process in the complete graph.

where ϕK simplifies to $\phi K = \phi(I - I + \frac{Q}{\phi}) = -Q$.

3.3.1. THE COMPLETE GRAPH

The SIS process on a complete graph can be described by a continuous-time Markov chain, as illustrated in Figure 3.2. States are numbered from 1 to N, while the number of infected nodes in state i is $i-1$. The infinitesimal generator is given by:

$$-Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1+(N-1)\tau & -(N-1)\tau & 0 & 0 & 0 & 0 \\ 0 & -2 & 2+2(N-2)\tau & -2(N-2)\tau & 0 & 0 & 0 \\ 0 & 0 & -3 & 3+3(N-3)\tau & -3(N-3)\tau & 0 & 0 \\ 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 \\ 0 & 0 & 0 & 0 & 0 & -N & N \end{bmatrix}$$

System (3.6) can be solved using Gaussian elimination, that is, by reducing the augmented matrix $[-Q|b]$ to row echelon form. Because we are interested in the worst-case scenario, we only need to know the value of w_{N+1}, which corresponds to the hitting time of the absorbing state starting from the all-infected state. In Gaussian elimination, the system matrix is reduced to an upper-triangular matrix. Reducing all non-zero subdiagonal elements from column k for $k \geq 2$ to zero is achieved by adding $\frac{k}{d_k}$ times row k to row $k+1$:

$$r_{k+1} \rightarrow r_{k+1} + \frac{k}{d_k} r_k,$$

where r_k is row k in $[-Q|b]$. The multiplication factor d_k can be found recursively for $k > 2$ as

$$d_k = (k-1)(1+(N-k+1)\tau) - \frac{(k-1)(k-2)(N-k+2)\tau}{d_{k-1}}, \tag{3.7}$$

with initial conditions $d_1 = 1$ and $d_2 = 1+(N-1)\tau$. Since all row operations are performed on the augmented system matrix, the entries in b also change. The average survival time starting from the all-infected state, is found by dividing the last element in b by $w_{N+1} = d_{N+1}$:

$$E[T] = b_{N+1}/d_{N+1}, \tag{3.8}$$

where b_k is given by the recursive relation for $k > 1$ as

$$b_k = 1 + \frac{(k-1)}{d_{k-1}} b_{k-1}.$$

and initial condition $b_1 = 0$.

We will now show that (3.8) is equal to (3.3), and, as a consequence, that the first term in the Lagrange series of the second largest eigenvalue ζ of the infinitesimal generator Q is precisely the average worst-case survival time of the SIS process on K_N.

We first rewrite b_{N+1}:

$$\begin{aligned} b_{N+1} &= 1 + \frac{N}{d_N} b_N \\ &= 1 + \frac{N}{d_N}(1 + \frac{(N-1)}{d_{N-1}} b_{N-1}) \\ &= 1 + \frac{N}{d_N} + \frac{N(N-1)}{d_N d_{N-1}} b_{N-1} \end{aligned}$$

Repeating this process for all b_i until $i = 1$ results in

$$b_{N+1} = \sum_{j=0}^{N-1} \frac{N! \prod_{i=2}^{N-j} d_i}{(N-j)! \prod_{i=2}^{N} d_i}$$

Finally, dividing by d_{N+1} yields the average survival time

$$E[T] = \sum_{j=0}^{N-1} \frac{N! \prod_{i=2}^{N-j} d_i}{(N-j)! \prod_{i=2}^{N+1} d_i}. \tag{3.9}$$

Next, we write:

$$p_j = \prod_{i=2}^{j} d_i,$$

then it follows that

$$p_j = d_j p_{j-1} = d_j \prod_{i=2}^{j-1} d_i, \tag{3.10}$$

and

$$d_j = \frac{p_j}{p_{j-1}}.$$

substituted in (3.7) yields

$$d_j = (j-1)(1 + (N-j+1)\tau - \frac{(j-1)(j-2)(N-k+2)\tau p_{j-2}}{p_{j-1}}. \tag{3.11}$$

Substituting (3.11) in (3.10) leads to the following recursive expression for p_j:

$$p_j = (j-1)(1 + (N-j+1)\tau) p_{j-1} - (j-1)(j-2)(N-j+2)\tau p_{j-2},$$

which can be further simplified by first splitting and then iterating the first term:

$$\begin{aligned} p_j &= (j-1)p_{j-1} - (j-1)(j-2)(N-j+2)\tau p_{j-2} + (j-1)(N-j+1)\tau p_{j-1} \\ &= (j-1)(j-2)(1 + (N-j+2)\tau) p_{j-2} - (j-1)(j-2)(j-3)(N-j+3)\tau p_{j-3} \\ &\quad - (j-1)(j-2)(N-j+2)\tau p_{j-2} + (j-1)(N-j+1)p_{j-1} \\ &= (j-1)(j-2)p_{j-2} - (j-1)(j-2)(j-3)(N-j+3)\tau p_{j-3} + (j-1)(N-j+1)p_{j-1}. \end{aligned}$$

The first term can be expanded recursively to arrive, with $p_2 = d_2 = 1 + (N-1)\tau$, at

$$\begin{aligned} p_j &= (j-1)(j-2)...2p_2 - (j-1)(j-2)...(N-1)\tau p_1 + (j-1)(N-j+1)p_{j-1} \\ &= (j-1)!(1+(N-1)\tau) - (j-1)!(N-1)\tau + (j-1)(N-j+1)p_{j-1} \\ &= (j-1)! + (j-1)(N-j+1)\tau p_{j-1}. \end{aligned} \tag{3.12}$$

From (3.12), we find that $p_{N+1} = N!$ and (3.9) becomes

$$E[T] = \sum_{j=0}^{N-1} \frac{p_{N-j}}{(N-j)!} = \sum_{k=1}^{N} \frac{p_k}{k!}. \tag{3.13}$$

Moreover, let $y_k = \frac{p_k}{(k-1)!}$, then in (3.12)

$$(j-1)!y_j = (j-1)! + (j-1)(N-j+1)\tau(j-2)!y_{j-1}$$

which, after rearranging terms, yields

$$y_j = 1 + (N-j+1)\tau y_{j-1}$$

Comparing with (3.4), we observe that y_j obeys the same recursion (including the initial conditions) as x_j in (3.4). Due to the uniqueness of the solution of a linear difference equation $y_j = x_j$, so that $E[T] = \sum_{k=1}^{N} \frac{x_k}{k}$, which is precisely (3.2), proving that the first term in the Lagrange series of the second largest eigenvalue ζ of the infinitesimal generator Q is equal to the average worst-case survival time of the SIS process on K_N. As the survival time starting from the meta-stable state is exponentially distributed with a rate equal to ζ, the order term in (3.1) is a measure for the difference between starting in the meta-stable state and in the all-infected state.

Figure 3.3 shows the average survival time of the SIS process in a complete graph on 100 nodes as a function of the normalised effective infection rate $\tau/\tau_c^{(1)}$ computed via the hitting time (3.13) and by simulations. As shown in Chapter 2.2.3, the NIMFA epidemic threshold $\tau_c^{(1)}$ is defined as the reciprocal of the largest eigenvalue λ_1 of the adjacency matrix and is a lower bound of the real epidemic threshold [29, 55], i.e. $\tau_c^{(1)} \leq \tau_c$. The simulations are in perfect agreement with the average hitting time. The approximation of the survival time using the second largest eigenvalue of the infinitesimal generator, that is, $E[T] \geq |\zeta|^{-1}$, is also shown in Figure 3.3 and is very accurate for large effective infection rates.

SURVIVAL TIME DISTRIBUTION AROUND THE EPIDEMIC THRESHOLD

Besides the expectation of the survival time $E[T]$, the uniformised embedded Markov chain enables us to derive the complete pdf of the survival time T. Let P^k be the k-step transition probability matrix of the uniformised embedded Markov chain describing the SIS process on the complete graph, then entry $(N+1,1)$ is the probability that the virus has died out after k steps starting with all nodes infected. The probability that the virus has not died out yet after k steps is then given by $\Pr[T \geq k] = 1 - (P^k)_{N+1,1}$, and $\Pr[T = k] = \Pr[T \geq k] - \Pr[T \geq k+1]$. Figure 3.4 shows the survival time distribution in a complete graph of 100 nodes for values of the effective infection rate τ at different

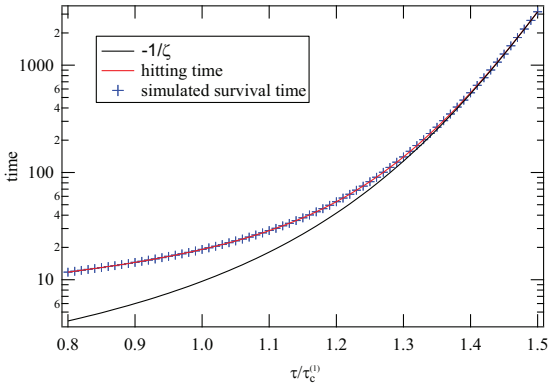


Figure 3.3: Average survival time in a complete graph of 100 nodes as a function of the normalised effective infection rate as computed via the hitting time, simulations and approximated by $|\zeta|^{-1}$.

fractions of the epidemic threshold ranging from 0.8 to 1.2. The curve for $\tau = \tau_c^{(1)}$ is indicated by the dashed black line. Figure 3.4 illustrates that the curves for both above and below the epidemic threshold are similarly shaped. For values of the effective infection rate τ above $\tau_c^{(1)}$, the peak of the distribution reduces and moves slightly towards the right while the exponential tail becomes less steep. The exponential tails of the distributions are drawn on lin-log scale in the inset of Figure 3.4. The inset shows that nothing special happens around the threshold in terms of the survival time distribution of the SIS process. The slopes of the exponential tail become less and less steep for increasing infection rates, but no transition is observed around the epidemic threshold.

When the effective infection rate τ is reduced to zero, the probability density function of the survival time of the SIS process is given by $f_X(x) = \delta I(1-e^{-\delta x})^{I-1}e^{-\delta x}$, where I is the number of initially infected nodes, which is the distribution of the maximum of I i.i.d exponentially distributed random variables [53]:

$$f_X(x) = \frac{d}{dx}\Pr[\max_{1\le m\le I} X_m \le x] = \frac{d}{dx}\prod_{m=1}^{I}\Pr[X_m \le x] = \frac{d}{dx}(1-e^{-\delta x})^I$$
$$= \delta I(1-e^{-\delta x})^{I-1}e^{-\delta x} \qquad (3.14)$$

which may explain the bell-shape in Figure 3.4.

Even if the changes in the survival time distribution are subtle around the epidemic threshold, the influence on the average survival time displays transition behaviour. Figure 3.5 shows the survival time of the SIS process in the complete graph around the epidemic threshold for various network sizes. The threshold behaviour around τ/τ_c is clearly visible. Even for a network of 1600 nodes, the average survival time increases very sharply after the threshold. This is in agreement with the asymptotic approximation of

$$F(\tau) \sim \frac{f\sqrt{2\pi}\exp(N(\log(f)+f^{-1}-1))}{\delta(f-1)^2\sqrt{N}},$$

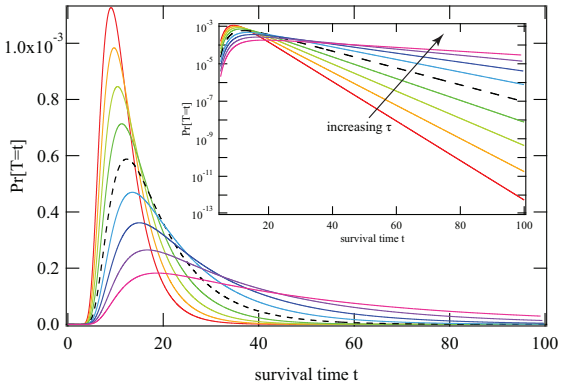


Figure 3.4: Survival time distribution of the SIS process in a complete graph of 100 nodes for different values of the normalised effective infection rate $\frac{\tau}{\tau_c^{(1)}}$ ranging from 0.8 to 1.2. The curve for $\tau = \tau_c^{(1)}$ is indicated by the dashed black line. The inset shows the same data on a lin-log scale to emphasise the exponential tale of all distributions, both above and below the epidemic threshold.

with $f = \tau/\tau_c$ in [47]. Figure 3.5 also shows the average fraction of infected nodes in the NIMFA solution $y_\infty = 1 - \frac{1}{(N-1)\tau}$ on the right-hand side axis for reference. With increasing network size, even small fractions of infected nodes can last in the network for very long times. A comparison with the NIMFA solution also shows that the transition point moves closer towards $\tau_c^{(1)}$ with increasing N, very similar to the peak in $E[T]_{I=1}/E[T]_{I=N}$ in Figure 3.8a.

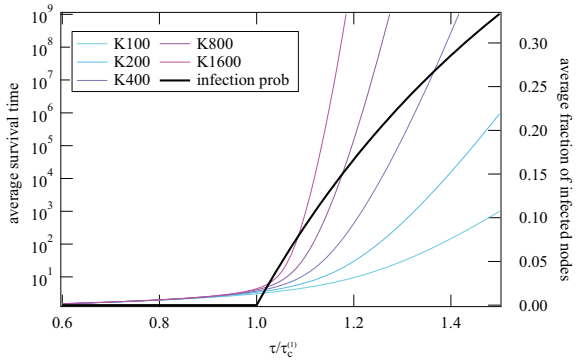


Figure 3.5: The average survival time of the SIS process in the complete graph as a function of the normalised effective infection rate $\frac{\tau}{\tau_c^{(1)}}$ for various network sizes.

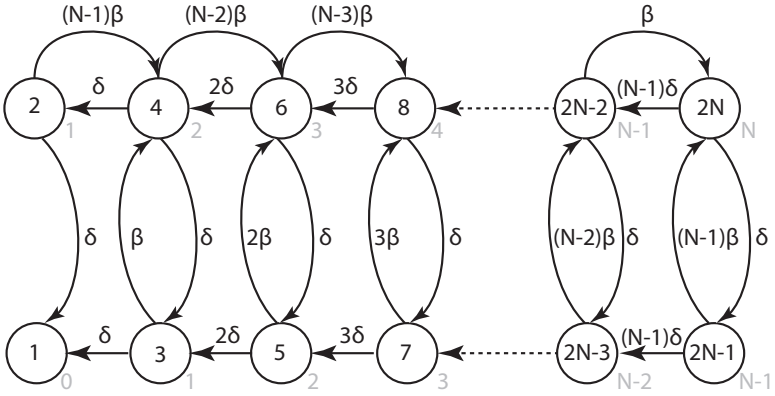


Figure 3.6: Markov Chain representation of the SIS virus process in the star graph. In the odd state the centre node is not infected, whereas in the even states the centre nodes is infected. The number of infected nodes is indicated next to the states.

3.3.2. THE STAR GRAPH

The continuous-time Markov chain for the star graph contains $2N$ states, as shown in Figure 3.6. The Markov chain in Figure 3.6 is a relabeling of the chain in [49] so that the infinitesimal generator has a five-band structure. In the odd states, the centre node is healthy, whereas in the even states it is infected. The number of infected nodes in each state is indicated in blue next to the state. Just as in the case of the complete graph, we solve the system $\phi K w = b$, where $\phi K = -Q$ and $b = \boldsymbol{u} - e_1$ is the all one vector minus the first standard basis, by reducing the augmented matrix $[-Q|b]$ to row echelon form. The system matrix is given by:

$$-Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & q_2 & 0 & -(N-1)\tau & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & q_3 & -\tau & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & -1 & q_4 & 0 & -(N-2)\tau & 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 0 & q_5 & -2\tau & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -2 & -1 & q_6 & 0 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & \ddots & 0 & q_7 & \ddots & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \ddots & -1 & q_8 & 0 & -\tau \\ 0 & 0 & 0 & 0 & 0 & 0 & -(N-1) & 0 & q_9 & -(N-1)\tau \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -(N-1) & -1 & q_{10} \end{bmatrix}$$

Contrary to the complete graph, odd columns have two non-zero elements below the diagonal and reducing them to zero will also lead to two non-zero elements below the diagonal of the next (even) column.

The non-zero sub-diagonal entries of odd columns for $k > 1$ are reduced to zero by

two row operations:

$$r_{k+1} \leftarrow r_{k+1} + \frac{\delta}{d_k} r_k$$
$$r_{k+2} \leftarrow r_{k+2} - \frac{(k+1)\delta}{2d_k} r_k$$

The non-zero sub-diagonal entries of even columns for $k > 2$ are also reduced to zero by two row operations:

$$r_{k+1} \leftarrow r_{k+1} - \frac{m_k k\delta}{2d_{k-1}d_k} r_k$$
$$r_{k+2} \leftarrow r_{k+2} + \frac{k\delta}{2d_k} r_k$$

where d_k is given by:

$$\begin{cases} \frac{k\delta}{2} + (N - \frac{k}{2})\beta - \frac{(k/2-1)(N-k/2+1)\beta\delta}{d_{k-2}} + \frac{\delta m_k}{d_{k-1}} & \text{for even } k > 2 \\ \delta + (N-1)\beta & k = 2 \\ (k-1)(\delta+\beta)/2 & \text{for odd } k > 0 \end{cases}$$

and m_k is given, for $k > 4$, by

$$m_k = -(\frac{k}{2} - 1)\beta + \frac{(k/2-1)(N-k/2+1)\beta\delta m_{k-2}}{d_{k-2}d_{k-3}},$$

with initial condition $m_4 = -\beta$. The worst-case expected survival time in the star graph is obtained by dividing $w_{2N} = d_{2N}$ by the last element in b:

$$E[T] = b_{2N}/d_{2N}$$

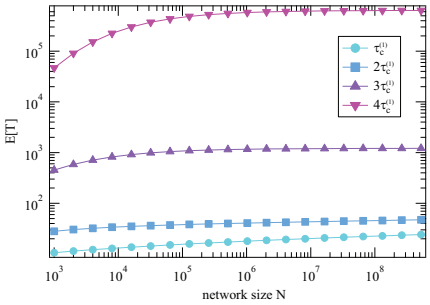
where b_k is given by

$$\begin{cases} 1 + (\frac{k}{2} - 1)\frac{\delta}{d_{k-2}} b_{k-2} + \frac{\delta}{d_{k-1}} b_{k-1} & \text{for } k > 1 \text{ even} \\ 1 + \frac{(k-1)\delta}{2d_{k-2}} b_{k-2} - \frac{(k-1)m_{k-1}\delta}{2d_{k-2}d_{k-1}} b_{k-1} & \text{for } k > 1 \text{ odd,} \end{cases}$$

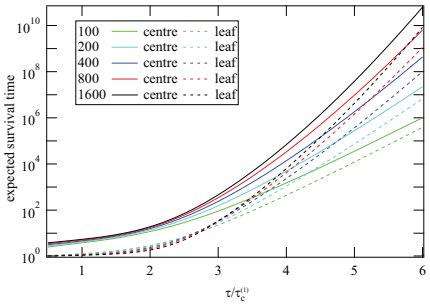
and initial conditions $b_1 = 0$, $b_2 = 1$.

Although the expression for the survival time in the star graph is more involved than in the case of the complete graph, it enables the numerical evaluation of the survival time in a linear time and constant space complexity.

Figure 3.7a shows the survival time as a function of the network size for different values of the effective infection rate $\tau \geq \tau_c^{(1)}$. The average survival time increases logarithmically with the size of the network for relatively small effective infection rates. For effective infection rates larger than approximately 2.5 times the epidemic threshold $\tau_c^{(1)} = 1/\lambda_1 = 1/\sqrt{N-1}$ the expected survival time $E[T]$ of the virus seems to stabilise with increasing network size. The point of stabilisation depends on the effective infection rate τ and is the result of the the scaling of the real epidemic threshold. In Section 3.3.3 the epidemic threshold in K_N and $K_{1,N}$ is determined via the survival time.



(a) Survival time of a virus on a star graph as a function of the size of the graph for various fractions of the epidemic threshold $\tau_c^{(1)}$.



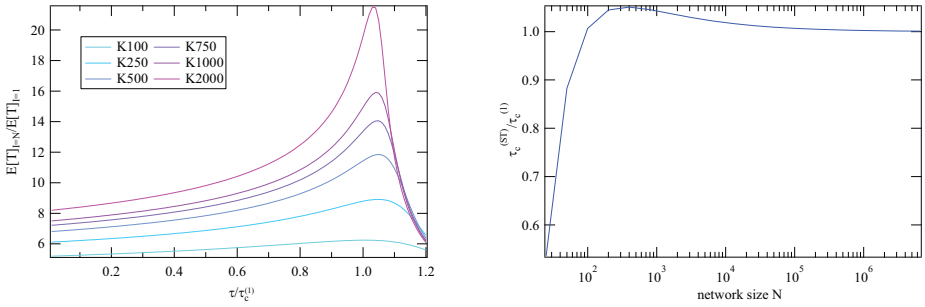
(b) Survival time of the SIS process starting from a single infected node in a star graph as a function of the normalised effective infection rate for various network sizes. For each size two curves are shown: a solid one indicating the process started at a centre node, and a dashed one indicating the process started at a leaf node.

Figure 3.7

Figure 3.7b shows the expected survival time starting from a single infected node in the star graph as a function of the normalised effective infection rate $\tau/\tau_c^{(1)}$. Obviously, there is a difference in the star graph between starting an epidemic at the centre node or at one of the leaf nodes. The difference in survival time between these two starting positions decreases with an increasing infection rate. In contrast to the complete graph, in the star graph the infection rate has to be a few times larger than ${\lambda_1}^{-1}$ to stay in the network for a long time. In the next section it is determined, via the hitting time, precisely how much larger than ${\lambda_1}^{-1}$ the epidemic threshold is. After passing the epidemic threshold, the increase in survival time is not as explosive as in the complete graph.

3.3.3. EPIDEMIC THRESHOLD VIA SURVIVAL TIME

Although the result in (3.13) gives the average hitting time of the absorbing state in K_N starting from the all-infected state, the hitting time equations (3.6) can be solved numerically for any initial state. Using the numerical solution to the system in (3.6) we can compare the survival time of a virus starting with a single node infected to the survival time of virus starting with all nodes infected. The ratio of the survival time starting with a single infected node and starting in the all-infected state, $E[T]_{I=N}/E[T]_{I=1}$, is relatively small. Figure 3.8a shows $E[T]_{I=N}/E[T]_{I=1}$ as a function of the normalised effective infection rate $\tau/\tau_c^{(1)}$ for various network sizes. The maximum ratio depends on the network size, and is generally largest just after the NIMFA threshold $\tau_c^{(1)}$ and slowly seems to move towards $\tau_c^{(1)}$ with increasing N. Below the epidemic threshold, the survival times are approximately the same because the virus dies out quickly. Above the epidemic threshold, the ratio rapidly becomes smaller, indicating that the initial number of infected nodes becomes less relevant with an increasing effective infection rate $\tau > \tau_c^{(1)}$. The peak in

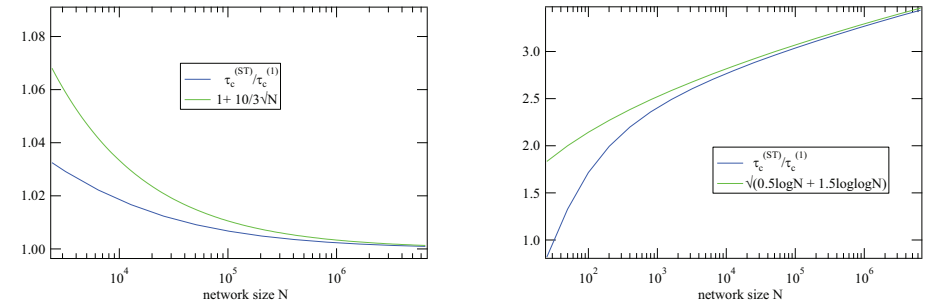


(a) (b)

Figure 3.8: The ratio between the average survival time of the SIS process in the complete graph starting with one node infected and with all nodes infected $\frac{E[T]_{I=N}}{E[T]_{I=1}}$ as a function of $\tau/\tau_c^{(1)}$ for different network sizes (a). The ratio of $\tau_c^{(ST)}/\tau_c^{(1)}$ as a function of the network size (b).

this ratio might be an accurate location of the precise epidemic threshold, as it coincides with the transition point of the survival time as a function of $\tau/\tau_c^{(1)}$ in K_N as shown in Figure 3.5. Time-based properties of the SIS process, instead of the fraction of infected nodes, to identify the epidemic threshold have recently attracted attention [56]. We define $\tau_c^{(ST)}$ (ST for survival time) as $\max\{E[T]_{I=N}/E[T]_{I=1}\}$ and plot the ratio $\tau_c^{(ST)}/\tau_c^{(1)}$ as a function of the network size N in Figure 3.8b. For large N, both the NIMFA epidemic threshold and the one found using the survival time are the same. For very small networks, however, the NIMFA threshold is larger than the one found using the survival time, which is suprising since the NIMFA threshold is a lower bound for τ_c. Either NIMFA is not a lower bound for τ_c for $N < 100$, or $\tau_c^{(ST)}$ is not the epidemic threshold.

Similar to the complete graph, $\tau_c^{(ST)}$ can also be determined numerically for the star graph. In Figure 3.9 the ratio between $\tau_c^{(1)}$ and $\tau_c^{(ST)}$ is compared to the ratio between of $\tau_c^{(1)}$ and the "real" epidemic threshold τ as found in [49]. In [49], the ratio $\tau_c/\tau_c^{(1)}$ is



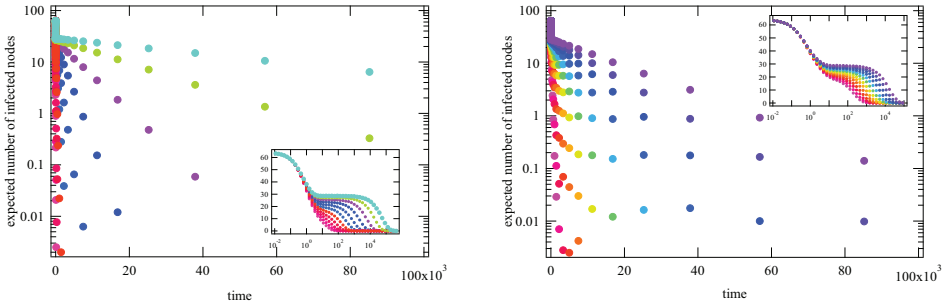
(a) (b)

Figure 3.9: The ratio of $\tau_c^{(ST)}/\tau_c^{(1)}$ compared to the scaling in [49] as a function of the network size for the complete graph (a) and the star graph (b).

given as $\sqrt{1/2\log N + 3/2\log\log N}$ for the star graph, and $1+\frac{c}{\sqrt{N}}$ with $c > \frac{10}{3}$ for the complete graph. Especially the correction factor for the star graph is very close to the ratio $\tau_c^{(ST)}/\tau_c^{(1)}$ for large N suggesting that $\tau_c^{(ST)}$ is indeed a good estimate of the epidemic threshold for large N. In the case of the complete graph the difference between the epidemic threshold found using the survival time and the corrected NIMFA threshold is also shrinking with increasing N.

3.4. THE AVERAGE SURVIVAL TIME IN OTHER GRAPH TYPES

In this section, we use simulations and numerical methods to investigate the extinction time in general graphs and the survival time distribution in the complete graph. The simulation results are obtained as follows. Starting from the initial state with all nodes infected, we simulate the SIS process and sample the number of infected nodes. The sample points are placed at time 1.5^k for $-20 \le k \le 32$ (3×10^{-4} to 4.5×10^5) and samples are taken regardless of the viral state of the network. If the virus dies out before the end of the simulation, all subsequent samples for that particular run will be 0. The results shown in this section are averaged over 10,000 runs. Figure 3.10 shows that the



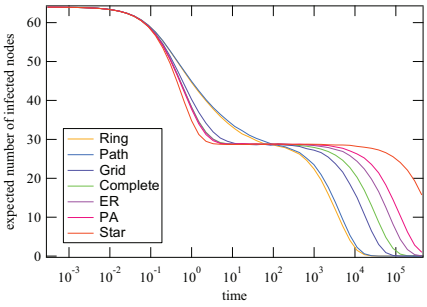
(a) An Erdős-Rényi graph of 64 nodes for beta ranging from 0.13 to 0.23.

(b) A square lattice of 64 nodes for beta ranging from 0.55 to 0.67.

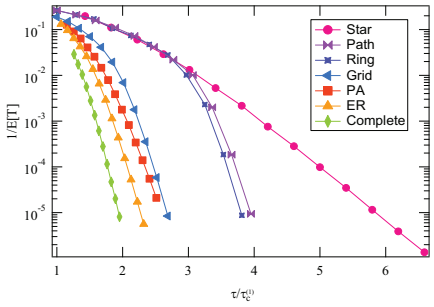
Figure 3.10: The expected number of infected nodes as a function of time for two graphs of 64 nodes for various values of the infection rate β. The curing rate $\delta = 1$ for all simulations. The insets in the plots show the expected number of infected nodes on a logarithmic time axis, highlighting the plateau-like meta-stable state of the SIS process.

survival time in the Erdős-Rényi random graph and the square lattice is exponentially distributed, as is to be expected since the SIS process in any graph is fully described by a Markov chain. After the initial quick drop in the number of infected nodes, the decay of the number of infected nodes are straight lines on the lin-log plot, indicating the exponential decay. The insets in Figure 3.10 show the expected number of infected nodes as a function of time on a logarithmic time axis. The compression of large time values makes the plateau-like meta-stable state visible. Both the Erdős-Rényi graph and the square lattice show similar behaviour.

Indeed, all graph types are expected to show similar meta-stable state behaviour, the only two differences between different graph types are the number of infected nodes in



(a) The expected number of infected nodes in various graph types for an effective infection rate τ that is chosen to have 45% nodes infected in the meta-stable state.



(b) The exponents of the tail of the survival time distribution for various graphs of size $N = 64$ as a function of $\tau/\tau_c^{(1)}$ where $\tau_c^{(1)} = \lambda_1^{-1}$ is the epidemic threshold in the NIMFA approximation.

Figure 3.11: Expected survival time in various graphs.

the meta-stable state as a function of the effective infection rate τ and the duration of the meta-stable state: the survival time.

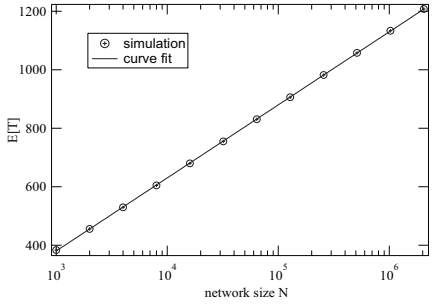
We first show the difference in the expected survival time for a fixed fraction of infected nodes in the meta-stable state y_∞ for various different graphs: the ring graph, the path graph, the square lattice (grid), the complete graph, the connected Erdős-Rényi random graph with link probability $2\log(N)/N$, a grown preferential attachment graph with $m = 4$ new links per node, and a star graph. All graphs have 64 nodes and the effective infection rate τ is chosen in such a way that the average fraction of infected nodes in the meta-stable state is the same for all graphs: 45%. Figure 3.11a shows that the ring and path graphs are the two graph types that can sustain an outbreak the shortest, whereas the star graph can sustain an outbreak the longest. A possible explanation for the fact that the virus stays alive in the star graph the longest is that, when scaled for the link density, the effective infection rate to achieve 45% infected nodes in the meta-stable state is the smallest for the star graph.

All simulations start with all nodes infected, so the number of infected nodes first drops to the level of the meta-stable state. Figure 3.11a illustrates that the longer the process stays in the meta-stable state, the quicker it reaches the meta-stable state from the all-infected state. However, the differences are not nearly as large as the difference in the survival time for most graphs, with the exception of the ring and path graphs. The time to reach the meta-stable state is explored in detail in Section 3.5.

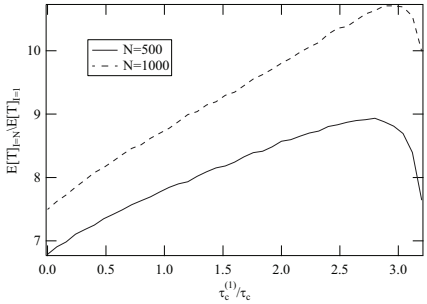
Figure 3.11b shows the decay rate $1/E[T]$ for seven different graph types as a function of the normalised effective infection rate $\tau/\tau_c^{(1)}$ Clearly, the complete graph has the steepest decline in decay rate when the effective infection rate increases above the epidemic threshold. In the star graph, the SIS process needs a much higher normalised effective infection rate to survive as long as in the complete graph. The path and ring graphs follow the star graph for smaller values of the effective infection rate but then

branch off for larger values of the infection rate. The branching suggests that the epidemic threshold lies around that point for the ring and the path graph, and that those two graph types can sustain a virus longer for an effective infection rate that is above the epidemic threshold than the star graph. Figure 3.11b also suggests that, as the star and the complete graph are two extreme graphs, the average survival time $E[T]$ as a function of the normalised effective infection rate $\tau/\tau_c^{(1)}$ in all other graphs is expected to lie between these two graphs. Combining the information from Figs. 3.11a and 3.11b leads to the conclusion that the star graph is the graph where a virus stays active the *longest* when the meta-stable state fraction of infected nodes is kept at 45% but the *shortest* when the normalised effective infection rate $\tau/\tau_c^{(1)}$ is kept constant. In part, this is caused by the fact that the NIMFA threshold for the star graph is not very accurate.

The ring graph is an extreme graph, just as the complete and the star graph. In Figure 3.12a the expected survival time in the ring graph is shown as a function of the network size for an effective infection rate $\tau = 3\tau_c^{(1)} = 1.5$. The survival time in the ring increases logarithmically with the number of nodes in the graph. The non-exponential scaling of the survival time with the network size N is surprising. Mountford *et al.* [57] have shown that the survival time of a contact process on a tree with bounded degree is exponential in the number of nodes, which implies that the survival time on a path graph is exponential in N. As the ring graph is a path graph with one extra link, the survival time for a given effective infection rate τ is expected to be *at least* equal to that of the path graph. In Figure 3.12b we use the survival time to determine the epidemic threshold in the ring graph of $N = 500$ and $N = 1000$ nodes. For $N = 500$ the epidemic threshold lies at approximately $2.8\tau_c^{(1)}$, but for $N = 1000$ it lies at approximately $3\tau_c^{(1)}$. This shows that the epidemic threshold indeed scales with N and that for much of the range of Figure 3.12a the infection rate is below the epidemic threshold, explaining the logarithmic scaling. Unfortunately, computing the epidemic threshold based on simulations for larger graphs is computationally expensive.



(a) Survival time in the ring graph as a function of the size for an effective infection rate $\tau = 3\tau_c^{(1)}$. The curve fit is of a logarithm $(249.82\log(N) - 369.17)$ and fits perfectly.



(b) The ratio between the average survival time of the SIS process in the ring graph starting with one node infected and with all nodes infected $\frac{E[T]_{I=N}}{E[T]_{I=1}}$ as a function of $\tau/\tau_c^{(1)}$ for different network sizes.

Figure 3.12

3.5. TIME TO THE META-STABLE STATE

In addition to the extinction or survival time of the virus outbreak, it is important to know the time until an outbreak reaches the meta-stable state. The average time to first reach a number of infected nodes equal to the average number of infected nodes in the meta-stable state is called the *spreading time*. Of course, the spreading time depends on the initial condition, as will become clear in the following sections. The spreading time of a virus in a particular graph is an important metric, since it limits the time in which the virus can still be contained with relative ease. In the extreme case of only a single infected individual, quarantining is easy and highly effective. Once the virus reaches the meta-stable state ($\tau > \tau_c$), it will be more difficult to eradicate the infection. Hence, the spreading time is generally the longest time for an authority (government, agency, etc.) to react and to start immunisation actions (such as vaccination and quarantining).

In this section, we numerically compute the spreading time of the SIS process on the complete graph and the star graph and investigate the scaling behaviour both in terms of the graph size N and effective infection rate τ. For the complete graph we derive an analytical expression for the average time to the meta-stable state starting from a single node. For other graph types, we use simulations to derive the spreading time.

3.5.1. SPREADING TIME VIA THE HITTING TIME

In general, the spreading time in a particular graph can be found using the Markov description of the SIS process. Let $\bar{y}$ be the average number of infected nodes in the meta-stable state, rounded to the nearest integer, i.e. $\bar{y} = \lfloor N y_\infty(\tau) \rceil$. The spreading time can be defined as the average hitting time of the set of states for which the total number of infected nodes equals $\bar{y}$, starting from any state. Unfortunately, this method is infeasible for a general graph due to the exploding state space. As we have seen in the previous sections, however, for the complete graph and the star graph, however, the state space scales linearly in the number of nodes, which enables us to use the hitting time to determine the spreading time.

Just as in Section 3.3, we use an embedded Markov chain to transform the continuous-time SIS Markov process to a discrete-time one. Before proceeding, however, we modify the SIS Markov chain and remove the absorbing state. As explained in [26, 49], the meta-stable state for finite graphs can be defined in two ways: either we add a nodal self-infection [26] or we remove the absorbing state [49]. The latter modified SIS model, denoted by MSIS, obeys the same evolution rules as the SIS model, except that when there is only one infected node in the network, this node is forbidden to heal. In both cases, the Markov chain is irreducible and features a unique steady-state, that corresponds (by definition) to the meta-stable state in the original SIS Markov process. Also, by our modification, the virus/infection will always stay in the network and can never die out. Thus, as the infection is prevented to die out, our modification will return an upper bound to the exact SIS spreading time in a graph.

The average hitting time of the set of states $\mathscr{A}$ starting from any state in the Markov chain can be found as the minimal non-negative solution of the following system [53, 54]:

$$\begin{cases} w_i = 0 & \text{for } i \in \mathscr{A} \\ w_i = 1 + \sum s_{ij}(\phi) w_j & \text{for } i \notin \mathscr{A} \end{cases}, \tag{3.15}$$

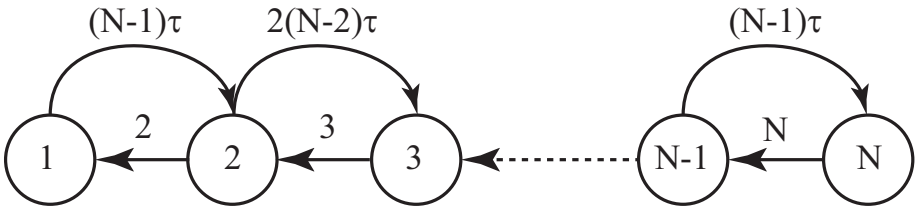


Figure 3.13: Markov graph of the modified SIS process on a complete graph. The state numbers coincide with the number of infected nodes.

where w_i is the average hitting time of the set $\mathscr{A}$ starting from state i, and $s_{ij}(\phi)$ is the transition probability from state i to j in the uniformised embedded Markov chain. Analogous to the analysis in Section 3.3, system (3.15) can be written as $Kw = b$, where $K = I - \widetilde{S}(\phi)$, and $\widetilde{S}(\phi)$ is the transition matrix of the uniformised embedded Markov chain with each row $i \in \mathscr{A}$ replaced with the standard basis vector e_i multiplied by some scalar. Because of the structure of b, the scalar can take any value, but for numerical reasons its convenient to take ϕ. The vector b is defined as $b = \boldsymbol{u} - \sum_{i\in\mathscr{A}} e_i$ where $\boldsymbol{u}$ is the all-one vector. Again, as in Section 3.3, we solve the system

$$\phi K w = b, \tag{3.16}$$

where ϕK simplifies to $\phi K = \phi(I - I + \frac{\widetilde{Q}}{\phi}) = -\widetilde{Q}$.

3.5.2. SPREADING TIME IN K_N

In this section, we investigate the spreading time in the complete graph K_N with N nodes. The meta-stable state is specified via the equilibrium point in K_N, which is defined as the number I of infected nodes for which $\beta(N-I)I = \delta I$ holds. At the equilibrium point in K_N, and generally in the meta-stable state for any graph, the total infection and curing rate are in balance.

The time to the equilibrium point can be found as the hitting time of the state where $\beta I(N-I) = \delta I$ holds, starting from any state. In the case of a real virus outbreak, the spreading time is indicative of how quickly measures to counter the infection need to be taken. Figure 3.13 shows the Markov graph of the modified SIS process on the complete graph. Since the state numbers coincide with the number of infected nodes, the spreading time or time to equilibrium is found by solving system (3.6) for $\mathscr{A} = \{\bar{y}\}$. The modified infinitesimal generator $\widetilde{Q}$ can be derived by inspection from Figure 3.13. Figure 3.14 shows the average spreading time as a function of the fraction of infected nodes in the complete graph for various network sizes N. The equilibrium point was chosen at 15%, 25%, 50% and 75% of infected nodes for all sizes. The average spreading time drops sharply from a single node infected to a few infected. The sharp initial drop is caused by the fact that, when a single node is infected, the probability that the virus dies out is the highest. However, in the MSIS model, dying out is forbidden, so that the time to reach the meta-stable state or equilibrium point is long. As soon as a few more nodes are infected, the infection is very likely to reach the equilibrium point. Usually, infectious diseases start at a few individuals/nodes and Figure 3.14 illustrates that *it is crucial to de-*

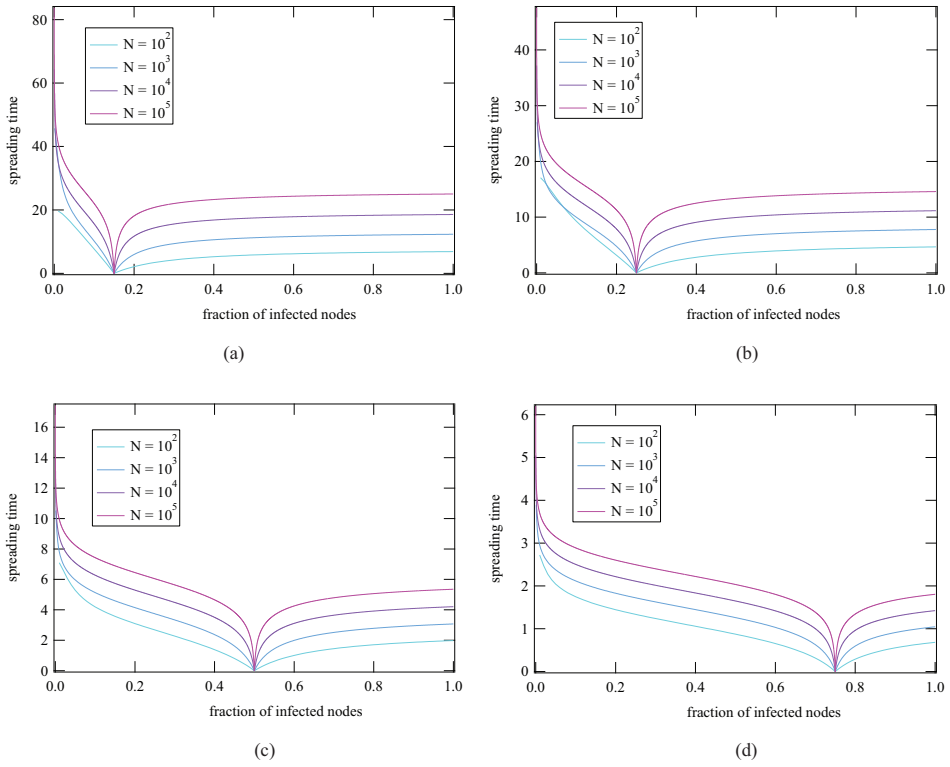


Figure 3.14: Spreading time in the complete graph as a function of the fraction of infected nodes for four values of the equilibrium point: 0.15 (a), 0.25 (b), 0.5 (c), and 0.75 (d).

tect an outbreak in a very early state, because only at a very early state (with few infected) the average spreading time is significantly larger than for other states. When initially far more nodes are infected than in the meta-stable state, the time to the equilibrium point is almost constant.

Figure 3.15 shows the average spreading time starting from a single infected node as a function of the network size for various positions of the equilibrium point. The average spreading time scales logarithmically with the network size for K_N. The logarithmic scaling in the number of nodes is especially interesting as the scaling of the infection rate to keep the equilibrium point, for example, at 50% infected is linear, as follows from $\tau(N-I)I = I$. Which can be written as $(\tau N-1)I-\tau I^2 = 0$, and solving for I gives $I = 0$ and $I = N-\frac{1}{\tau}$. Despite the fact that the infection rate decays with the reciprocal of the network size, the time to reach the equilibrium increases only logarithmically. This means that in a larger network a weaker virus can infect the same fraction of nodes in roughly the same time, again emphasizing the importance to spot an outbreak early.

Figure 3.15 shows that the spread time in the complete graph scales as $\log(N^\alpha)+b$, Figure 3.16 shows the fits of α in Figure 3.15 as a function of the equilibrium point y_∞. With an increasing equilibrium point, α reduces quickly, indicating that a stronger virus

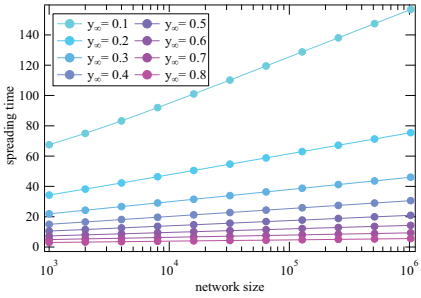


Figure 3.15: Spreading time to the equilibrium point starting from a single infected node in the complete graph as a function of the network size N for various values of y_∞.

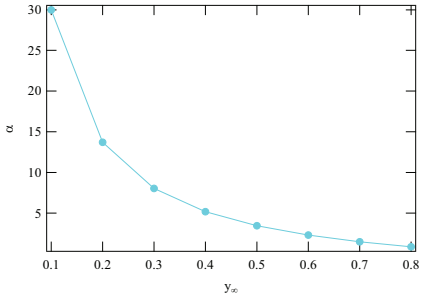


Figure 3.16: Slope of the fit of the spreading time in Figure 3.15 as a function of the equilibrium point y_∞.

does not only infect more individuals, but also does so quicker than a weaker virus. Fig-

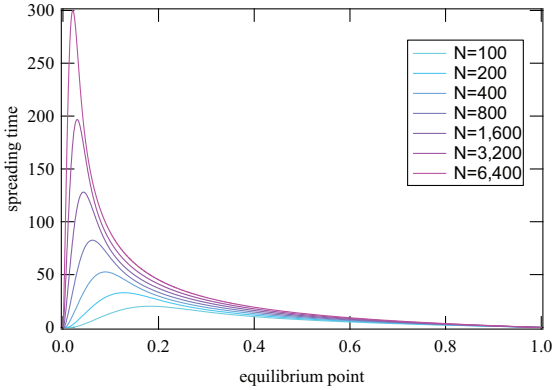


Figure 3.17: Spreading time in a complete graph starting from a single infected node as a function of the equilibrium point for various graph sizes.

ure 3.17 shows the average spreading time starting from a single infected node as a function of the equilibrium point for various network sizes. We observe from Figure 3.17 that for small values of the equilibrium point, the time to reach that equilibrium point starting from 1 node infected increases. This is caused by the smaller values of the effective infection rate τ needed to reach the equilibrium point. Interestingly, the time to reach the equilibrium point peaks due to the probability that the virus dies out when it starts with only a single infected node and a low effective infection rate τ. Because the absorbing state is removed, the process spends more time in the state with only one node infected for low effective infection rates, which increases the time to reach the equilibrium point. With an increasing number of infected nodes at the equilibrium point, the effective infection rate τ increases until the infection rate is so large that dying out becomes

unlikely. This illustrates that an increasing effective infection rate does not only increase the number of infected nodes in the meta-stable state but also increases the probability that the meta-stable state will be reached from a single infection, and reduces the time to reach the meta-stable state.

3.5.3. ANALYTICAL EXPRESSION SPREADING TIME IN K_N

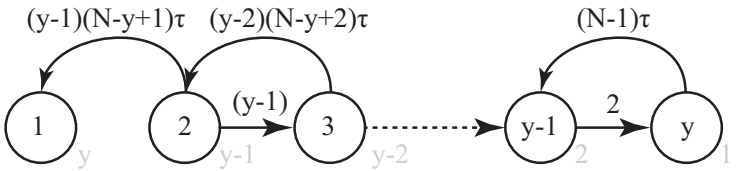


Figure 3.18: Markov graph of the modified SIS process on a complete graph. The number of infected nodes is indicated next to the state.

By modifying the Markov chain in Figure 3.13 and solving matrix equation (3.16) for the modified chain we derive an exact expression for the average spreading time in K_N starting from a single infected node. The modified Markov chain is shown in Figure 3.18, where we order the states from the meta-stable state back to a single infected node. Matrix equation (3.16) can be solved by reducing the augmented matrix $[-\widetilde{Q}|b]$ to row echelon form. To reach row echelon form, we iteratively perform the following row operation for $k \geq 2$.

$$r_{k+1} \rightarrow r_{k+1} + \frac{(y-k)(N-y+k)\tau}{d_k} r_k$$

where r_k indicates row k and d_y is given recursively by

$$d_k \begin{cases} (y-1)+(y-1)(N-y+1)\tau & \text{for } k=2 \\ (y-k+1)[1+(N-y+k-1)\tau - \frac{(y-k+2)(N-y+k-1)\tau}{d_{k-1}}] & \text{for } 2<k<y \\ (N-1)\tau - \frac{2(N-1)\tau}{d_{k-1}} & \text{for } k=y \end{cases}$$

The elements in column vector b can be found recursively as

$$b_k = 1 + \frac{(y-k+1)(N-y+k-1)\tau}{d_{k-1}} b_{k-1}$$

The average spreading time is given by b_y/d_y. Iterating the recursive relation for b_k from $k=y$ backwards leads to

$$b_y = \sum_{j=1}^{y-1} \frac{(j-1)!(N-1)!\tau^{(j-1)} \prod_{i=2}^{i=y-j} d_i}{(N-j)! \prod_{i=1}^{i=y-1} d_i},$$

and dividing by d_y gives

$$E[T] = \sum_{j=1}^{y-1} \frac{(j-1)!(N-1)!\tau^{(j-1)} \prod_{i=2}^{i=y-j} d_i}{(N-j)! \prod_{i=1}^{i=y} d_i} \tag{3.17}$$

Writing $p_j = \prod_{i=2}^{j}$ and using $p_j = d_j p_{j-1}$, we can write

$$p_j = (y-j+1)p_{j-1} + (y-j+1)(N-y+j-1)\tau p_{j-1} \\ -(y-j+2)(y-j+1)(N-y+j-1)\tau p_{j-2}$$

for $2 < j < y$. Iterating the recursive relation for p_j from j backwards leads to

$$p_j = \frac{(y-1)!}{(y-j)!(N-y)!} \sum_{i=0}^{j-1} (N-y+i)!\tau^i$$

and using the correct expression for d_y,

$$p_y = \frac{(y-1)!(N-1)!\tau^{y-1}}{(N-y)!}$$

Filling in the expressions for p_j and p_y into (3.17) leads to our final result

$$E[T] = \sum_{j=1}^{y-1} \sum_{i=0}^{y-j-1} \frac{(N-y+i)!\tau^{i+j-y}}{j(N-j)!}$$

3.5.4. MEAN-FIELD SPREADING TIME IN REGULAR GRAPHS

We briefly move from the exact world of Markov chains to the approximate world of mean-field theory and show that mean-field predicts a spreading time in r-regular graphs that is independent of the degree r. In the N-intertwined Mean-Field Approximation (NIMFA) [29, 31] the probability v_i that node i is infected is given by the following first-order nonlinear ordinary differential equation

$$\frac{dv}{dt} = r\beta v(t)(1-v(t)) - \delta v(t), \tag{3.18}$$

which also appeared in [15, 18], and has the solution (also derived in [19]):

$$v(t) = \frac{1}{(\frac{1}{v_0} - \frac{1}{1-\frac{1}{r\tau}})\exp(-(r\tau-1)\frac{t}{\delta}) + \frac{1}{1-\frac{1}{r\tau}}} \tag{3.19}$$

According to NIMFA, the fraction of infected nodes in the meta-stable state for an r-regular graph is given by

$$y_\infty = 1 - \frac{1}{r\tau},$$

alternatively, the effective infection rate τ that is needed to achieve y_∞ nodes infected in the meta-stable state is given by

$$\tau = \frac{1}{r(1-y_\infty)}$$

Substituting τ in (3.19) and starting from a single infected node ($v_0 = \frac{1}{N}$) leads to:

$$v(t) = \frac{1}{(N - \frac{1}{y_\infty})\exp(-\frac{y_\infty}{1-y_\infty}\frac{t}{\delta}) + \frac{1}{y_\infty}}$$

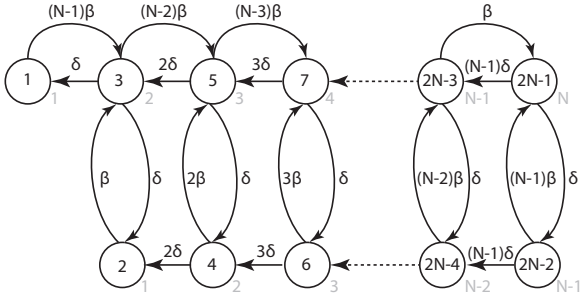


Figure 3.19: Markov graph of the modified SIS process on a star graph. In the top row (odd states) the centre node is infected, whereas on the bottom row (even states) the centre is healthy. The number of infected nodes in each state is indicated next to the state.

which suggests that the time to reach the meta-stable state is independent of r. The spreading time according to NIMFA is found by defining a distance between $v(t)$ and y_∞ and finding the time for which the distance is small. Starting from

$$v(t) = \frac{1}{f(t) + \frac{1}{y_\infty}}$$

with $f(t) = (N - \frac{1}{y_\infty}\exp(-\frac{y_\infty}{1-y_\infty}\frac{t}{\delta}))$, we define the distance to meta-stable state as $y_\infty - \frac{1}{f(t)+\frac{1}{y_\infty}}$, and the spreading time as the time for which the distance to the meta-stable state is smaller than ε:

$$f(t) \leq \frac{\varepsilon}{y_\infty(y_\infty - \varepsilon)}$$
$$t \geq \frac{(1-y)\delta}{y}(\ln(N - \frac{1}{y_\infty}) + \ln(y^2 - y\varepsilon) - \ln(\varepsilon))$$

Although the NIMFA spreading time is not always accurate, and in some cases not even predicts the right scaling of the spreading time, as shown in Section 3.7, at least for the complete graph it correctly shows a logarithmic scaling in N.

3.6. SPREADING TIME IN $K_{1,N-1}$

In this section, we investigate the spreading time in the star graph $K_{1,N-1}$ with N nodes. The Markov graph of the modified SIS process on a star graph, as adapted from [49] is shown in Figure 3.19. The Markov chain consists of $2N-1$ states, to make a distinction between the situation where the centre node is infected (all odd states), and the situation where the centre node is healthy. The number of infected nodes in each state is indicated in blue next to the state in Figure 3.19. For each possible number of infected nodes (with the exception of N infected nodes), the network can be in one of two states. As a result, the average spreading time can be computed by solving system (3.6) with $\mathscr{A} = \{2\bar{y} - 1, 2\bar{y}\}$, for $\bar{y} < N$, or $\mathscr{A} = \{2N-1\}$ for $\bar{y} = N$.

Figure 3.20 shows the average spreading time in the star graph for various network sizes and values of $\bar{y}$, under the condition that the centre node is infected. The spreading

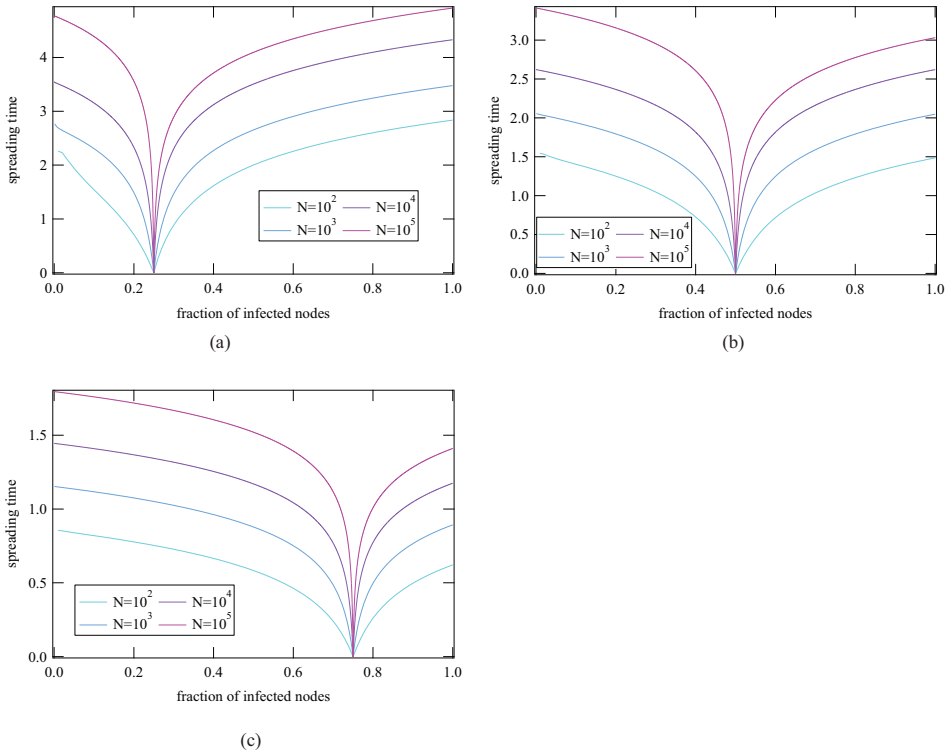


Figure 3.20: Spreading time in the star graph for the case that the centre is infected as a function of the fraction of infected nodes for three different values of $\bar{y}$: 0.25 (a), 0.5 (b), and 0.75 (c).

time in the star graph is smaller than for the complete graph. This is caused by the higher effective infection rate τ that is needed to reach the same $\bar{y}$ in the star graph compared to the complete graph. Paradoxically, the better we protect our networks, the quicker a virus will reach the meta-stable state. Of course, for a virus with a fixed effective infection rate τ protecting the network by, for example, link removal will have an effect on the number of infected nodes in the meta-stable state. However, in the context of computer viruses or other engineered infectious processes, better network protection will have to lead to stronger viruses to achieve the same goal (infecting nodes). A similar effect might occur in biological evolutionary processes. The average spreading time in Figure 3.20 is under the condition that the centre node is infected. An infection starting in the centre node has a better chance of spreading through the network than an infection starting in one of the leaf nodes. Yet, the average spreading time for a virus starting in a leaf node does not differ too much from that of a virus starting in the centre, as shown in Figure 3.21. Figure 3.21 shows the average spreading time as a function of the number of infected nodes for three different network sizes, both under the assumption that the centre node is infected (round markers) and under the assumption that the centre node is healthy (triangular markers). In terms of the modified Markov process in Figure 3.19:

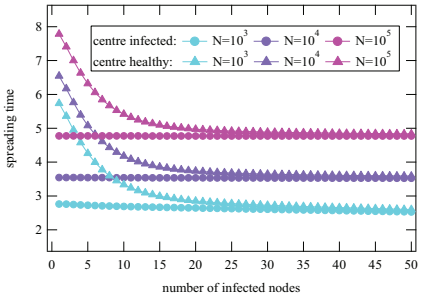


Figure 3.21: Spreading time in the star graph as a function of the number of infected nodes for three different network sizes, both for the case that the centre is infected (round markers) and for the case that the centre is healthy (triangular markers).

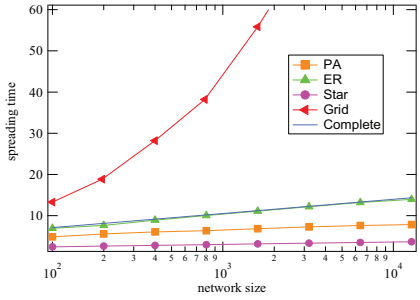


Figure 3.22: Spreading time starting from a single infected node as a function of the network size for five different graph types. The fraction of infected nodes in the meta-stable state is 50%
.

the round markers indicate the spreading time starting in an odd state, the triangular markers indicate the spreading time starting from an even state. For all three network sizes, the difference between starting in an odd state (centre infected) or an even state is roughly the same and diminishes quickly. If more than 20 nodes are infected, the spreading time under the assumption that the centre is infected is approximately the same as under the assumption that it is healthy.

3.7. SPREADING TIME IN OTHER GRAPHS

The average spreading time in a general graph cannot be determined using the Markov transition probability matrix as for the complete graph K_N or the star, due to the 2^N state space in general. In the case of a general graph, we simulate the time until the process first reaches the number of infected nodes of the meta-stable state. We first determine the meta-stable state as a function of the effective infection rate for each graph and select the effective infection rate τ that corresponds to 50% infected in the meta-stable state. The data points for each graph type and size are averages over 100,000 runs.

Figure 3.22 shows the average spreading time as a function of the network size N for 5 different graph types, the connected Erdős-Rényi random graph $G_p(N)$ with a link density or link probability p at the connection threshold $p_c = \frac{\ln N}{N}$, the preferential attachment graph (PA) with m links per new node (m is chosen so that the link density is equal to the ER graph), the star graph, the square lattice and the complete graph.

Curiously, the connected Erdős-Rényi random graph $G_p(N)$ behaves almost identical to the complete graph $K_N = G_1(N)$: the spreading time seems hardly to depend on the link density $p = \frac{2L}{N(N-1)}$. Probably the difference in link density is compensated for by the higher effective infection rate τ that is needed to reach the same $\bar{y}$.

The presence of hub nodes in the preferential attachment graph, however, leads to a smaller time to the meta-stable state, especially for larger graphs. The extreme case in terms of hub nodes is the star graph, and indeed in the star graph the time to reach the meta-stable state is shortest.

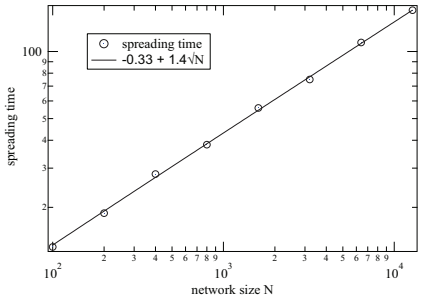


Figure 3.23: Spreading time starting from a single node infected in the rectangular grid. The fit shows that the spreading time scales as $O(\sqrt{N})$.

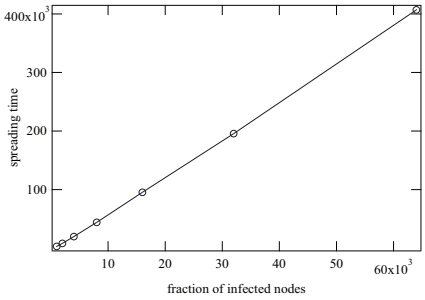


Figure 3.24: Spreading time starting from a single node infected in the ring graph.

These four graph types all show a logarithmic scaling of the average spreading time with the number of nodes, but the slope at the log-lin scale is smaller for graphs with more hub nodes. The square grid does not show the logarithmic scaling in N that is observed in the other graph types. Indeed, Figure 3.23 shows the spreading time for just the rectangular grid and a fit, suggesting that the spreading time scales as $\sqrt{N}$ with the size N. This is most likely caused by the large average hopcount in this graph type. The average hopcount in the lattice [53, p.630] equals $E[H_{\text{lattice}}] \simeq \frac{2}{3}\sqrt{N}$, for large N, whereas $E\left[H_{G_{p_c}(N)}\right] \sim \frac{\ln N}{\ln\ln N}$ (see [53, p. 428]) increases somewhat slower than logarithmically in N. Also for sparse scale-free graphs of which the Barabasi-Albert preferential attachment graph is an instance [53, p. 428], the average hopcount is $E[H_{\text{PA}}] \sim \frac{\ln N}{\ln(d_{av}-1)}$, where $d_{av} = \frac{2L}{N}$ is the average degree. Hence, for the lattice, the connected Erdős-Rényi random graph $G_{p_c}(N)$ and scale-free graphs, the scaling of the spreading time and the average hopcount agree. However, the relation between spreading time and hopcount ceases for the star and complete graph: the average hopcount in K_N is $E\left[H_{K_N}\right] = 1$ and in the star $K_{1,N-1}$ with N nodes is $E\left[H_{K_{1,N-1}}\right] = 2 - \frac{2}{N}$, thus both bounded by a constant, whereas logarithmic increases is observed in Figure 3.22. In any graph, the average time [58],[53, p. 460] to hit the absorbing state in SIS epidemics for $\tau < \tau_c$ scales logarithmically in N, illustrating that, on average, the epidemic process cannot tend faster to the meta-stable state than $O(\log N)$, irrespective of the topology. The influence of the network topology on the SIS dynamic process is measured via the average hopcount which is a good representative measure for the diffusive spread of the epidemic. Above the epidemic threshold ($\tau > \tau_c$), the epidemic diffuses over the network. The spreading time thus reflects the combined logarithmic scaling (as a lower bound for the average epidemic extinction time and, hence, of the entire SIS dynamic process) and the average diffusion time, which is related to the average hopcount of the graph. In conclusion, only when the average hopcount exceeds a logarithmic scaling, the spreading exhibits the hopcount scaling, else logarithmic scaling is observed. As a final example of a spreading time that sales with the hopcount, Figure 3.24 shows the linear scaling in N of the spread time on a ring graph.

3.8. CHAPTER SUMMARY

In this chapter, we shed light on the survival time T of an SIS process on a network. We have derived exact equations for the average survival time in the complete graph K_N and the star graph $K_{1,N}$ using the hitting time of a uniformised embedded Markov chain. For the SIS process on the complete graph K_N, the first term in the Lagrange series (3.1) of the second largest eigenvalue ζ of the infinitesimal generator of the continuous-time Markov chain describing the SIS process can be efficiently computed using a sum (3.3) over recursive terms and is equal to the expected wost-case survival time $E[T]$.

The embedded Markov chain enables us to compute, via the peak in $\frac{E[T]_{I=N}}{E[T]_{I=1}}$, the precise epidemic threshold that corrects for the scaling behaviour in the number of nodes N of the NIMFA threshold (which is the inverse of the spectral radius of G).

Via simulations, we have shown that other graphs than K_N and $K_{1,N}$ also exhibit exponentially distributed survival times. When the steady-state fraction y_∞ of infected nodes is kept constant at 45%, the star graph sustains the infection the longest, whereas the ring graph sustains the infection the shortest. As a function of the normalised effective infection rate $\tau/\tau_c^{(1)}$, however, the virus dies out the quickest in the star graph. This is caused by the accuracy of the NIMFA model that is used for the normalisation and illustrates the difficulty of finding an unbiased way of comparing the behaviour of the SIS process in different graph types. In addition to the survival time, we also investigate the time to reach the meta-stable state, which we call the spreading time. We derive an exact expression for the spreading time in a complete graph, and show that in the case of regular graphs, the NIMFA spreading time can be very far off. The spreading time seems to scale with the hop count as a function of the network size N.

4

COMPETITION BETWEEN SIS EPIDEMICS

When two viruses compete for healthy nodes in a simple network and both infection rates are above the epidemic threshold, only one virus will survive. However, if we prevent the viruses from dying out, new and rich dynamics emerge. When both viruses have identical infection and curing processes, one virus always dominates the other, but the dominating and dominated virus alternate. We show in the complete graph that the domination time depends on the total number of infected nodes at the beginning of the domination period and, moreover, that the distribution of the domination time decays exponentially yet slowly. When the viruses differ moderately in strength and/or speed the weaker/slower virus can still dominate the other, but for a short time. Interestingly, depending on the number of infected nodes at the start of a domination period, being quicker can be a disadvantage.

4.1. INTRODUCTION

The SIS model we have studied in the previous two chapters can be used to model the spread of diseases, rumours, opinions, and habits in populations or on online social networks, as well as the effect of marketing and product adoption [59–62].

Viruses, however, do not necessarily live in isolation; a fact that has attracted attention recently. Work on virus competition either focuses on single layer networks, or on multi-layer networks. Prakash et al. [63] proved that when two viruses compete on a single layer network, the stronger of the two will completely suppress the other, provided that the viruses are mutually immune. This is in line with the competitive exclusion principle found in ecology [64, 65]. When viruses are not mutually immune, a region can exist where both viruses survive in the network [66]. Even when nodes are mutually immune, there is a region where stronger and/or quicker viruses can coexist with weaker or slower competitors in the SIR model [67]. In the SIR model, a node does not return to the susceptible state after being infected, but to an inactive recovered/removed state.

The coexistence of SIR type epidemics is greatly influenced by node mobility in spatially structured populations [68].

In multi-layer networks, each virus propagates over a separate link set. Wei et al. [69] developed a predictor for the winner of the competition between viruses on multi-layer networks and offer mitigation strategies to slow one of the two viruses down. Recently, Sahneh and Scolgio [70] improved on the understanding of virus competition in a multilayer network and showed that when the degrees in the two layers are negatively correlated, coexistence of two viruses is in fact possible, contrary to the findings in [69]. Marceau et al. [71] studied the effect of asymmetric a partial immunity in SIR epidemics where the spread of knowledge that a virus exists in the network helps to combat the spreading of the virus. Meyers et al. [72] developed a model that describes the interaction between multiple viruses (or contagions) under the assumption that a virus can also improve the spreading properties of another virus and validated that model on messages propagating through the Twitter network.

A common feature in the study of virus spread is that viruses can become extinct, as treated in chapter 2 and 3. What happens, however, if one node will stick to its opinion or preference or (intentionally) stays infected? Such a model can be used to describe competing marketing campaigns where each campaign has a (small) fixed support base and explain ebbs and flows in the success of such campaigns. It can also describe the arrival of a new product/idea in a population and model its adoption starting from a fixed fraction of infected nodes. We will see that by disallowing a virus to die out, new and rich dynamics emerge. In line with previous research, one of the two viruses will be dominated by the other, but, contrary to what might be expected, this dominance will not last for ever. We restrict a virus from dying out by reinfecting the last node with a a particular virus after it cures, but other schemes such as en ε-SIS model with an added nodal mutation can be thought of.

4.2. MSIS PROCESS DESCRIPTION

We model the process of multiple competing viruses by extending the SIS model. As explained in Chapter 2.1, in the SIS model, a node is either healthy or infected. If a node is infected, it will spread the infection to each of its neighbours. After some time, the infected node becomes healthy again but will remain susceptible to the infection. The curing process per node is a Poisson with rate δ and the infection rate per link is a Poisson process with rate β. All processes are independent. The effective infection rate τ is defined as the ratio between the spreading and curing rate, i.e. $\tau = \frac{\beta}{\delta}$.

The extension from a single virus to multiple competing viruses is as follows. A node can still be either infected or healthy. The infectious state, however, is labelled with the virus type. If a node is infected with virus A, for example, it will spread the infection of type A to its healthy neighbours. The healthy state is not labelled, so that, irrespective of the virus type, nodes will only be infected with one virus at a time. In other words, infected nodes are mutually immune; an infected node cannot become infected by another virus.

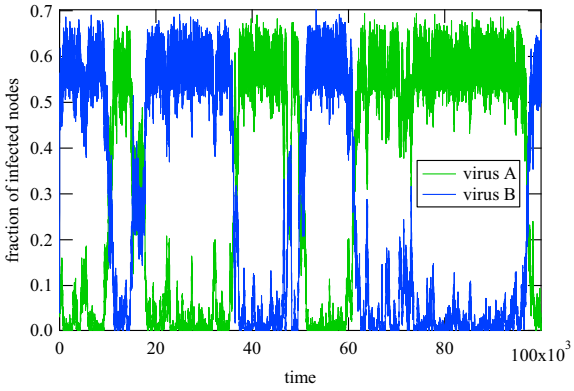


Figure 4.1: Competition between two viruses in a complete graph of 500 nodes. Each virus is prevented from dying out by reinfecting the last infected node of a virus at the moment it is cured.

4.3. MODELLING AND SIMULATION

We start by considering the case of two viruses in a complete graph of N nodes, K_N, and use Markov theory to completely describe the process. Although the complete graph is an extreme graph, simulations in Section 4.3.1 show similar behaviour in very different graphs. The symmetry of the complete graph allows us to use a Markov chain with a manageable number of states to describe the process, just as we saw in chapter 3.

We define N_A and N_B as the number of nodes infected with virus A and B respectively, and N_H as the number of healthy nodes. Clearly $N_A + N_B \leq N$ and $N_H = N - N_A - N_B$. A state in the Markov chain corresponds to a combination of N_A and N_B in the network. Every time the total number of infected nodes in the network changes, a state transition in the embedded Markov chain of the continuous-time Markov process occurs. The total number $\frac{(N+1)(N+2)}{2}$ of states in the Markov chain is visualised as the lower half of a rectangular grid, as illustrated in Figure 4.2. The lower left-hand corner corresponds to the all-zero state; the other states in the Cartesian grid have coordinates that encode the number of infected nodes per virus type. The horizontal axis reflects the change in the number of nodes infected with virus A, and the vertical axis reflects a change in the number of nodes infected with virus B. State changes in the Markov chain can be thought of as being either North (N_B increases), South (N_B decreases), East (N_A increases) or West (N_A decreases).

North($\mathcal{N}(N_A, N_B)$): This state transition occurs when a virus B spreading event happens before any other event. Spreading events happen with an exponentially distributed waiting time over all $N_B N_H$ links between nodes infected with virus B and healthy nodes. The first spreading event happens after the minimum of the $N_B N_H$ waiting times, which is again an exponential random variable with rate $\beta_B N_B N_H$. The first event that is not a virus B spreading event happens after the minimum of the virus A spreading events and the virus A and B curing events, of which there are $N_A N_H$, N_A and N_B respectively. The waiting time until the first non-virus-B spreading event is also an exponential random variable with rate $\beta_A N_A N_H + \delta_A N_A + \delta_B N_B$. The probability that the

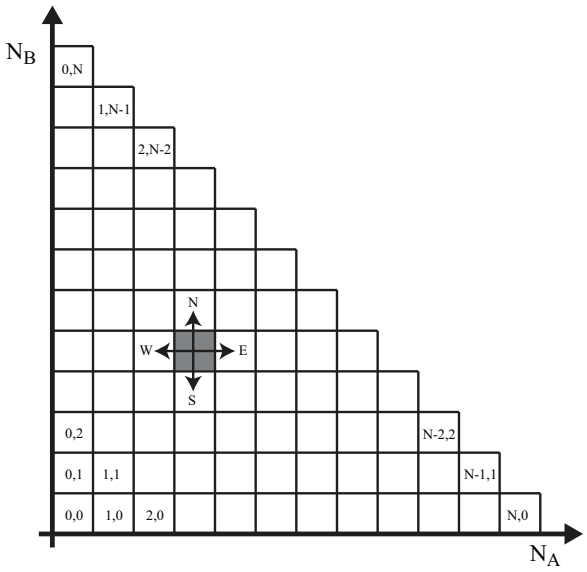


Figure 4.2: Markov chain representation of the number of infected nodes. From each state there are four different transitions possible: North, South, East and West. The transition probabilities of a direction can be zero at the borders of the "triangle".

first spreading event of virus B happens before the first non-spreading event is given by $\frac{\beta_B N_B N_H}{\beta_B N_B N_H + \beta_A N_A N_H + \delta_A N_A + \delta_B N_B}$.

South($\mathcal{S}(N_A, N_B)$): This state transition occurs when a virus B curing event happens before any other event. Analogous to the transition to North, this probability is given by $\frac{\delta_B N_B}{\beta_B N_B N_H + \beta_A N_A N_H + \delta_A N_A + \delta_B N_B}$.

East($\mathcal{E}(N_A, N_B)$): This state transition occurs when a virus A spreading event happens before any other event. The probability is the same as North, but with N_A and N_B interchanged: $\frac{\beta_A N_A N_H}{\beta_B N_B N_H + \beta_A N_A N_H + \delta_A N_A + \delta_B N_B}$.

West($\mathcal{W}(N_A, N_B)$): This state transition occurs when a virus A curing event happens before any other event. The probability is the same as South, but with N_A and N_B interchanged: $\frac{\delta_A N_A}{\beta_B N_B N_H + \beta_A N_A N_H + \delta_A N_A + \delta_B N_B}$.

From the transition probabilities for the four directions it is clear that if N_A (or N_B) is zero, only transitions along the North-South (East-West) axis are possible. As a result, all states in Figure 4.2 that are not along either the horizontal or vertical axis are transient states if we ignore the fact that the all zero state is an absorbing state. For an effective infection rate far enough above the epidemic threshold, the probability of reaching the absorbing all zero state is very small, but the probability of reaching either of the two axes of Figure 4.2 is not. This reaffirms earlier results that one virus will die out much quicker than the other.

By numbering the states appropriately, we can write the transition probability matrix

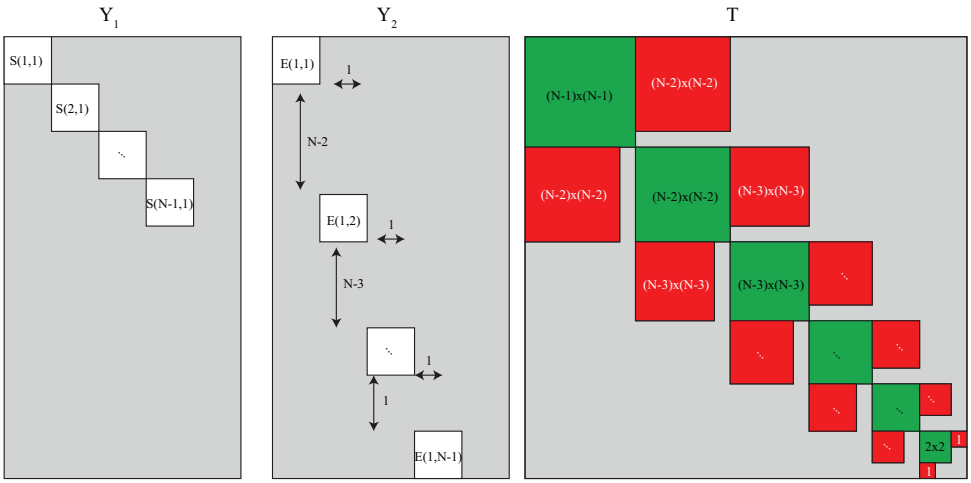


Figure 4.3: The structure of matrices Y_1, Y_2 and T. Grey areas consists of only zeros.

P as follows [73]:

$$P = \begin{bmatrix} C_1 & 0 & 0 \\ 0 & C_2 & 0 \\ Y_1 & Y_2 & T \end{bmatrix},$$

where C_1 is a matrix containing the transition probabilities of the closed set of states with only virus A active, C_2 is a matrix containing the transition probabilities of the closed set of states with only virus B active, Y_1 and Y_2 are the matrices containing the transition probabilities from transient states to either of the two closed sets, and finally T is a $\frac{N(N-1)}{2} \times \frac{N(N-1)}{2}$ substochastic matrix containing the transition probabilities among transients states.

The structure of the submatrices of P is further detailed in Figure 4.3. Matrix Y_1 is rectangular with only non-zero elements on the diagonal, with the exception of the last diagonal element. Matrix Y_2 is also rectangular but the vertical spacing is $N-1$ between the first non-zero elements and decreases for every next element. Matrix T consists of successively shrinking diagonal and off-diagonal blocks. The diagonal blocks of T are indicated in green in Figure 4.3 and have the following internal structure:

$$G = \begin{bmatrix} 0 & \mathscr{E}(1,B) & & & 0 \\ \mathscr{W}(2,B) & 0 & \mathscr{E}(2,B) & & \\ & \mathscr{W}(3,B) & 0 & \ddots & \\ & & \ddots & 0 & \mathscr{E}(N-B-1,B) \\ 0 & & & \mathscr{W}(N-B,B) & 0 \end{bmatrix}$$

where B indicates the block number (starting from 1 for the top left corner in T in Figure 4.3). The off diagonal matrices in T are diagonal matrices:

$$R_L = diag\{\mathscr{S}(1,B+1), \mathscr{S}(2,B+1), \ldots, \mathscr{S}(N-B-1,B+1)\}$$

$$R_R = diag\{\mathscr{N}(1,B), \mathscr{N}(2,B), \ldots, \mathscr{N}(N-B-1,B)\}$$

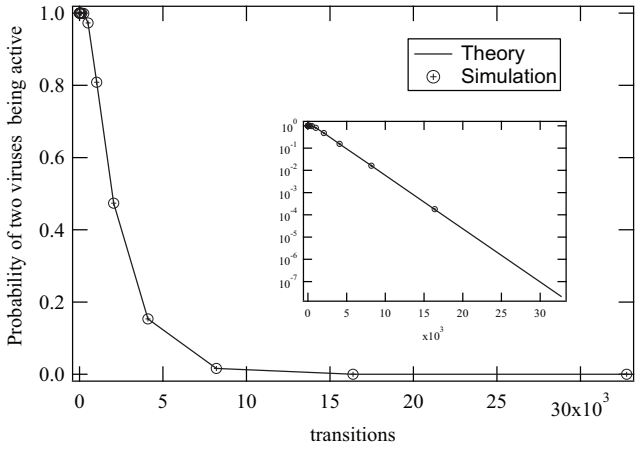


Figure 4.4: The probability that both viruses are alive as a function of the number of transitions. The solid line indicates the numerical results obtained from the transition matrix and the markers indicate the simulation results. The probability that both viruses are alive is the same as the probability that the epidemic process is in a transient state, i.e. the element wise sum of the entries of sub-matrix T. The inset show the same plot on a logarithmic vertical axis and emphasises the exponential decay of the transient state.

where R_R is the right hand side off-diagonal matrix, R_L is the left hand side off-diagonal matrix.

The m-step transition probability from state i to j, that is, the probability of being in state i after m steps when starting in state j, is given by $P^m_{i,j}$. Because of the block structure of P, its powers can be expressed as

$$P^m = \begin{bmatrix} C_1^m & 0 & 0 \\ 0 & C_2^m & 0 \\ X_1 & X_2 & T^m \end{bmatrix},$$

where X_1 and X_2 are the m step transition probabilities from transient states to either of the two closed sets. Since submatrix T represents the transient states, T^m decreases with increasing m. When all elements in T^m are zero, one of the two viruses has died out. Figure 4.4 shows that in a complete graph of 50 nodes the probability of both viruses being active decays exponentially after an initial period. The same results were obtained via simulations, as also indicated in Figure 4.4. The simulation results are averaged over 10^7 runs. In the initial state, 20 nodes are infected with each virus. The simulation results overlap exactly with the predicted curve.

4.3.1. PERPETUAL COMPETITION

When both viruses are prevented from dying out, the two viruses will be in everlasting competition. In the Markov chain this can be achieved by turning the South (East) transition for all states with just one node infected with virus B (A) into the probability to stay in the same state. In simulations we reinfect a node if curing it would lead to one of the

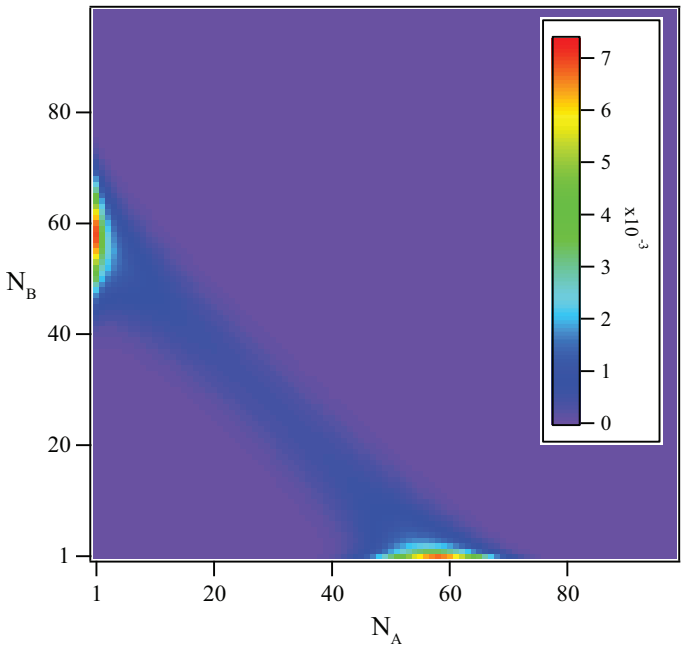


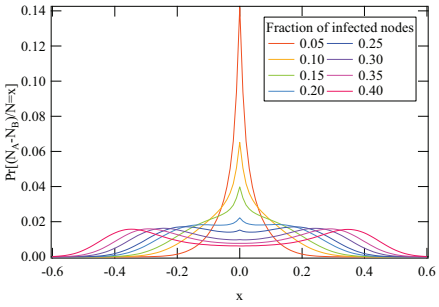
Figure 4.5: Heatmap representation of the steady-state distribution of two equally strong viruses in a complete graph of 100 nodes, and effective infection rate $\tau = 0.025$.

two viruses dying out.

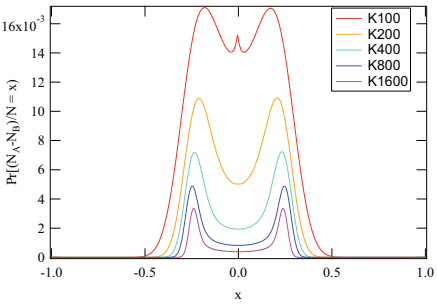
In the simulations of the continuous-time virus competition process, the number of infected nodes with virus A and B is logged every time the total number of infected nodes changes. The moments of data gathering coincide precisely with the state changes in the Markov chain approach. For each network we simulate the competition process for 10^8 time units and log the time the process is in each state.

Figure 4.5 shows a heatmap representation of the steady-state distribution of the Markov chain of the virus competition in a complete graph. The number of infected nodes with virus A (B) increases along the horizontal (vertical) axis similarly as in Figure 4.2. The hot spots along the axis show that the network is most likely in a state where one of the two viruses dominates the other. Either virus is equally likely to dominate the other as the symmetry along the $N_A = N_B$ axis indicates. The diagonal band connecting the two hotspots illustrates that the viruses alternate between being dominant and being dominated. The steady-state distribution confirms the behaviour seen in Figure 4.1. Figure 4.6 shows the same information as the heatmap but in such a way that multiple graph types and sizes are more easily compared. It draws the probability that the epidemic process is in a state with $(N_A - N_B)/N$ nodes infected. The probabilities $(N_A - N_B)$ are sums over all lines parallel to $N_A = N_B$ in Figure 4.5.

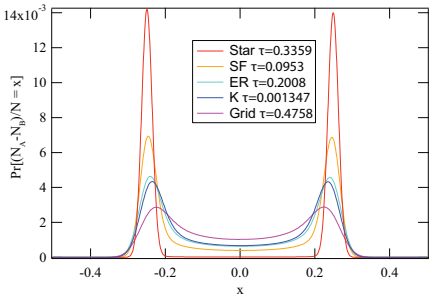
Figure 4.6a shows the probability that the network is in a state with $(N_A - N_B)/N$ nodes infected for different infection rates in a complete graph of 100 nodes. The different infection rates are chosen in such a way that a single virus in this network will cause



(a) The probability density function of $(N_A - N_B)/N$ in a complete graph of 100 nodes for different infection rates leading to different fractions of infected nodes for a single virus. $\tau \in \{0.0104, 0.0118, 0.0125, 0.0133, 0.0139, 0.0149, 0.0158, 0.0171\}$



(b) The probability density function of $(N_A - N_B)/N$ in a complete graph for various network sizes. At zero the number of nodes infected with virus A and B are the same. The infection rate for both viruses is the same and chosen to result in a 25% infection probability. $\tau \in \{0.0139, 0.0068, 0.0034, 0.0017, 8.34 \times 10^{-4}\}$



(c) The probability density function of $(N_A - N_B)/N$ for different network types of 1000 nodes. The infection rate for both viruses is the same and chosen to result in an average number of infected nodes of 25%.

Figure 4.6: The probability density function of $(N_A - N_B)/N$ for different fractions of infected nodes in the complete graph (a), different sizes of the complete graph (b) and different graph types of the same size (c).

an average fraction of infected nodes between 5 and 40 percent. For small fractions of infected nodes, both viruses will approach the absorbing state relatively frequently. The states around $N_A = N_B = 1$ are therefore visited often in the steady state and cause the high peak at $x = 0$. In the extreme case of no spreading, all states except for $N_A = N_B = 1$ are transient and the distribution features a single peak at $x = 0$. For higher infection rates the distribution shows two peaks around the average fraction of infected nodes that characterises the alternating domination. These two peaks are around the average fraction of infected nodes, indicating that one virus is responsible for almost all infections. They correspond to the hotspots in Figure 4.5. The symmetry around 0 indicates that both viruses are equally likely to dominate.

Figure 4.6b shows the distributions for complete graphs of different sizes ranging from 100 to 1,600 nodes. The infection rates are chosen in such a way that the steady-state fraction of infected nodes is 0.25. For larger networks, the probability of both viruses approaching the absorbing state becomes smaller as can be seen from the lower probability of being in a state where $(N_A - N_B)/N = 0$ for larger networks. Also, the peaks in the distribution tend to be closer to the average fraction of infected nodes, in this case 0.25, for larger networks. For $N = 1{,}600$ the peaks are located very close to the average fraction of infected nodes: around those peaks one virus is almost non-existent while the other has almost the entire fraction of infected nodes. The plateau between the two peaks indicates that during the cross-over from one domination to the other, on average the same number of nodes are infected by either virus.

The competition behaviour between two viruses is observed in various different graph types. Figure 4.6c shows the probability that the process is in a state where $(N_A - N_B)/N$ nodes are infected in five different graph types: the complete graph, the star graph, a scale-free graph grown following the preferential attachment paradigm, a connected Erdős-Rényi random graph with link probability $\log(N)/N$, and a 32 by 32 square grid; all graphs, except for the grid, contain 1,000 nodes. Again the effective infection rate of both viruses is the same and on average 25% of the nodes is infected.

In the most extreme graph, the star graph, the peaks indicating one virus dominating the other are much sharper than in the complete graph. Almost no time is spent in states where both viruses have a substantial number of nodes infected. For the scale-free graph the peaks are also markedly sharper than for the complete graph. The hub structure in both graph types probably enables a virus to dominate the other more easily and for longer, as will be discussed Section 4.4. The dominated virus has to compete for the hub nodes in order to have a chance to defeat the other.

In the case of an Erdős-Rényi random graph the difference with the complete graph is not as pronounced. The peaks are only marginally higher and the value at 0 is almost exactly the same. On the contrary, the 32x32 rectangular grid shows a much higher probability to be in a state where the number of infected nodes with virus A and virus B are more or less equal. The high hop-count and regular structure of the grid allows the two viruses to exist in more isolation from each other. It is still more likely to find the network in a state where one virus dominates the other, but to a much lesser extent than for the star graph.

The differences in hopcount, degree distribution and other structural properties of various graphs clearly have an influence on the competition process between the two

viruses, but overall, one virus will alternately dominate the other.

4.4. DOMINATION TIME OF MATCHED VIRUSES

Although the steady-state of the Markov chain shows that the process will most likely be in one of two regions in the state space diagram corresponding to one virus dominating the other (see Figure 4.5), it does not provide any insight in the alternating of dominating/dominated periods. In this section we investigate the domination time T of two matched viruses, i.e. $\beta_A = \beta_B, \delta_A = \delta_B$.

We say that virus A dominates virus B if there are more nodes infected with virus A than with virus B, i,e, $N_A > N_B$. A domination period of virus A starts when there are more nodes infected with virus A than with virus B and ends when there are more nodes infected with virus B than with virus A. The domination time is measured in the number of state changes between the start and end of a domination period. We first derive the expected domination time in Section 4.4.1 and compute the distribution of the domination time in Section 4.4.2.

4.4.1. EXPECTED DOMINATION TIME

Consider the example state space in Figure 4.7 for a network of 7 nodes. The Cartesian coordinates (N_A, N_B) of the state space encode the number of infected nodes with each virus. The states in which virus A dominates are coloured blue, the states in which virus B dominates are coloured green, and the states where virus A and B have the same number of infected nodes are coloured yellow.

The transition probabilities in this state space can be computed using the expressions for North, South, East and West derived in Section 4.3, with the exception of the states along the axes. These states have a self-loop in the Markov transition graph to avoid the virus from dying out. The probability of staying in the same state along the axes is equal to the probability of going one more step West or South.

Let T_A be the time that virus A dominates virus B. A domination period of virus A can start at any state just above the diagonal (states 2, 8, or 12 in Figure 4.7). Let's assume for now that the domination starts in state 8. The average domination time, $E[T_A]$, is the average hitting time of any state below the diagonal (the green states) given that we start in state 8. Using the same techniques we used in chapter 3.3 to find the survival time, the average hitting time from state 8 to any green state is the value k_8 for the minimal non-negative solution [54] to the following system of linear equations:

$$\begin{cases} k_i = 0 & \text{for } i \notin \{1,...,12\} \\ k_i = 1 + \sum_{j \notin \{1,...,12\}} p_{ij} k_j & \text{for } i \in \{1,...,12\} \end{cases} \tag{4.1}$$

where p_{ij} is the probability of going from state i to state j. The solution for k_i will generally be different for different values of i. In other words, the state in which a domination period starts influences the average duration. Figure 4.8 shows $E[T_A]$ in the complete graph K_{50} as a function of y_s, the number of infected nodes at the start of the domination period. Figure 4.8 also shows the measured average domination time of virus A as a function y_s and the measured probability density of y_s. The measured values are averages over 10^6 domination periods. The infection rate for both viruses was chosen in

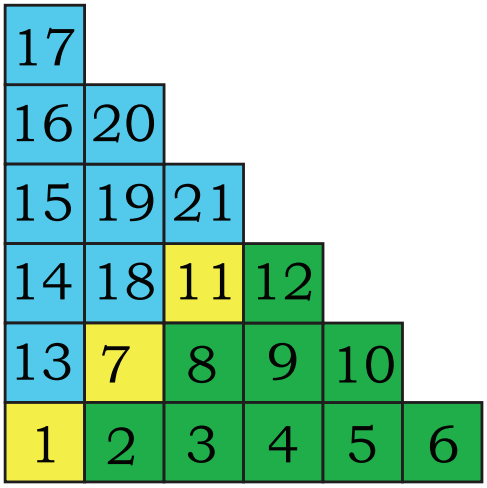


Figure 4.7: Example state space for a network with 7 nodes. States where virus A dominates (the lower half) are green, states where virus B dominates (the upper half) are blue, and states where the viruses are balanced are yellow. In this state diagram state 1 corresponds to $N_A = N_B = 1$.

such a way that y, the expected number of infected nodes, is 33, which is about 65% of the network. Moreover, with 33 nodes infected it is possible for a domination period to start with precisely the average number of nodes infected, i.e. $y_s = y$.

Figure 4.8 demonstrates that our simulation results are in agreement with the numerical solution to the linear system in (4.1). For values of y_s that occur only rarely, the simulated results are further from the numerical results because of a lack of statistics. The average domination time of a virus depends on the viral state of the network at the start of domination. When $y_s < y$ a domination period can be significantly longer than when the domination period starts close to the expected number of infected nodes. This means that in a situation with only two nodes infected, the virus that spreads first is likely to be stronger than the other for initially a rather long period. After the process converges, the influence of the initial advantage will disappear.

The effect that a domination period is longer if it starts when the number of infected nodes is smaller than the expected number of infected nodes ($y_s < y$) is more pronounced for larger infection rates. The average domination time T_A for periods that start with more nodes infected is only slightly larger for larger infection rates. This means that the initial domination period for a process that starts with only two nodes infected can be very long for high infection rates, but will decrease when the other virus catches up, since it is very unlikely to find the network in a state with few nodes infected after the start of the process.

The distributions of y_s as shown in Figure 4.8, are identical to the distributions of the number of infected nodes for the three different values of the infection rate. The distributions for y are not shown in Figure 4.8 to avoid further cluttering. Because the probability of a domination period to start with y_s infected nodes is the same as the probability of the network to have y nodes infected, it is unlikely that the beginning of a

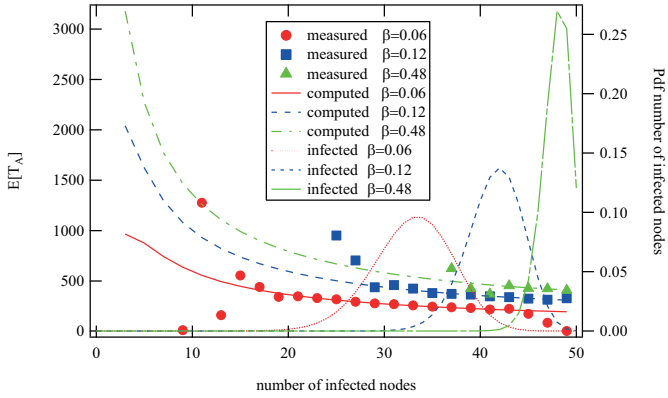


Figure 4.8: Measured and calculated domination time in a complete graph of 50 nodes as a function of the number of infected nodes at the start of the domination period for three different infection rates β, and unit curing rate. The right hand side vertical axis shows the measured probability density function of the number y_s of infected nodes at the start of the domination period.

domination period is related to any variations in the total number of infected nodes. If this would be the case, the number of infected nodes at the beginning of a domination period would have been higher or lower than the expected number of infected nodes. However, because of the dependency of T_A on the number of infected nodes y_s, if a domination period starts during a spell of few infected nodes, it is likely to last longer.

4.4.2. DOMINATION TIME DISTRIBUTION

The knowledge of the domination period is not complete without the distribution of the domination time, as the average might be a poor descriptor. We consider again the state space as illustrated in Figure 4.7. To compute the distribution of the domination time, we make all the states below the diagonal (the green states) absorbing. The Markov chain that describes such a process consists of a set of transient states in which virus A dominates virus B and a set of absorbing states.

Let M be the submatrix of the transition matrix P that only contains the transitions between the transient states. In the example of Figure 4.7, M is a 12×12 substochastic matrix containing the green and yellow states. Let s_k be the state distribution after k steps. In our example case, $s_0 = e_8$ (an all zero vector with only the 8^{th} element one) since we assume that the domination of virus A starts in state 8. The L_1 norm of s_k is the probability that after k steps the process is in one of the transient states. If the process is still in one of the transient states after k steps, then the domination time has to be larger than k. This means that $||s_k||_1 = \Pr[T_A > k]$, and $\Pr[T_A = k] = ||s_k||_1 - ||s_{k-1}||_1$, for $k > 0$.

Let M be the transition matrix containing the transitions within the transient region. Assuming that all eigenvalues of M are distinct, M can be decomposed as $M = V\Lambda V^{-1}$, where V is a matrix with the eigenvectors of M as its columns and Λ is a diagonal matrix

of the eigenvalues of M. The k step transition probabilities, M^k can be expressed as $V\Lambda^k V^{-1}$. The probability of being in a transient state after k steps starting from some initial state s_0 are found [27] by summing the k step probabilities in row s_0 of M^k.

$$\Pr[T \geq k] = \sum_{j=1}^{N} (M^k)_{s_0,j}$$

Because Λ is a diagonal matrix, the entries of row s_0 in $V\Lambda^k$ can be written as $v_{s_0,j}\xi_j^k$, where ξ_j is the jth eigenvalue of M. The sum over the entries of row s_0 in M^k can be written as

$$\sum_{j=1}^{N} (M^k)_{s_0,j} = \sum_{j=1}^{N}\sum_{r=1}^{N} v_{s_0,j}\xi_j^k \tilde{v}_{r,j} = \sum_{j=1}^{N} \xi_j^k c_j,$$

where $c_j = \sum_{r=1}^{N} v_{s_0,j}\tilde{v}_{r,j}$ and $\tilde{v}_{r,j}$ are the elements of V^{-1}. Using $\Pr[T = k] = \Pr[T \geq k] - \Pr[T \geq k+1]$ the probability of being in a transient state after k steps, which is the same as the domination time, can be written as

$$\Pr[T = k] = \sum_{j=1}^{N} \xi_j^k c_j (1 - \xi_j)$$

which is a weighted sum of exponentials. Because M is a substochastic matrix, all eigenvalues are smaller than 1, and the probability of being in a transient state vanishes. After sufficiently long time, only the largest eigenvalue has an influence on the distribution of the domination time leading to a purely exponential tail.

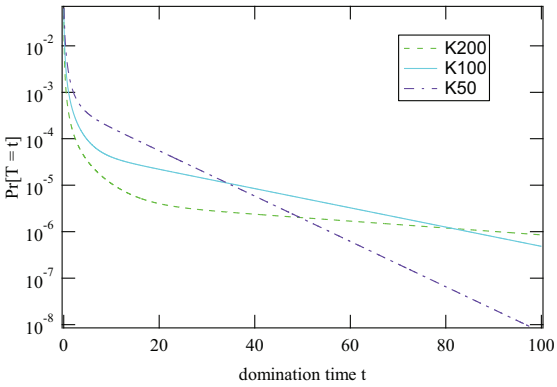


Figure 4.9: Probability distribution of the domination time T in the complete graph for three different network sizes. For each network size $y_s = y = N/4$.

The hitting time T of the absorbing region in the embedded Markov chain is expressed in the number of transitions. In order to transform the number of state changes to time units in the continuous-time Markov process, we create a uniformised embedded Markov chain, just as we did in Chapter 3.3. In such a chain, self-loops ensure that

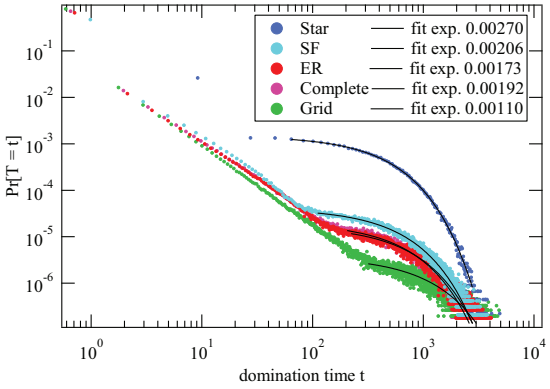


Figure 4.10: Measured probability density function of the domination time T for a domination period that starts with $y_s = y$ in five different graph types of 1000 nodes. The infection rate is chosen so that the $y = N/4$.

transitions in the embedded chain happen at the same rate as in the continuous time process. As a result, the steady-state distributions of the uniformised embedded Markov chain and the continuous-time Markov chain are the same. The number of state changes is transformed into time by dividing by the maximum transition rate in the continuous time process.

Figure 4.9 shows the distribution of the domination time T (starting from $y_s = y = N/4$) in a complete graph for $N = \{50, 100, 200\}$. Small domination times are most likely, however, much longer domination times are also possible as a result of the exponential tail with small exponent. The exponential tail of all three distributions can be clearly seen on the log-lin scale. For increasing network sizes the slope of the tails becomes increasingly flat, indicating that for larger networks a virus can more easily dominate for a longer time.

The domination times for the five different graph types discussed in Section 4.3.1 are shown in Figure 4.10. The tails of all these distributions are exponential, but the differences lie in where the tail starts. For the star graph the exponential tail starts earliest. Not only are the domination periods more pronounced for the star graph, the probability of having long domination periods is much higher than for the other graphs, although the exponent is larger. For the grid the exponential tail starts the latest, indicating that small domination times are most likely. The parts of the distributions for smaller values for the domination time are very similar for all graphs except for the star graph.

4.5. DOMINATION TIME OF NON-MATCHED VIRUSES

When the two viruses are not matched, the domination periods will no longer be the same for the two viruses. We investigate two different ways in which the two viruses can be non-matched: in speed and in effective infection rate. A further line of research is to investigate two viruses that have different spreading and curing rate distributions, but the same average number of spreading events during an infection period, motivated by

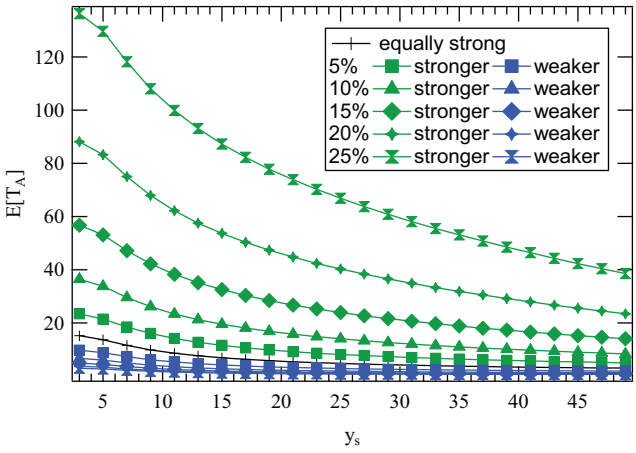


Figure 4.11: Average duration of a domination period for two viruses with different effective infection rates as a function of y_s. The green curves indicate the stronger of the two viruses, whereas the blue curves indicate the weaker of the two.

the results in chapter 5. The average number of spreading events during an infection period is a more natural way to express the spreading effectiveness in SIS processes as shown in the next chapter. However, the survival time of a virus depends on the inter-arrival time distribution of the infection attempts.

4.5.1. DOMINATION TIME FOR A STRONGER VIRUS

In order to investigate the effect that one virus is stronger than the other, we vary the strength of one virus with respect to the other and compute the expected domination time. The curing and infection rates for the two viruses are given as

$$\beta = \begin{cases} \gamma\beta_0 & \text{for virus } A \\ \beta_0 & \text{for virus } B \end{cases} \qquad \delta = \begin{cases} \delta_0 & \text{for virus } A \\ \delta_0 & \text{for virus } B \end{cases}$$

where β_0 and δ_0 are scaled for different values of γ to ensure that the average number of infected nodes stays 33. The value of 33 (around 65%) is arbitrary, but the odd number of expected infected nodes means a domination period can start at exactly the expected number of infected nodes.

Figure 4.11 shows the expected duration of a domination period for both the stronger and weaker viruses for six different strength ratios. The case in which the two viruses are matched is also shown for reference. As intuitively expected, the stronger viruses dominates the weaker virus for longer, but for values of γ not too far removed from 1, the weaker virus can still dominate the stronger one.

The effect of the difference in effective infection rate is almost independent of y_s for small values of γ, as illustrated in Figure 4.12 where the ratio $E[T_B]/E[T_A]$ is plotted for various values of γ as a function of y_s. The ratio between the expected domination

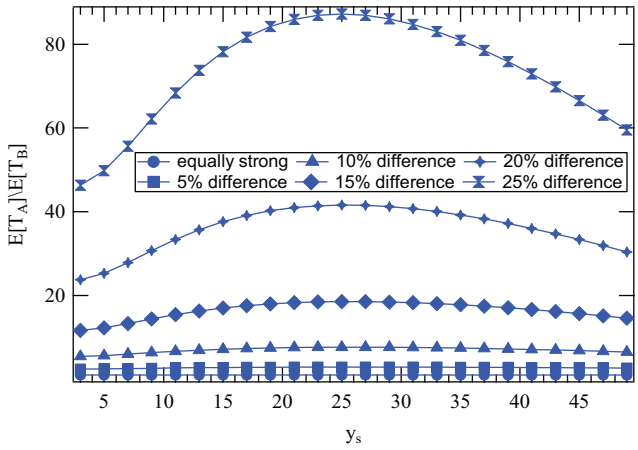


Figure 4.12: Ratio between the average duration of a domination period for the stronger virus and the duration of a domination period for the weaker virus for various differences in strength.

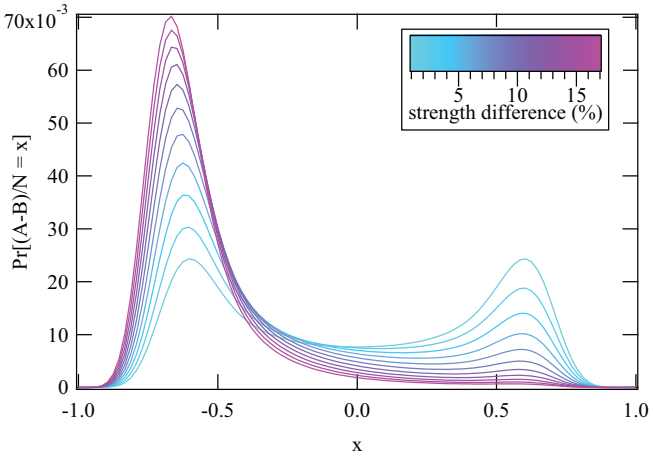


Figure 4.13: Probability density function of $(N_A - N_B)/N$ for a complete graph of 50 nodes where $\tau_B = \gamma\tau_A$, with $1 \leq \gamma \leq 1.17$.

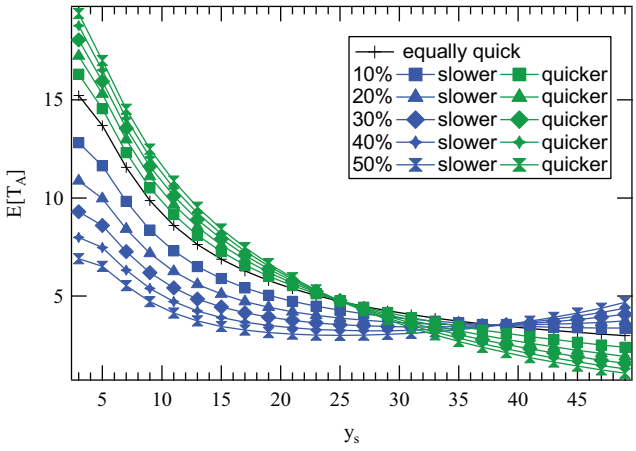


Figure 4.14: Average duration of a domination period for two viruses with different spreading speed as a function of the number of infected nodes at the start of the domination period. The green curves indicate the quicker of the two viruses, whereas the blue curves indicate the slower of the two.

time for the weaker virus and the stronger virus increases with y_s. The maximum ratio between the weaker and stronger virus occurs when y_s is smaller than the expected number of infected nodes, but not too much smaller.

Figure 4.13 shows the probability of the network being in a state with $(N_A - N_B)/N$ nodes infected for different values of τ_B where $\tau_B > \tau_A$. When virus B is only a few percent stronger than virus A, there is still a fairly high probability of finding the network in a state where virus A dominates, but that probability drops quickly with an increasing difference in strength. Viruses need to be matched in strength within a few percent to show any real competition.

4.5.2. DOMINATION TIME OF A QUICKER VIRUS

Two viruses that have the same effective infection rate can still differ in behaviour. In this section we compare the average domination periods of viruses that differ in speed, keeping the effective infection rate $\tau = \frac{\beta}{\delta}$ constant. The infection and curing rates are given by

$$\beta = \begin{cases} \alpha\beta_0 & \text{for virus } A \\ \beta_0 & \text{for virus } B \end{cases} \qquad \delta = \begin{cases} \alpha\delta_0 & \text{for virus } A \\ \delta_0 & \text{for virus } B \end{cases}$$

Where β_0 and δ_0 are the starting values for the virus process such that the average number of infected nodes is 33. Contrary to the case of the varying mutual strength, the difference in speed does not lead to a difference in the *total* number of infected nodes.

Figure 4.14 plots the average domination time as a function of y_s for various values of α. As expected, the quicker virus has an advantage for almost all values of y_s, but surprisingly, when $y_s > y$ being quicker becomes disadvantageous. This can be explained by the fact that when a domination period starts during a period when the number of

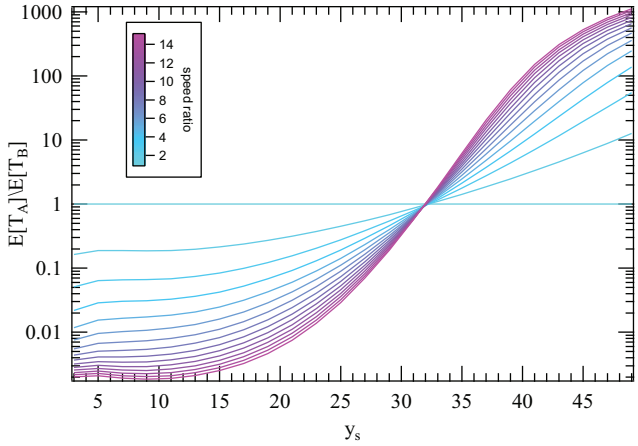


Figure 4.15: Ratio between the average duration of a domination period for the slower virus and the duration of a domination period for the quicker virus for various differences in speed: $\beta_B = \alpha\beta_A$ for $1 \leq \alpha \leq 15$.

infected nodes is above the average, there is a driving force that will push the number of infected nodes back to the mean. Since the quicker virus dominates, there are more nodes infected with the quicker virus. All these nodes have a smaller curing time and it is therefore likely that the number of nodes infected with the quicker virus will decrease before the nodes infected with the slower virus will decrease, thereby shortening the expected domination time of the stronger virus.

Because of the disadvantage of being a quicker virus for values of y_s that are above the expected number of infected nodes, the effect of α on the ratio between the expected domination time of a slower virus and a quicker virus is not constant over y_s as for viruses that differ moderately in strength. Figure 4.15 shows the ratio $E[T_B]/E[T_A]$ for different values of α as a function of y_s. The shapes of these curves are markedly different from those of the strength difference in Figure 4.12. This means that although it is possible to balance a stronger virus with a quicker virus for a *single* value of y_s it is not possible for the entire range of y_s. Also, the curves are symmetric around $y_s = y$, resulting in a balance in the total time the quicker virus dominates the slower and vice-versa.

4.6. GENERALISED EPIDEMIC MEAN-FIELD MODEL

Recently, the Generalised Epidemic Mean-Field Model (GEMF) has been introduced by Darabi Sahneh *et al.* [74] as a generalisation of NIMFA, the N-Intertwined mean-field approximation [29, 74]. GEMF uses a node level description of the spreading process to arrive, through a mean-field approximation, at a set of non-linear ordinary differential equations that describe the network state as a whole. Nodes can be in one of several states (or compartments) and interact through a multi layer network depending on their state. The node level description of the process is given by a set of transition rate graphs; each transition rate graph can either be a node-based rate graph, if the rate is independ-

ent of the state of a node's neighbours, or an edge-based transition graph if the transition does depend on the state of a node's neighbours. In the case of the MSIS process, curing is a node-based transition, whereas spreading is an edge-based transition. The edge-based transitions are dependent on an influencer state. The influencer state is a label that indicates that node i's transition from one state to another depends on the number of neighbours of node i in the indicated state. The MSIS process with two viruses has three states in the GEMF model; the susceptible state (S), the infected with virus A state (I_A), and the infected with virus B state (I_B). Figure 4.16 shows the transition rate graphs for this MSIS process: a) The nodal rate graph representing the curing of infected nodes, which happens with curing rate δ_A for virus A and curing rate δ_B for virus B. b) The transition from state S to I_A, which happens with rate β_A per link and has influencer state I_A. c) The transition from state S to I_B, which happens with rate β_B per link and has influencer state I_B. The interaction networks for states I_A and I_B are identical.

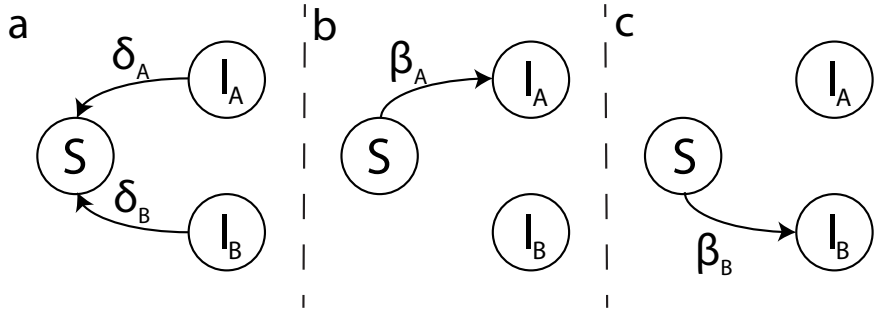


Figure 4.16: Transition rate graph of the MSIS process. a) Nodal transition rate graph: The three states are susceptible (S), infected with virus A (I_A), and infected with virus B (I_B). A node moves from state I_A to S with curing rate δ_A and from I_B to S with curing rate δ_B. b) Edge-based transition graph for type A: A node moves from state S to state I_A with infection rate $\beta_A$for each incident link from a node in I_A. c) Edge-based transition graph for type B: A node moves from state S to state I_B with infection rate $\beta_B$for each incident link from a node in I_B. The contact networks for I_A and I_B are identical.

The state $x_i(t)$ of node i at time t is represented by the standard unit vector corresponding to the state i. The expected value of $x_i(t)$ is defined as ([74, eq. (2)])

$$E[x_i(t)] \quad = \quad [\Pr[x_i(t) = e_i], \ldots, \Pr[x_i(t) = e_M]]^T \triangleq v_i(t)$$

where e_i is the i-th standard unit vector and M the number of states. The governing equations of the GEMF model follow from the transition graphs ([74, eq. (27)]):

$$\frac{dv_i}{dt} = -Q_\delta^T v_i - \sum_{l=1}^{L} \Big(\sum_{j=1}^{N} a_{i,j|l} v_{j,q_l} \Big) Q_{\beta_l}^T v_i, i = \{1, \ldots, N\} \tag{4.2}$$

where Q_δ is the Laplacian matrix of the nodal transition rate graph, $Q_{\beta_l}$are the edge transition graphs for the various layers, and $a_{i,j|l}$ are the adjacency matrix elements of the network describing the interactions on layer l, q_l is the influencer state at layer l, L is the number of network layers and N is the number of nodes in the network.

For the two-virus process described by the transition graphs in Figure 4.16, we have the following Laplacian matrices:

$$Q_\delta = \begin{bmatrix} 0 & 0 & 0 \\ -\delta_A & \delta_A & 0 \\ -\delta_B & 0 & \delta_B \end{bmatrix}, Q_{\beta_A} = \begin{bmatrix} \beta_A & -\beta_A & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},$$

$$Q_{\beta_B} = \begin{bmatrix} \beta_B & 0 & -\beta_B \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.3}$$

The interaction networks for the two infectious states, I_A and I_B, are identical for the 2-MSIS process, so we can drop the index l from $a_{i,j|l}$. As indicated above, the influencer states for I_A and I_B are the states themselves. Equation (4.2) in the case of 2-MSIS reduces to:

$$\dot{v}_i = -Q_\delta^T v_i - \sum_{j=1}^{N} a_{ij} v_{j,I_A} Q_{\beta_A}^T v_i - \sum_{j=1}^{N} a_{ij} v_{j,I_B} Q_{\beta_B}^T v_i \tag{4.4}$$

which we can expand to

$$\begin{bmatrix} \dot{S}_i \\ I_{A_i} \\ I_{B_i} \end{bmatrix} = - \begin{bmatrix} 0 & -\delta_A & -\delta_B \\ 0 & \delta_A & 0 \\ 0 & 0 & \delta_B \end{bmatrix} \begin{bmatrix} S_i \\ I_{A_i} \\ I_{B_i} \end{bmatrix} - \sum_{j=1}^{N} a_{ij} I_{A_j} \begin{bmatrix} \beta_A & 0 & 0 \\ -\beta_A & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} S_i \\ I_{A_i} \\ I_{B_i} \end{bmatrix} - \sum_{j=1}^{N} a_{ij} I_{B_j} \begin{bmatrix} \beta_B & 0 & 0 \\ 0 & 0 & 0 \\ -\beta_B & 0 & 0 \end{bmatrix} \begin{bmatrix} S_i \\ I_{A_i} \\ I_{B_i} \end{bmatrix} \tag{4.5}$$

If $\beta_A = \beta_B$ and $\delta_A = \delta_B$, the total number of infected nodes equals the number of infected nodes of a single SIS process. We take the first row in Equation (4.5).

$$\frac{dS_i}{dt} = \delta_A I_{A_i} + \delta_B I_{B_i} - \sum_{j=1}^{N} a_{ij} I_{A_j} \beta_A S_i - \sum_{j=1}^{N} a_{ij} I_{B_j} \beta_B S_i$$

$$= \delta(I_{A_i} + I_{B_i}) - S_i \beta \sum_{j=1}^{N} a_{ij} (I_{A_j} + I_{B_j}) \tag{4.6}$$

After using $I = I_A + I_B$ and $S + I = 1$, Equation (4.6) reduces to

$$\frac{dS_i}{dt} = \delta(1 - S_i) - S_i \beta \sum_{j=1}^{N} a_{ij} I_j \tag{4.7}$$

which is the governing equation of the N-Intertwined Mean-Field Approximation of the normal SIS processes, see, for example, [29, Eq (1)].

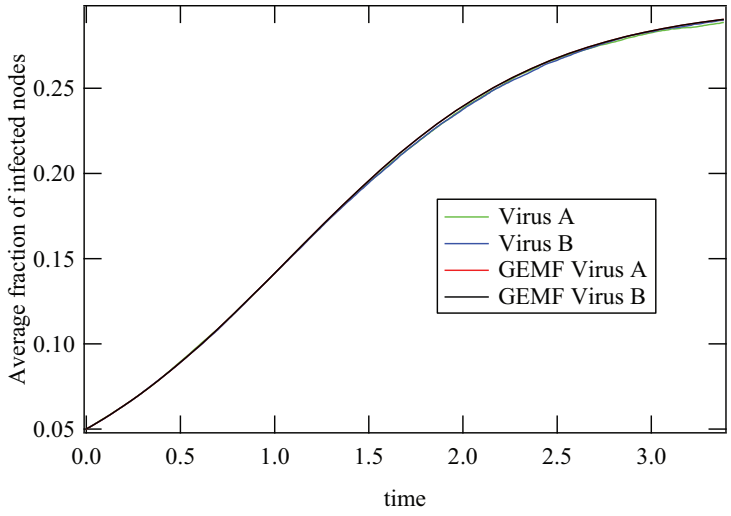


Figure 4.17: Comparison between the average number of infected nodes in the GEMF model and simulations in a complete graph of 500 nodes. The simulation results are averaged over 100,000 runs.

4.6.1. EVALUATION OF THE GEMF MODEL

To verify the correctness of the GEMF model description of the MSIS process, we compare the numerical solution of Equation (4.4) to simulations in a complete graph of 500 nodes. The average number of infected nodes per virus type as a function of time can be computed from Equation (4.4) as the average number of nodes in one of the infectious states. For the simulations, we have averaged the number of infected nodes at regular sample intervals over 100,000 runs using the event based simulator first described in [75]. At the start of each simulation 20% of the nodes are infected with virus A and another 20% of the nodes are infected with virus B.

In Figure 4.17 the average fraction of infected as computed from Equation 4.4 and simulations are plotted as a function of time. The GEMF model and the simulations both coincide and show that the average fraction of infected nodes is the same for viruses A and B. Yet, Figure 4.17 should be interpreted with care. Just as mean field models for normal SIS, the GEMF model for MSIS cannot describe the process of a virus dying out. In reality, one of the two viruses rapidly dies out as a result of the competition.

The effects of the competition can be highlighted by preventing the viruses from dying out. This is done by reinfecting the last node of a virus type at the moment it is cured. Figure 4.1 shows the fraction of infected nodes per virus type as a function of time, again in a complete graph of 500 nodes. In contrast to Figure 4.17, the fraction of infected nodes is doubled for the dominant virus while it is very small for the dominated virus. Interestingly, the two viruses alternate from being dominant to being dominated. This behaviour is further discussed in Section 4.3.1.

4.7. CHAPTER SUMMARY

When two viruses compete for healthy nodes in a network, we show that one of the two viruses dies out exponentially fast, even when both viruses have an effective infection rate above the epidemic threshold. However, when we prevent two equally strong viruses from dying out, a rich dynamic process emerges where one of the two viruses dominates the other but loses that dominance after some time. We show that the average domination time of a virus depends on the number of infected nodes at the beginning of the domination period and that it can be computed by solving a linear system. The distribution of the domination time can also be computed numerically.

When the two viruses are not balanced, but differ in either strength or speed, the domination times will also be unbalanced. If the differences are not too big, the weaker and/or slower viruses still periodically dominates their quicker and/or stronger rival, but only for a short time. Because the effect of speed and strength as a function of the number of infected nodes at the start of the domination time differs, it is impossible to balance a stronger virus with a quicker rival for all values of y_s, but it is possible to balance them for a single value of y_s. Contrary to being stronger than a rival, being quicker is not always a benefit but depends on both how much quicker the virus is and how many nodes are infected at the beginning of a domination period. Interesting future work includes the effect of non-Markovian spreading properties of both viruses. In this case the strength of two viruses can be matched while they differ in distributions for the inter-arrival rate of infection and curing events.

5

NON-MARKOVIAN SIS

SIS epidemics in networks implicitly assume Markovian behaviour: the time to infect a direct neighbour is exponentially distributed. Much effort so far has been devoted to characterise and precisely compute the epidemic threshold in SIS Markovian epidemics on networks. In this chapter we show the rather dramatic effect of a non-exponential infection time (while still assuming an exponential curing time) on the epidemic threshold by considering Weibullean infection times with the same mean, but different shape parameter α. For all investigated graph classes, the epidemic threshold significantly increases with the shape parameter α. By using the average number of infection attempts during an infectious period in stead of the effective infection rate, however, the influence of the inter-arrival time distribution on the epidemic threshold is greatly reduced. Moreover, the NIMFA equations are still valid in that case. The accuracy of NIMFA and the survival time of the epidemic depend on the inter-arrival time distribution.

5.1. INTRODUCTION

In the SIS model of virus spread that we have seen so far, the infection and curing processes are modelled as Poisson processes. Poisson processes are nice to work with because of the memoryless property of the exponential inter-arrival time distribution. However, the curing time of an epidemic, be it a disease or computer virus or an opinion, is most likely not exponentially distributed. Similarly, the inter-arrival times of infection events are probably not exponential random variables. Research into human online behaviour has revealed a wealth of evidence of heavy-tailed distributions in various situations, such as email traffic, website usage, instant-messaging and phone calls [59, 76–82]. In this chapter we describe the effects of modelling the inter-arrival times of curing and infection events by non-exponential distributions. Clearly this turns the SIS model into a non-Markovian process and we can no longer use Markov theory to describe the infectious state of the network.

5.2. THE NON-MARKOVIAN EPIDEMIC THRESHOLD

As described in Chapter 2.1, the epidemic threshold of a network distinguishes between the overall-healthy network regime and the effective infection regime where permanently a non-zero fraction of the nodes is infected. The epidemic threshold reflects how effective an epidemic is in a particular network and is a major indicator or tool to protect the nodes (people, computers, ...) and to take preventive measures (governmental immunisation strategies, anti-virus software protection).

Recently (see e.g. [17, 23, 26, 49, 83–85]) much effort has been devoted to the precise computation of the epidemic threshold in the continuous-time SIS Markov model in networks. A basic property of a continuous-time Markov SIS process is the exponentially distributed infection time [27, p. 184]: a node i infects its neighbours at an exponential time T with mean $\frac{1}{\beta}$. Only the exponential distribution possesses the memoryless property which enables to reduce the process history to only the previous event to compute the current one. Hence, all events in the past are uncoupled as if the process restarts in the state at the previous event [27, 349-351]. Thus, the memoryless property of the exponential distribution makes Markovian processes attractive and analytically tractable. When a stochastic process is not Markovian, its mathematical description and analysis is considerably more complex. For example, the relatively simple and intuitive equation (2.8) and, more generally, the Chapman-Kolmogorov basic equations [27, p. 180] that characterise Markov processes do not apply anymore. The (non-Markovian) time-depending branching process [86], that is still tractable, exemplifies the increased mathematical complexity, while a non-Markovian epidemic model is analyzed in [87]. Hinrichsen [88] has overviewed the recent progress in the field of non-equilibrium phase transitions into absorbing states with long-range interactions and non-Markovian effects. In many non-Markovian processes on networks, computer simulation or measurement is often the only resort to investigate its behavior and properties.

As it turns out, the effect of a non-exponential infection time on the average steady-state fraction of infected nodes is rather dramatic. In this section, while the curing process is still Poissonean with rate δ, the infection process at each node infects direct neighbours in a time T that is Weibullean distributed [27, p. 56], with probability density function

$$f_T(x) = \frac{\alpha}{b}\left(\frac{x}{b}\right)^{\alpha-1} e^{-\left(\frac{x}{b}\right)^{\alpha}} \tag{5.1}$$

and mean $E[T] = b\Gamma\left(1+\frac{1}{\alpha}\right)$. In (5.1), α is generally called the shape parameter, and b the scale parameter. In order to compare the Weibull with the exponential distribution, we fix the average infection time to be equal to that of an exponential distribution with rate β as $\frac{1}{\beta}$, so that

$$b = \left(\Gamma\left(1+\frac{1}{\alpha}\right)\beta\right)^{-1}$$

Thus, the shape parameter α in (5.1) tunes the power-law start and the tail of the Weibull distributions that all have the same mean infection time $E[T] = \frac{1}{\beta}$, but variance equal to

$$\mathrm{Var}[T] = \frac{1}{\beta^2}\left(\frac{\Gamma\left(1+\frac{2}{\alpha}\right)}{\Gamma^2\left(1+\frac{1}{\alpha}\right)} - 1\right)$$

When $\alpha = 1$, the Weibull distribution reduces to the exponential distribution. For $\alpha < 1$, the tail decreases slower, but the probability of small infection times increases as a power law, proportional to $x^{\alpha-1}$. In the extreme limit for $\alpha \to 0$, the Weibull distribution tends to the Zipf distribution. For $\alpha < 1$, the distribution is heavy-tailed and reduces to a Zipf distribution for $\alpha = 0$. For large values of α, the tail falls off exponentially and for $\alpha \to \infty$ the Weibull distribution reduces to a Dirac function. In addition to the natural generalisation of the exponential distribution, the Weibull distibution also recently appeared in epidemic studies [89].

The SIS process with Weibullean infection times is simulated as explained in Section 2.3.1 and Appendix B: the complicating absorbing state is removed so that the infection always remains in the network. When only one node is infected, the process restarts. At the precise moment that the last infected node is cured, that same node is reinfected. In order to determine the steady-state of the fraction of infected nodes we run two simultaneous, but independent simulations on the same network. One simulation is initialised with a fraction of infected nodes equal to 10%, whereas the second simulation starts with all nodes infected. During the simulation, the time averaged number of infected nodes is measured. After running for 100,000 state changes (or events) per simulation, we start comparing the average number of infected nodes in the two simulations and conclude that the steady-state is reached when $\frac{|\bar{y}_1(t)-\bar{y}_2(t)|}{\bar{y}_1(t)+\bar{y}_2(t)} < 10^{-4}$, where $\bar{y}_1(t)$ is the average fraction of infected nodes in the first simulation as a function of time and $\bar{y}_2(t)$ that of the second. The steady-state fraction of infected nodes is taken to be $\frac{\bar{y}_1(t)+\bar{y}_2(t)}{2}$. Fig-

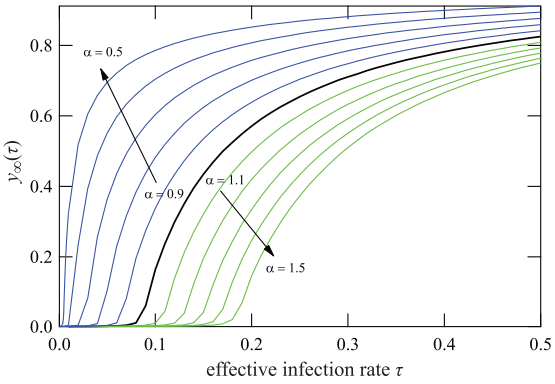


Figure 5.1: Steady-state fraction of infected nodes as a function of the effective infection rate for various Weibull exponents on ER networks ($N = 500, p = 2p_c, d_a = 12.4$).

ure 5.1 shows the average steady-state fraction $y_\infty(\tau) = \lim_{t\to\infty} \frac{1}{N} E\left[\sum_{j=1}^{N} X_j(t)\right]$ of infected nodes in an Erdős-Rényi (ER) random graph $G_p(N)$ with $N = 500$ nodes and with link density $p = 2p_c$, where the disconnectivity threshold [27, p. 338] equals $p_c \sim \frac{\log N}{N}$, versus the effective infection rate τ for various α, both larger and smaller than 1. The curves for $y_\infty(\tau)$ clearly shift to the right with increasing α. More dramatically, when α

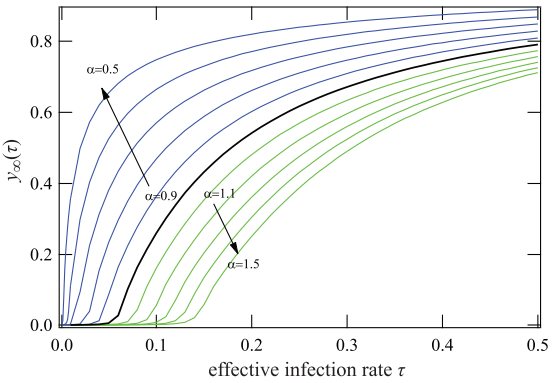


Figure 5.2: Steady-state fraction of infected nodes as a function of the effective infection rate for various Weibull exponents on SF networks ($N = 500, L = 2970, d_a = 11.8$).

decreases, the epidemic threshold also decreases. Not only does the threshold decrease, the smaller α, the quicker the steady-state fraction of infected nodes increases when the epidemic threshold is passed. In other words, the derivative $\frac{d}{d\tau} y_\infty(\tau)$ increases with decreasing α for τ above the epidemic threshold: the epidemic infects a larger number of nodes for the same average strength τ. Hence, in non-Markovian SIS epidemics with the same mean infection time (and the same effective infection rate τ), the sensitivity of the power law for values of α smaller than 1 on the epidemic threshold is large. A similar tendency is found in other types of graphs, for example a scale-free (SF) graph[1] (Figure 5.2) and a rectangular grid (Figure 5.3). In the case of the grid, the curves for successively smaller values of α are spaced further apart, while for larger values of α the opposite holds.

For small values of $\alpha < 1$, the Weibull pdf $f_T(t)$ in Figure 5.4 shows that small infection times are more likely to occur, implying that many short infection attempts are fired to neighbouring nodes, interchanged with a relatively long inactive time (because the mean $E[T]$ is constant). The exponential curing process is less effective to counteract the fast infection attempts leading to more infected nodes (higher $y_\infty(\tau)$ for the same τ) for all tested networks (Figures 5.1, 5.2, and 5.3. Similarly, when $\alpha > 1$, small inter-arrival times for curing events takes place more frequently than for infection attempts, thereby shifting $y_\infty(\tau)$ to larger effective infection rates τ. Figure 5.4 draws the measured probability density function of the infection time T for various values of the exponent α. The infection time is only logged if it leads to a spreading action; that is, if it is smaller than the current curing time of the infected node. In all simulations the curing time is exponentially distributed with a mean of 1 (i.e. $\delta = 1$ and $\tau = \beta$). As the infection time cannot be larger than the cure time of the infected node, the infection time distribution

[1] As SF graph, we have simulated a Barabási-Albert graph, whose initial topology is a ring with m nodes and where m is the number of links added per newly added node and chosen in such a way that the resulting SF graph has approximately the same number of links as $G_{2p_c}(N)$.

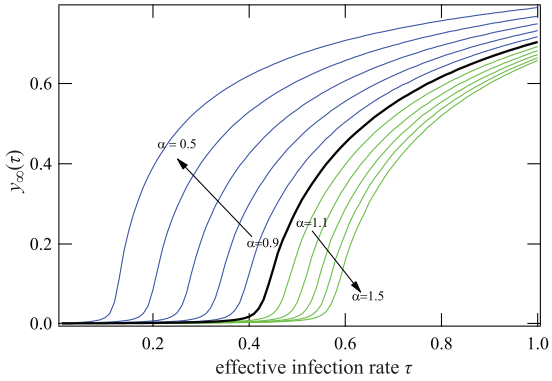


Figure 5.3: Steady-state fraction of infected nodes as a function of the effective infection rate for various Weibull exponents on a rectangular grid ($N = 484, L = 924, d_a = 3.8$).

is cut off for larger values by the exponential distribution of the cure time. Since the area below the curves equals one, by definition of a probabity density function, each curve for $\alpha \neq 1$ intersects with the exponential distribution ($\alpha = 1$) twice. If $\alpha < 1$, then small as well as very long infection times are more probable than for the exponential distribution, while the opposite holds for $\alpha > 1$, as can be seen from Figure 5.4. The thick line in Figure 5.4 depicts the exponential case of $\alpha = 1$. For each value of α approximately $7.5 \cdot 10^6$ infection times are logged.

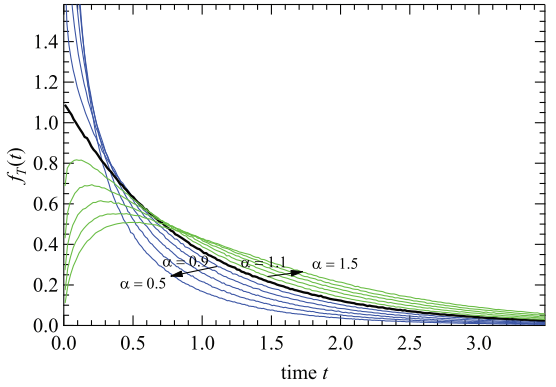


Figure 5.4: The effectively simulated probability density function $f_T(t)$ of the infection T for various α.

Figure 5.5 illustrates how the epidemic threshold varies as a function of α. For the Erdős-Rényi (ER) and scale-free (SF) graph, the epidemic threshold is approximately linear for $\alpha > 1$ and convex for $\alpha < 1$, while the grid exhibits an opposite trend (linear

for $\alpha < 1$ and concave for $\alpha > 1$). In addition to the larger effect of α on the epidemic threshold in lattices, the SIS epidemics in grids or lattices behaves, indeed, different from that of the denser graphs (in which the average degree increases with N). Also, mean-field approximations are less accurate for lattices than for the other two classes, as is shown in chapter 2.

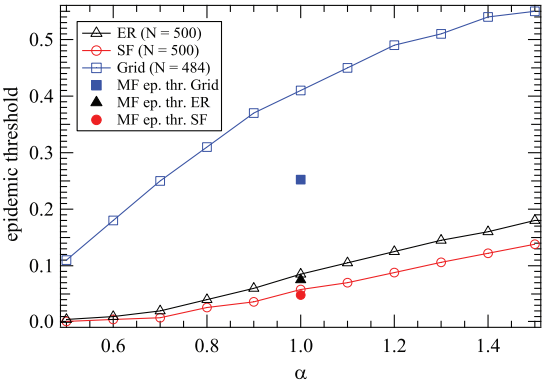


Figure 5.5: The epidemic threshold, deduced from Figure 5.1-5.3, versus the parameter α for the three types of graphs with approximately the same number of nodes N. The full markers correspond to the first-order mean-field epidemic threshold $\tau_c^{(1)} = \frac{1}{\lambda_1(A)}$ for $\alpha = 1$.

Finally, Figure 5.6 shows that the epidemic threshold τ_c for ER and SF graphs as a function of the inverse spectral radius λ_1 (for various sizes of N ranging from $N = 10^3$ up to $64\,10^3$) are all power laws, suggesting that

$$\tau_c(\alpha) = \frac{q(\alpha)}{\lambda_1^{r(\alpha)}} \tag{5.2}$$

For the simulated $\alpha \in [0.5, 1.5]$ and for both ER and SF graphs, we found that the exponent in (5.2) is approximately $r(\alpha) \approx \frac{1}{\alpha}$, but that $q(\alpha)$ is less accurate and not monotone in α. For $\alpha = 1$, $q(1) \simeq 1.2$ indicating that the NIMFA lower bound $\tau_c^{(1)} = \frac{1}{\lambda_1}$ is about 20% smaller (for the considered graphs). The corresponding scaling of the grid with N is different: all considered sizes (above $N = 1000$) have approximately the same epidemic threshold $\tau_c(\alpha)$ (because λ_1 is about 4 for all $N \geq 1000$ and nearly independent of N), but $\tau_c(\alpha)$ depends on α: $\tau_c(0.5) \simeq 0.11$, $\tau_c(1) \simeq 0.41$ and $\tau_c(1.5) \simeq 0.56$. Based on the general bound [30], $\max\left(d_{av}, \sqrt{d_{\max}}\right) \leq \lambda_1 \leq d_{\max}$, where d_{av} and $d_{\max}$ are the average and maximum degree in a graph, respectively, Chung *et al.* [90] have proved for SF graphs with degree distribution $\Pr[D \geq x] \sim x^{-\gamma}$ that $\lambda_1 \sim \sqrt{d_{\max}} = O\left(N^{\frac{1}{2\gamma}}\right)$ and Krivelevich and Sudakov [91] that $\lambda_1 = O(\ln N)$ for $G_{p_c}(N)$. Combining these exact scaling laws with the law for $\tau_c(\alpha)$ in (5.2) suggests us to conclude that the epidemic threshold in non-Markovian SIS epidemics will vanish with N as a power-law in SF graphs and as $O\left(\ln^{-\frac{1}{\alpha}} N\right)$ in ER networks.

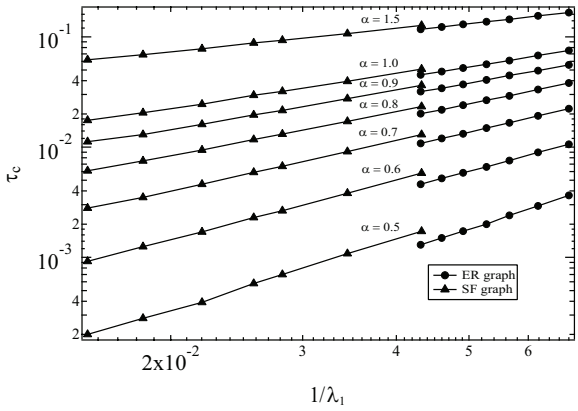


Figure 5.6: The epidemic threshold in ER and SF graphs for various sizes ($N = 2^k 10^3$, $0 \le k \le 6$) and value of $\alpha = (0.5, 1.0, 1.)$ versus the inverse of the spectral radius of the graph.

5

5.3. NIMFA IN NON-MARKOVIAN SIS

In the classic SIS model, as we saw in chapter 2, the infection and curing processes are modelled as Poisson processes. This ensures that the state of the process, describing for each node whether it is infected or not, is a Markov chain. This, in turn, offers the possibility to derive mean-field approximations. We know, however, that these exponential distributions are in general not describing real-life epidemics well, and from the previous section that different distributions for the inter-arrival times of infection and curing events leads large differences in the fraction of infected nodes. In this section we show, after Cator et al. [92], how the NIMFA approximation can be extended to include general infection and curing inter-arrival time distributions. The continuous-time SIS model on any network, in which the infection time T and the curing or recovery time R have a general distribution, is called the generalised SIS or GSIS model, of which the classical continuous-time SIS Markov model is a special case.

5.3.1. THE GSIS MODEL WITH GENERAL WAITING TIMES

Just as in chapter 2, the viral state of the network can be described by a vector of Bernoulli random variables X, such that $X_i = 1$ if node i is infected, and $X_i = 0$ if node i is healthy. If at time t node i gets infected, we draw independently of everything else a recovery time $R_i(t)$ and given $R_i(t)$, we independently draw, for each neighbouring, a random number $M_{ij}(t)$ of infecting times $T_{ij}^{(1)}(t) \le \ldots \le T_{ij}^{(M_{ij}(t))}(t) \le R_i(t)$ such that at times $t + T_{ij}^{(k)}$ node i tries to infect node j. If node j is already infected at this time, nothing happens. Finally, at time $t + R_i(t)$ node i recovers and becomes healthy, but again susceptible to infection. This is exactly the simulation process as described in section 2.2.1 and appendix B. Each node i has a random vector associated to it

$$Z_i(t) = (R_i(t), Y_{ij_1}(t), \ldots, Y_{ij_{d_i}}(t)),$$

where d_i denotes the degree of node i, hence the number of its neighbours, and where

$$Y_{ij}(t) = (T_{ij}^{(1)}(t), \ldots, T_{ij}^{(M_{ij})}(t)).$$

We assume that the distribution of $Z_i(t)$ does not depend on t, and if $t_1, \ldots, t_k$ are the times of infection for node i, we will assume $Z_i(t_1), \ldots, Z_i(t_k)$ to be i.i.d. Furthermore, we will assume that $Y_{ij_1}(t), \ldots, Y_{ij_{d_i}}(t)$ are i.i.d.

An exact analysis of the GSIS on any network is very likely intractable, so that only an approximate treatment seems possible.

5.3.2. MEAN FIELD APPROXIMATION

For the meta-stable state to exist, the distribution of Z_i has to meet two conditions. The average recovery time of a node i is finite, $E[R_i] < \infty$, and the number of infection events of a node i has to be finite as well, so we assume that $E[M_{ij}] < \infty$. The mean field approximation [13, 23] in this setting entails that, when we determine the effect of the neighbours on node i, we assume that node i does not influence its neighbouring nodes in some relevant way.

Let v_i be the probability that node i is infected in the meta-stable state and consider node j as a neighbour of node i. In a large time interval $[0, S]$, the number of times node j was infected is asymptotically linear in S, by the elementary renewal theorem [27, p. 145]. Since the length of an infected period equals $E[R]$, the number of infected periods is equal to $v_j S/E[R]$. During each infected period, node j will try to infect node i an average number $E[M]$ of times, so that the total number of infection attempts from node j to node i asymptotically equals $v_j SE[M]/E[R]$. The mean-field approximation decouples the infectious state of node j from that of node i, so that the fraction of infection events from j to i that is successful (i.e., node i was in fact healthy at the time of infection) equals $1 - v_i$. Hence, the total number of successful infections that node i will receive in the time interval $[0, S]$ will asymptotically be equal to

$$S \sum_{j \in U_i} \frac{E[M]}{E[R]} v_j (1 - v_i),$$

where U_i denotes the index set of neighbours of node i, so $U_i = \{j \mid a_{ij} = 1\}$. In the meta-stable state (where equilibrium holds), this number must equal the number of infected periods of node i in $[0, S]$, so that

$$S \sum_{j=1}^{N} a_{ij} \frac{E[M]}{E[R]} v_j (1 - v_i) = v_i \frac{S}{E[R]}.$$

In other words, we arrive, for any node $i \in G$, at

$$E[M](1 - v_i) \sum_{j=1}^{N} a_{ij} v_j = v_i, \tag{5.3}$$

which is exactly the same equation as in NIMFA in the exponential case, if we replace $\tau = \beta/\delta$ by $E[M]$ and compare (5.3) to (2.12). The expected number of infection events in

a Poisson process with intensity β within an exponential recovery time with expectation $1/\delta$ indeed equals the effective infection rate $\tau = \beta/\delta$.

The analogy with the NIMFA equations in [29, 31] allows us to transfer the NIMFA analytic framework to the generalised SIS model. It follows from (5.3) and the epidemic threshold theorem in [31] that a *lower bound* for the epidemic threshold in GSIS epidemics satisfies

$$m_c = E[M_c] = \frac{1}{\lambda_1} \tag{5.4}$$

where λ_1 is the largest eigenvalue of the adjacency matrix A of the graph G. Thus, if $E[M_c] > m_c$, then the epidemic process is eventually endemic (in the mean-field approximation), in which case a non-zero fraction of the nodes remains infected, else the epidemic process dies out after which the network is overall healthy. While $E[M_c]$ describes the activity of the viral agents in the SIS epidemics, the right-hand side in the (mean-field) epidemic threshold equation (5.4) reflects the structural properties of the underlying network and emphasises the role of the spectral radius λ_1 of the graph G.

A practical conclusion is that, irrespective of our knowledge of the underlying distributions for the infection and recovery times, we can determine the average meta-stable state fraction of infected nodes, only based on an estimate of the average number $E[M]$ of infection attempts during a recovery time R. In most practical cases, the quantity $E[M]$ is more easy to measure than the ratio $\tau = \frac{E[R]}{E[T]}$. The generalised epidemic threshold rule (5.4) provides a useful criterion to verify whether a certain viral agent, whose precise properties are unknown except for its average activity $E[M]$, will cause a pandemic in a network or not.

5.3.3. DETERMINATION OF $E[M]$

Although (5.3) allows us to solve a system of non-linear equations to compute an approximation of the fraction of infected nodes in the meta-stable state, regardless of the type of infection and curing processes, determining $E[M]$ in general is not trivial. Using renewal theory [27, Chapter 8], a general expression for $E[M]$ can be deduced [92]:

$$E[M] = \frac{1}{2\pi i} \int_C \frac{\varphi_T(z)\,\varphi_R(-z)}{1-\varphi_T(z)} \frac{dz}{z} \tag{5.5}$$

where the contour C encloses the whole $\mathrm{Re}(z) > 0$ plane, and φ_R is the probability generating function of the recovery time R and φ_T is the probability generating function of the infection time T. If either the infection time or the spreading time is exponentially distributed, (5.5) can be simplified, and also be determined without integration.

The infection time is exponential:
Since the infection time T is exponentially distributed, the infection process is a Poisson process and the mean number of Poisson events in an interval is given by its rate β multiplied by the average length of that interval, which is $E[R]$. As a consequence $E[M] = \beta E[R]$.

The recovery time is exponential:
At the start of the infection, two cases can happen: either $T_1 < R$ or $T_1 > R$. In the latter

case, $M = 0$. In the first case, $M \geq 1$ and at time T_1 everything starts anew: the distribution of $(R - T_1) \mid R > T_1$ is again exponential because of the memoryless property [27] of the exponential distribution. This means that M has a geometric distribution,

$$\Pr[M = k] = p^k(1-p), \quad \text{with} \quad p = \Pr[R > T_1] \text{ and } k \geq 0,$$

Since all infection times $T_1, T_2, \ldots$ are i.i.d. and have a same distribution as T, we find the mean of a geometric random variable as

$$E[M] = \frac{\Pr[R > T]}{\Pr[R \leq T]} = \frac{\Pr[R > T]}{1 - \Pr[R > T]}.$$

Finally, instead of evaluating the integral (5.5), the expected number of spreading events during the infection time of a node can be computed numerically via the expression $E[M] = \sum_{k=0}^{\infty} k \Pr[M = k]$. As above, let M be the number of infection events over a link during the infection time R of a node, the waiting time until a curing event, and let T be the waiting time until the next spreading event. The probability that exactly k spreading events occur during an infected period follows (see [27, p. 140]) from the renewal equivalence $\{M(t) \geq n\} \iff \{W_n \leq t\}$ as $\Pr[M = k] = \Pr[W_k \leq R] - \Pr[W_{k+1} \leq R]$, where $W_n = \sum_{k=1}^{n} T_k$ denotes the sum of n independent realisations of T. Let $f_Y(x) = \frac{d}{dx}\Pr[Y \leq x]$ denote the probability density function (pdf) of the random variable Y. Since the pdf of the sum of two independent random variables is given by the convolution of the two pdfs of the random variables, we can express the pdf of W_k as a series of convolutions, i.e. $f_{W_3}(x) = f_T(x) * f_T(x) * f_T(x)$. To specify $\Pr[W_k \leq R]$, we can use a similar approach. Let $Y_k = W_k - R$, then $\Pr[W_k \leq R] = \Pr[Y_k \leq 0]$. Since the pdf of $-R$ is $f_R(-x)$ and $f_{Y_k}(x) = f_{W_k}(x) * f_R(-x)$, we find that $\Pr[W_k \leq R] = \int_{-\infty}^{0} f_{Y_k}(x)dx$. In general, the series of convolutions can be efficiently determined numerically using fast Fourier transforms. The number of terms in the sum needed to closely approximate $E[M] = \sum_{k=0}^{\infty} k \Pr[M = k]$ is typically in the order of 10.

5.3.4. EVALUATION OF THE MEAN-FIELD EQUATION (5.3) AND $E[M]$

The mean-field equations (5.3) were compared with simulation results. Every time a susceptible node becomes infected, an exponentially distributed curing time R is drawn and a curing event for the newly infected node is scheduled at that time. In addition to the curing time R, an infection time T is drawn from a Weibull distribution (5.1) for each neighbour. Only if the infection time T is smaller than the curing time R, the infection event is scheduled. When the infection event is processed, a new infection time is drawn and, if the infection time is smaller than the curing time of the node that scheduled the infection event, a new infection event is scheduled.

To find the meta-stable fraction of infected nodes in GSIS, we log the exact percentage of simulated time t_i that the network has spent in a state with i nodes infected and compute the average $\bar{y}(t)$ at time t as $\frac{1}{N}\sum_{j=0}^{N} j\, t_j$, the standard deviation of the fraction of infected nodes is calculated as $\frac{1}{N}\sqrt{\sum_{j=0}^{N} t_j (j - \bar{y}(t))^2}$. We prevent the infection process from dying out by reinfecting the last remaining infected node immediately after it is cured; the elimination of the absorbing state of the GSIS process is called the modified GSIS, similarly as in [49]. To be certain that the process is indeed in the *steady-state* of

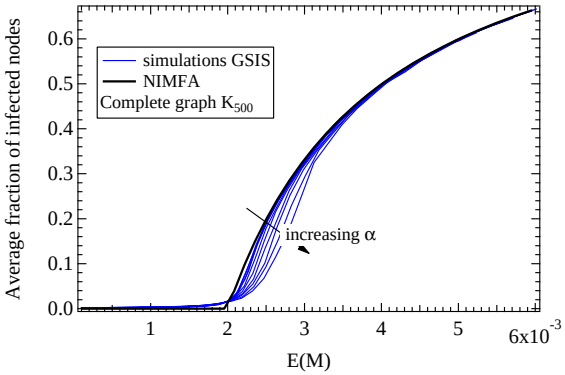


Figure 5.7: The average meta-stable state fraction of infected nodes versus $E[M]$ in the complete graph with $N = 500$ nodes.

the modified GSIS (corresponding to the *meta-stable state* of the GSIS process), we run two simultaneous but independent simulations on the same network and compare the time averages after an initial phase of 100,000 state changes. One process starts with 10% of the nodes infected, whereas the other process starts with all nodes infected. We conclude that the steady-state is reached when the difference between the time averages of the two simulations is small, i.e. $\frac{|\bar{y}_1(t)-\bar{y}_2(t)|}{\bar{y}_1(t)+\bar{y}_2(t)} < 10^{-4}$. The steady-state fraction of infected nodes is then given by $\frac{\bar{y}_1+\bar{y}_2}{2}$. To avoid very long simulation times for larger values of α we set a time limit of 20 minutes simulation time per run.

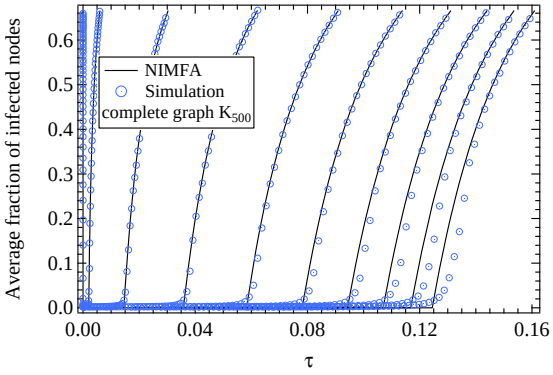


Figure 5.8: The average meta-stable fraction of infected nodes versus τ in the complete graph K_{500} for various α ranging from 0.5 (left) to 5 (right) with steps of 0.5.

Just as in NIMFA, which was shown to be a lower bound of the epidemic threshold, the mean-field equations (5.3) of GSIS are also lower bounding the epidemic threshold in our simulations illustrated in Figure 5.7 and Figure 5.8, that both contain the same

data. It is important to stress, however, that in general, the mean-field equations of GSIS are not upper bounding the steady-state faction of infected nodes, and as a result they could also over-estimate the epidemic threshold. Nonetheless, Figure 5.7 emphasises that $E[M]$ is the more natural parameter, instead of the ratio $\tau = \frac{E[R]}{E[T]}$, because all curves tends to each other, indicating that $E[M]$ constitutes a proper scaling.

When the infection time T tends to a deterministic time (i.e. $\alpha \to \infty$), the variance in the simulations increases, so that more realisations need to be simulated for increasing α. Still, the mean of the simulation seems nicely upper bounded by NIMFA. While we have concentrated here on the comparison based on the complete graph (for which NIMFA is exactly available [29] and the average meta-stable state fraction of infected nodes is given by $y_\infty(\tau) = \frac{1}{N}\sum_{j=1}^{N} v_{j\infty} = 1 - \frac{1}{(N-1)\tau}$), the same agreement is found on Erdős-Rényi graphs and Barabasi-Albert graphs as illustrated in Figure 5.10. For lattices, on the other hand, Figure 5.10 shows (as also found previously, see e.g. Figures 2.8, 5.3, and 5.5) that mean-field approximations are less accurate. Finally, also GSIS epidemics on the complete graph K_N with two non-exponential distributions for T and R follows the same behaviour, as shown in Figure 5.7.

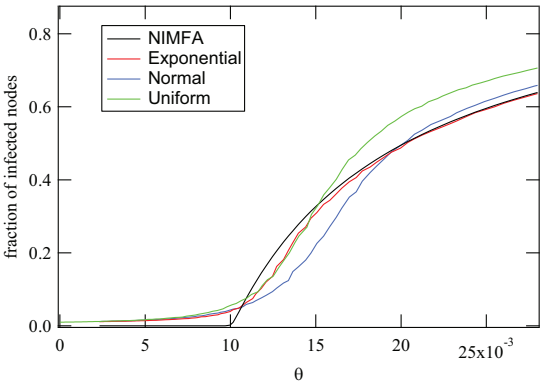


Figure 5.9: The fraction of infected nodes in a complete graph of 100 nodes as a function of the expected number of infection attempts during an infectious period θ for different infection and curing distributions: exponential, normal and uniform. The NIMFA solution is indicated in black.

The accuracy of the NIMFA approximation seems to depend on the distributions of the curing and infection inter-arrival times. Figure 5.9 shows the fraction of infected nodes in the steady-state of the modified SIS process (the absorbing state is removed) for different distributions of the inter-arrival times of both the curing and infection processes. The NIMFA solution is shown in black. The curves for the three simulations all start at 0.01 as a result of the absorbing state being removed. For exponential inter-arrival times, the fraction of infected nodes is well approximated by the NIMFA solution. For increasing values of $E[M]$, the fraction of infected nodes is almost equal to the NIMFA solution, and, crucially, lies below the NIMFA solution. NIMFA is proved in [55] to upper bound the infection probability, but only in the Markovian SIS process. For uniformly distributed inter-arrival times, however, the fraction of infected nodes follows the expo-

nential case until it reaches about 25%. The fraction of infected nodes for higher values of $E[M]$ lies above the NIMFA solution. This shows explicitly that the NIMFA solution in the case of non-Markovian spreading is not an upper bound, as also mentioned in [55]. For normal (Gaussian) inter-arrival times, the fraction of infected nodes is lower than the exponential case for smaller values of $E[M]$ and slightly above the NIMFA solution for larger values of $E[M]$.

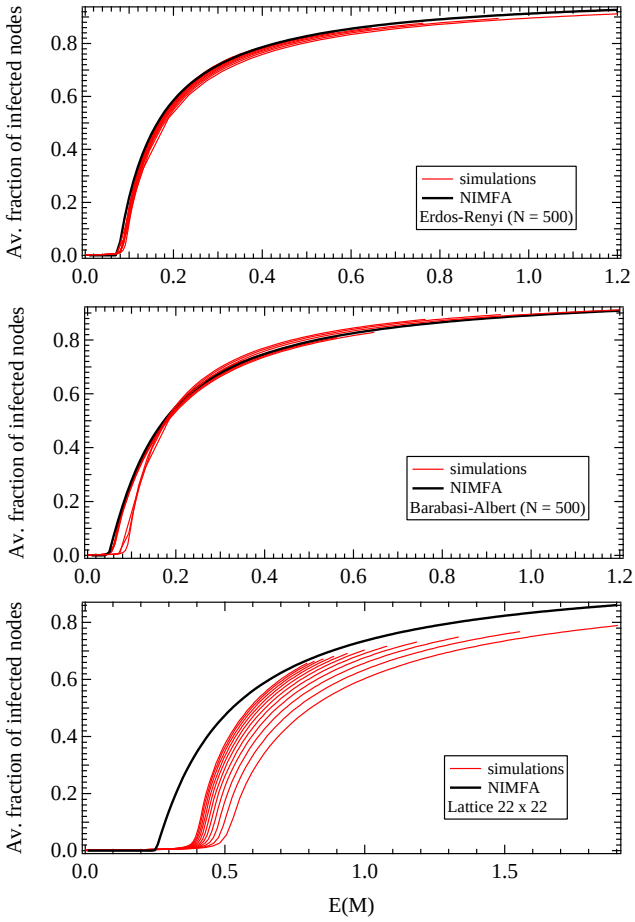


Figure 5.10: The average fraction of infected nodes versus $E[M]$ for three different graph types (with $N = 500$).

5.4. NON-MARKOVIAN SURVIVAL TIME

The exact results for the complete graph K_N and the star graph $K_{1,N}$ that underpin the results in Section 3.3 are only attainable because the infection and spreading processes are independent Poisson processes. The analysis of non-Markovian epidemic spreading

is complicated by the absence of the memoryless property of the exponential distribution. Yet, it has received attention in literature [87–89, 93]. In this section, we show the influence on the survival time of a non-Poissonian infection process. The NIMFA approximation of the infection probability in the meta-stable state is still valid, as derived in Chapter 5.3,

$$E[M](1-\nu_{i_\infty})\sum_{j=1}^{N} a_{ij}\nu_{j_\infty} = \nu_{i\infty},$$

where $E[M]$ is the expected number of infection attempts during an infectious period of a node, a_{ij} are the components of the adjacency matrix A of the graph and ν_{i_∞} is the probability that node i is infected in the meta-stable state. In the case of exponentially distributed inter-arrival times between infection and spreading events, $E[M]$ is given by $\tau = \frac{\beta}{\delta}$, while the general expression for $E[M]$ is deduced in Chapter 5.3.3. The NIMFA steady-state infection probability ν_{i_∞} is, however, no longer an upper bound, as shown in [55] and further illustrated in Chapter 5.3.4. The average survival time $E[T]$ for non-Markovian SIS processes is deduced by simulations.

In order to investigate the effect of heavy-tailed distributions for the infection times on the survival time, we replace the exponential distribution with a Weibull distribution given by

$$f_T(x) = \frac{\alpha}{b}\left(\frac{x}{b}\right)^{\alpha-1} e^{-(x/b)^\alpha}, \tag{5.6}$$

where α is generally called the shape parameter and b the scale parameter. For $\alpha = 1$, the Weibull distribution reduces to the exponential distribution. The shape of the Weibull distribution as a function of α is shown in the inset of Figure 5.12a.

The difference between the Markovian and non-Markovian SIS process is most visible in the distribution of the infection attempts during an infectious period of a node. Even when the average number of infection attempts during an infectious period $E[M]$ is kept constant, the steady-state fraction of infected nodes and the survival time of the process change because of the timing of infection attempts relative to the curing time of a node. In this section, we only change the distribution of the infection events, and keep the cure time of a node exponentially distributed.

Figure 5.11 shows the simulated time distribution of infection attempts over a single link, normalised to the infectious period of the source node; every time an infection attempt takes place in our simulations, we log the infection time relative to the beginning of the infectious period of the node and normalise that to the duration of the infectious period of the node. For the exponential distribution, it is known [53] that if an event has happened during an interval that the event time is uniformly distributed, but for other distributions this is not the case. Indeed, in Figure 5.11 the exponential case ($\alpha = 1$) is uniform, as expected, but for $\alpha < 1$ events tend to be early, and for $\alpha > 1$ late.

The SIS process only stays in a network for a long time if nodes become reinfected after they cure. Therefore, the timing of the infection attempts has a great influence on the behaviour of the process. When the virus spreads from a source node early in its infectious period, it is less likely that the newly infected node will reinfect the source node after it cures, because the newly infected node is also likely to spread early and the infectious state of the source node has most likely not changed yet. On the other hand, if the source node spreads the infection towards the end of its infectious period, it is likely

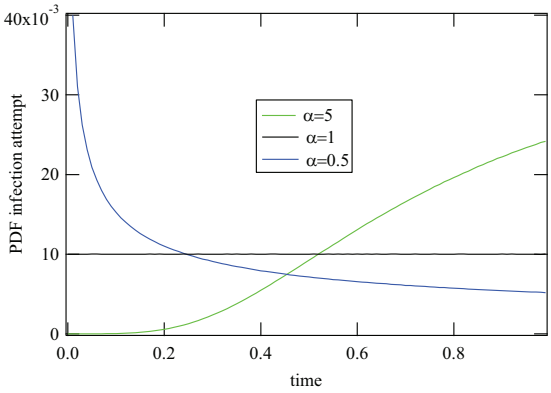
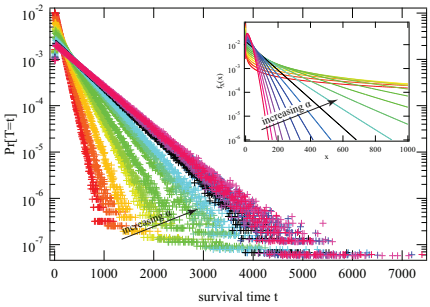


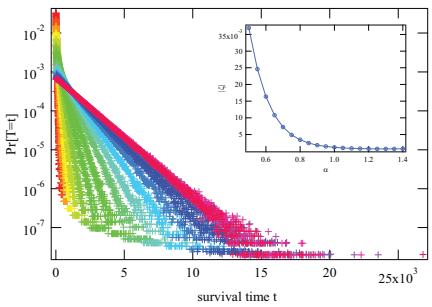
Figure 5.11: Distribution of the infection attempts over a link normalised to the infectious period of the source node.

to have cured (and become reinfected) if the newly infected node spreads the infection back.

Figure 5.12a shows the survival time distribution for α ranging from 0.5 to 1.5 and a constant value for the expected number of infection attempts during an infectious period $E[M] = 0.014$. For small values of the shape parameter α, the survival time distribution falls off much quicker than for the exponential case (shown in black). This is the result of the early infection attempts. The virus does not succeed in reinfecting nodes that have cured and dies out quicker. For larger values of α the opposite happens. The survival time distribution falls off slower than in the exponential case.

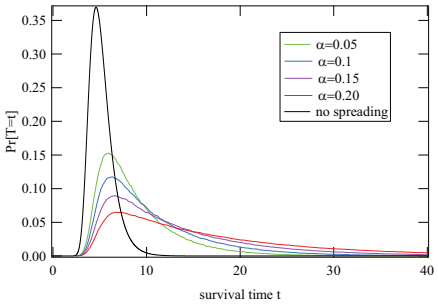


(a) Distribution of the survival time for Weibull distributed inter-arrival times for various values of the shape parameter α in the complete graph of 100 nodes. The exponential case ($\alpha = 1$) is indicated in black.

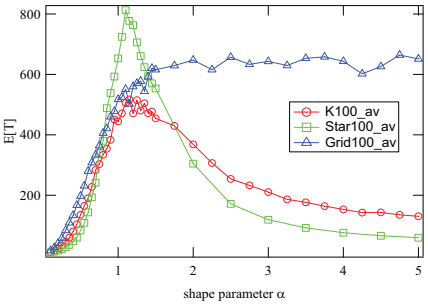


(b) Survival time distribution in a star graph of 100 nodes for different values of the shape parameter α. Scale parameter β is chosen such that on average 25% of the nodes are infected in the meta-stable state.

Figure 5.12: Survival time as a function of the shape parameter α in the complete graph.



(a) Survival time distribution in the complete graph of 100 nodes for small values of the shape parameter α. The scale parameter b is chosen such that $E[M]$ is constant.



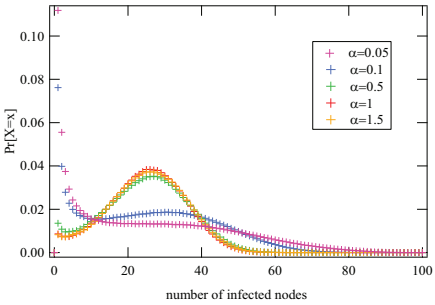
(b) Average survival time as a function of the shape parameter α in the complete graph, the star graph and the square lattice, all with $N = 100$ nodes. The scale parameter b is chosen such that on average 25% of the nodes are infected in the meta-stable state.

Figure 5.13

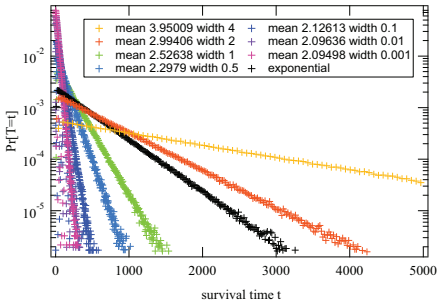
The same influence of the shape parameter α on the average survival time that is observed in the complete graph is also visible in the star graph, as shown in Figure 5.12b where the survival time distribution for various values of α is shown in a star graph of 100 nodes. The inset shows the exponent of the tails as a function of the shape parameter α. In this case, we have kept the steady-state fraction of infected nodes constant at 25% by changing the scale parameter b in (5.6) appropriately.

The effect of very small values for the shape parameter α is shown in Figure 5.13a, where the survival time distribution is shown for a complete graph of 100 nodes for various values of α. The black curve indicates the survival time of the virus in the case that the infection rate is zero, in which case nodes can only cure. Since all nodes are infected in the initial state, the black curve shows the the probability distribution (3.14) of the maximum of 100 exponential i.i.d. random variables with rate δ. Figure 5.13a illustrates that, for small values of the shape parameter α, the distribution of the survival time resembles that of the no spreading case. This is because all spreading attempts happen shortly after infection, as shown in Figure 5.11, after which all nodes cure again.

The effect of relatively large values for the shape parameter α, on the other hand, is shown in Figure 5.13b for three different graph types: the complete graph, the start graph and the square lattice. The scale parameter b in (5.6) is chosen so that the average number of infected nodes in the meta-stable state is 25%. For small values of α, the expected survival time of the virus is short, as mentioned above. As the shape parameter α increases, so does the expected survival time. However, the expected survival time peaks around $\alpha = 1.1$ for the star graph and complete graph, and then starts to reduce. This might be caused by a form of synchronisation in the infection times. The larger the shape parameter α becomes, the narrower the infection time distribution becomes. As a result, infections occur in an increasingly small interval after the source node becomes infected. When the infectious periods of the nodes are aligned in time, the chances that



(a) The average number of infected nodes in a square grid of 100 nodes for various values of the shape parameter α and a scale parameter b chosen in such a way that the average is constant. The virus is prevented from dying out.



(b) Survival time distribution of the SIS process with uniformly distributed cure and infection times for different values of the width of the infection rate. The mean of the infection rate is scaled to have on average 25% of the nodes infected in the meta-stable state. The exponential case is shown in black for reference.

Figure 5.14

the virus dies out increases. This is especially possible in the star and complete graph because the very small average hop count. In the case of the square grid, the shortening of the survival time with increasing α is not observed; the expected survival time does not peak but stabilises. Figure 5.13b illustrates that the peak is much sharper in the star graph than in the complete graph.

The influence of a small shape parameter α as described above on the distribution of the number of infected nodes in the meta-stable state is shown in Figure 5.14a, which shows the distribution of the number of infected nodes for various values of the shape parameter α. To simulate the meta-stable distribution of infected nodes, we simulate a modified SIS process where the absorbing state is removed and take the steady-state of that process to be the meta-stable state of the original SIS process. For smaller values of the shape parameter α, the distribution peaks around a single infected node and is relatively flat after that peak. States with a large number of infected nodes are not unlikely, indicating that the virus quickly spreads as a result of the nodes spreading early during the infectious period, and then dies out again.

To show the effect of both late infection times and synchronisation, we use uniformly distributed cure and infection times. The width and mean of the cure times is kept constant, but the width of the infection time distribution is varied while the mean is scaled to keep the average fraction of infected nodes y_∞ constant. Because both the curing and infection times are uniformly distributed, the interval in which infection attempts can fall is easily identified as between the start of the infection distribution and the end of the curing distribution. For the values used in this section, that means an interval for infection attempts between $[1.95, 2.1]$ and $[2.09, 2.1]$. Figure 5.14b shows the survival time distribution in the complete graph of 100 nodes for various values of the width of

uniform infection time distributions. The average survival time distribution also has an exponential tail when *both* the infection and the curing times are non-exponential. Figure 5.14b shows that the late infection times lead to longer survival times than for exponential infection times (shown in black), but that for the narrower intervals the expected survival time is shorter. For the narrowest width of the infection time distribution, any infection attempt lies in an interval only 0.01 time units wide, which leads to synchronised behaviour. Although late infections lead to longer expected survival times, synchronised spreading leads to shorter infection times.

5.5. CHAPTER SUMMARY

The significant effect of a non-exponential infection time on the average steady-state fraction $y_\infty(\tau)$ of infected nodes and, thus on the epidemic threshold, questions the huge efforts to precisely compute the epidemic threshold in Markovean SIS epidemics in networks if viruses in real epidemics (or in computer networks) do not infect in an exponential time. Unfortunately, it appears exceedingly difficult to measure accurately the infection time in real epidemics to verify the exponential assumption, made in almost all earlier SIS computations. From interactions with epidemiologists, rare measurements [94] and email activity [95], it seems quite likely that the infection time is not exponentially distributed so that our observations here may point to a complete revision of SIS epidemics on networks. If, however, we use the expected number of spreading attempts during an infectious period as the defining parameter of the process, the average fraction of infected nodes in the meta-stable state is relatively stable regardless of the shape parameter of the Weibull distribution. Moreover, the NIMFA equations are shown to be still valid if the effective infection rate is replaced by the expected number of infection attempts during an infectious period. In contrast to the average fraction of infected nodes in the meta-stable state, the survival time of the non-Markovian SIS process is influenced by the inter-arrival time distribution. The non-uniform distribution of infection events over the infectious period of a node either shortens or prolongs the survival time. If the infection attempts are highly concentrated towards either the beginning or the end of the infectious period, the survival time will be shorter than in the Markovian case.

6

GOSSIPICO: AN EPIDEMIC ALGORITHM

Mimicking, to a certain extent, the spread of a virus or infection through a network, we propose GOSSIPICO*, an epidemic algorithm to average, sum or find minima and maxima over node values in a large, distributed, and dynamic network.* GOSSIPICO *provides a continuous estimate of, for example, the number of nodes, even when the network becomes disconnected.* GOSSIPICO *converges quickly due to the introduction of a beacon mechanism that directs messages to an autonomously selected beacon node. The messages propagate along near-shortest paths towards the beacon. The proposed algorithm allows every node to trigger a reaction to network changes, which makes it robust against network dynamics without having to resort to periodic restarts. Simulations in various different network topologies (ranging in size up to one million nodes) demonstrate* GOSSIPICO*'s robustness against network changes and display a near-optimal count time. One of the distinguishing features of* GOSSIPICO *is that it contains a convergence detection mechanism that enables nodes to know the network size with certainty.*

6.1. INTRODUCTION

Just as a virus spreads from individual to individual, information spreads from person to person. The most recognisable variant of the uncontrolled (and maybe even unwanted!) spread of information is that of gossip. The key feature of gossip is that information usually spreads by means of repeated unicast: one unlucky person tells a secret to a "friend" and that friend cannot resist the urge to tell it to one of their friends, and before long the entire community is privy to the secret. In a sense, the secret is an infection that spreads through the community. In a similar fashion as gossip spreads through a community, the nodes in a computer network can communicate amongst each other.

Several developments over the past years, such as the growing use of peer-to-peer overlay networks and sensor networks have led to the deployment of very large decentralised networks. Although decentralised networks are very scalable, they have no cent-

ral point where information is stored, which makes it challenging to gather global network properties or perform coordinated actions.

One of the paradigms that has emerged to spread and gather information in a fully decentralised network, is that of "gossipping" [96]. During the gossip process, a node periodically selects one of its neighbours and either sends, requests or exchanges information with that neighbour. This simple communication structure can be used to perform various tasks in decentralised networks, ranging from overlay building and information dissemination to calculating functions such as sums and averages.

In this chapter we investigate averaging and summation over node values in large dynamic networks, and in particular the specific case of counting the number of nodes. Good estimates of the size of a decentralised network can be valuable in optimising the performance of protocols and services that run on top of it, such as topology building in a peer-to-peer network. Node counting can also provide information on how many sensors are still working in a sensor network or how large an ad-hoc network currently is. The latter can be useful in vehicular communication to estimate traffic conditions.

Algorithms specifically designed to estimate or count the number of nodes in a network can be roughly divided into three groups: probabilistic polling algorithms, random walk based algorithms, and gossip-based algorithms [97]. We highlight two representative examples of the first two techniques, after which we overview gossip-based algorithms. The probabilistic polling technique proposed by Kostoulas *et al.* [98] uses a conditional reply to a request message to estimate the network size, where the condition is determined by the distance between the replying node and the initiating node. In the algorithm by Massoulié *et al.* [99], a node sends a message containing an initial counter value on a random walk, and each node that is passed on the walk adds a degree dependent value to the counter. When the random walk returns to the initiating node, it can estimate the network size based on the counter value.

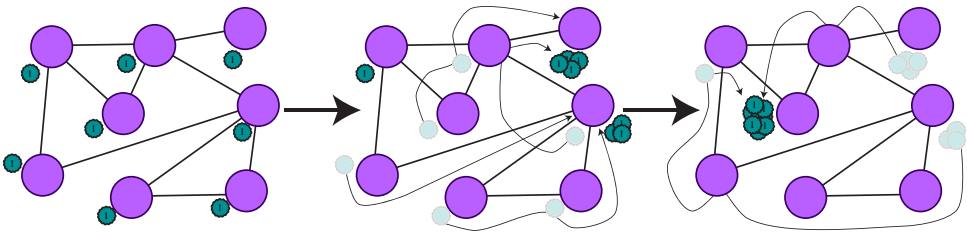


Figure 6.1: Three snapshots in the counting process.

The gossip-based algorithms in [100, 101] rely on averaging an initial value over all nodes to estimate the network size. Jelasity and Montresor [101] initialise the node values to 0, with the exception of a single "special" node that has value 1. Nodes find the estimate of the network size as the reciprocal of the average. Montresor and Ghodsi [100] first build an overlay structure based on random node values to determine an initial guess of the network size at each node. That initial guess is subsequently averaged to improve the estimate. In the work of Kempe *et al.* [102], each node has a weight-value pair and periodically sends half of its weight and value to both a random neighbour and itself. Nodes sum the pairs they receive and estimate the network size by dividing the

summed value by the summed weight. By assigning a 1 or 0 to the initial weight of one or all of the nodes, the algorithm either sums or averages. Finally, Guerrieri *et al.* [103] studied the performance of both an averaging-based gossiping algorithm and three variations on a token collecting algorithm in delay-tolerant networks.

A major drawback of the random walk and probabilistic polling strategies is the poor accuracy of a single walk or poll and the subsequent need of averaging over many runs [97]. Size estimation based on averaging using gossip, however, currently also has its drawbacks. First of all, the process has to be started by a single node, which requires some sort of coordination in the initial phase of the algorithm. Moreover, the estimate of the network size at a specific node can vary widely, and joining or leaving nodes cause large local variations in the estimate that have to be spread out over the network. More extreme network dynamics such as the joining of two networks will result in an estimate that is half the new network size, whereas splitting the network in two will lead each part to believe it consists of the original number of nodes.

Our goal is to develop a gossip algorithm that does not share the drawbacks that averaging-based algorithms have, and can be used in continuously changing networks, even when the network becomes disconnected. In our design, we aim for little processing and storage at each node. Also, all nodes are uniform in capabilities and function, i.e. no node is special. The only requirements for the nodes are that they have a unique identifier and can reliably pass messages to their direct neighbours.

6.2. GOSSIPICO

We propose GOSSIPICO, a gossip-based algorithm that does not rely on averaging to estimate the network size, but combines messages to count the number of nodes. Message combining offers much flexibility in algorithm design, and allows for the simultaneous execution of several functions such as summation, finding maxima, and averaging in a single algorithm.

GOSSIPICO consists of two parts: COUNT and BEACON. COUNT performs the actual counting of the nodes, while BEACON is used to speed up COUNT. We first introduce the basic ideas behind COUNT and BEACON in Section 6.2.1 and 6.2.2, respectively, and will further develop those in Section 6.2.3 where we add robustness against dying nodes and disconnecting networks. In Section 6.3.1 we propose a convergence detection scheme.

Both in COUNT and in BEACON nodes *initiate* communication periodically. The time period during which all nodes have initiated communication with one of their neighbours exactly once is called a gossip cycle. Although a node initiates communication only once during a gossip cycle, it can be contacted by more than one neighbour during a single cycle. The time between gossip cycles in a deployment scenario will be a trade-off between bandwidth consumption and speed.

6.2.1. COUNT

COUNT can be seen as an improvement on the token collection principle. In such a token collection scheme, all nodes are initially given a single token. Each cycle, every node contacts one of its neighbours and gives that neighbour all of its tokens. As time progresses, all tokens will end up in one pile at a particular node. This idea is illustrated

in Figure 6.1, where, in the left-hand side picture, every node has a single token. In the centre picture, tokens start to accumulate at different points in the network. Note that the sum of the tokens in the three piles is still the same as the number of tokens we started with. In the right-hand side picture all tokens have piled up and the centre node now knows how many nodes there are in the network.

The token collecting idea has three major drawbacks. First, the pile of tokens has to be passed on from node to node until all nodes have had possession of the pile of tokens in order for all nodes in the network to know how many tokens there are. Second, since the piles of tokens make random walks through the network, it will take a very long time for all the tokens to pile up. Third, although the centre node in the little example has all the tokens, it cannot know that it does: there is no convergence detection. GOSSIPICO addresses all these problems: the first drawback will be addressed in COUNT, whereas the second will be so in BEACON, and finally GOSSIPICO as a whole will contain convergence detection.

The slow spread of the number of tokens in the final pile in a token collection scheme is solved in COUNT by gossipping the number of tokens in the pile from node to node using information spreading messages. Nodes should only gossip the latest information that is locally known, so they need a way to tell which information is most recent. As they also need to make the distinction between the gossiping messages and the equivalent of the tokens, the exchanged messages in COUNT are more complex than tokens. A message $M = \{C, F, T\}$ contains a count value $C \in \mathbb{Z}$, a freshness value $F \in \mathbb{N}$, and a type value $T \in \{0, 1\}$. The type value indicates whether the message is an Information Spreading (IS) ($T = 0$) message, or an Information Collecting (IC) ($T = 1$) message. Information Collecting messages can be seen as the tokens in Figure 6.1. The count value C is the current estimate of the network size and has a subtle difference in meaning in the two types of messages. In an IS message, the count value represents the number of tokens in the latest pile that has been seen locally, whereas in an IC message the count value represents the number of tokens in the pile that is represented by that IC message. The freshness value F is a measure for the recentness of the information in the message: the message with the highest value for F contains the most recent information.

Every node that joins the network is initialised with an IC message with count and freshness value 1, i.e. $M = \{1, 1, 1\}$, analogous to the single token in Figure 6.1. Nodes send their message periodically to one of their neighbours and always create a new IS message afterwards to overwrite the message that was sent. Messages are processed by the receiving node by following one of four rules, based on the type and the freshness of both the received message, M_r, and the receiving node's current message, M_w (subscripts r and w stand for received and waiting, respectively). The rules govern both the spreading of IS messages and how IC messages are combined. By changing how IC messages are combined, the algorithm can be made to count both upwards and downwards, average or find minima and maxima. The four processing rules are explained below:

1 ($T_r = 0, T_w = 0$) If both messages are IS messages, the received message will replace the waiting message if the received message has a higher freshness value F than the waiting one. By keeping the message with the freshest information, only the most recent information is spread through the network.

2 ($T_r = 1, T_w = 0$) A received IC message will always replace a waiting IS message. IC messages should never be deleted, otherwise the total number of tokens in the network decreases.

3 ($T_r = 0, T_w = 1$) A received IS message will be discarded if the waiting message is an IC message. This again ensures that no collected information is lost.

4 ($T_r = 1, T_w = 1$) If both messages are IC messages, a new message will replace the waiting message. The new message is the result of combining the two IC messages. In the case of counting: the waiting and received messages' C and F values are summed, i.e. $M = \{C_w + C_r, F_w + F_r, 1\}$.[1]

Rule number 2 forces nodes to discard IS messages when an IC message is received, regardless of the freshness of the information in the IC message. It can happen that a node's estimate of the network size decreases temporarily when it is forced to discard an IS message. In order for nodes to be able to create a new IS message that contains the fresher (but discarded) information after it passed on this IC message, nodes keep two state values: a count value $C_s \in \mathbb{Z}$, and a freshness value $F_s \in \mathbb{N}$ (subscript s stands for state). After processing a received message, the receiving node updates its state values C_s and F_s if the new waiting message contains fresher information. The state information is used to create new spreading messages that contain the most recent information. These spreading messages are always created after a node has sent its waiting message when it was its turn to initiate contact.

A message's freshness value will only differ from the count value in dynamic networks or when the algorithm is used to sum or average. When the algorithm is used in dynamic settings, the freshness value could run out of its range, but this can be fixed by allowing nodes to accept messages with a freshness value much lower than their F_s as fresher, if F_s is close to the maximum, or, alternatively, to trigger a recount using the mechanisms described in Section 6.2.3.

6.2.2. BEACON

The second drawback of a token collecting scheme that is solved in GOSSIPICO is that of the long convergence time. Instead of passing on IC messages randomly, nodes guide them towards each other to speed up the counting process. IC messages have a much higher chance of meeting each other when they are guided to a particular node in the network. We propose BEACON to guide IC messages towards each other by using a beacon. A beacon is a node whose location information spreads through the network by means of gossip.

The beacon node is not known beforehand, but is the result of a competition among the nodes. Every node starts out as a beacon, and competes for dominance with the other nodes. The competition is designed in such a way that eventually there will be only one beacon. As an analogy we describe the competing nodes as leaders or members of

[1] Nodes should be initiated with a message $M_w = \{V, 1, 1\}$ to sum over node values V instead. Finding a minimum or maximum value is achieved by not summing the values but keeping the highest or lowest value, whereas averaging requires an extra field in the message where the number of summed values are counted to allow the average to be calculated as V/C.

armies. Every node j belongs to an army A_j that has a certain strength $S_j \in \mathbb{R}$; node j also knows which neighbour P_j is the first hop towards the beacon of its army, and the estimated hopcount H_j to that beacon. The state of node j is represented by the vector $B_j = \{A_j, P_j, H_j, S_j\}$

Initially, every node is the leader of its own one-node army that has a randomly chosen strength, i.e. $B_j = \{j, j, 0, \text{random}\}$.[2] Nodes periodically and randomly select one of their neighbours to fight with. The outcome of such a fight is determined by two rules:

1 If the nodes are in the same army, the shortest path to the beacon is updated.

2 If the nodes are not in the same army, one must be stronger than the other, and the weaker node is incorporated into the army of the stronger one. The losing node takes over the values for A and S from the winning node, sets the winning node as the next hop to the beacon, and sets the estimated hopcount to one more than the estimated hopcount of the winning node.

Following these two rules, the strongest node in the network will defeat all other nodes and become the only beacon. At the same time, the paths to the winning beacon are continuously updated. Every node continuously receives new information about which neighbour can reach the beacon in the fewest hops and selects that neighbour as the first hop to the beacon. The hop sequences towards the beacon converge to shortest paths along which IC messages can combine.

IMPROVED NEIGHBOUR SELECTION

BEACON's neighbour selection, as discussed in Section 6.2.2, is completely random. To speed up the beacon competition, the neighbour selection can be adapted to avoid repeated contacts to the same neighbour before the other neighbours have been contacted. In the improved neighbour selection, nodes keep their neighbours in a list and every time they communicate with a neighbour (either because they initiate the communication or because they are contacted by one of their neighbours) they move that neighbour to the bottom of the list. When they select a neighbour to fight with, they take the node at the head of the list. We use the improved neighbour selection in the simulations testing convergence detection, where its effect is maximal.

6.2.3. NETWORK DYNAMICS

GOSSIPICO, as outlined so far, can cope very efficiently with a growing network: new nodes simply initialise their COUNT message at $M_w = \{1, 1, 1\}$ and their BEACON state as $B_j = \{j, j, 0, \text{random}\}$, and start communicating. Node (and link) removals, however, are more challenging. A node can leave gracefully by sending an IC message with a count value of -1 ($M_w = \{-1, 1, 1\}$) to account for its departure, but when a node dies, it cannot send this message. Moreover, even when a node leaves the network gracefully, there is no guarantee that the network will remain connected after the departure. If the network indeed becomes disconnected, a restart of the counting process must be triggered.

[2] We assume perfect randomness, i.e. $\Pr[S_j = S_k] = 0$, for $j \neq k$. In a practical setting where perfect randomness is not achievable, the recount trigger described in chapter 6.2.3 can be used to break the tie when two different armies with equal strength meet.

In order for nodes to be able to trigger a recount, the counting process is restricted to work only within the BEACON armies. In terms of the token collecting analogy: nodes only give their tokens to nodes that are in the same army. As a result, nodes use COUNT to count the size of their *army* instead of the network. When a node is incorporated in a new army, it forgets everything it knew about the old army and starts with a fresh IC message, i.e. $M_w = \{1,1,1\}$. As the growing army conquers the network, the network is counted. Since nodes count the size of their army, a recount can be triggered by allowing a conquered node to build a new army that will defeat the reigning one.

GOSSIPICO copes with dying nodes and disconnecting networks by following a single rule: every time a link is removed, the nodes adjacent to the link rebuild their armies to trigger a recount. In case the link removal did not disconnect the network, the two new armies will compete for dominance and the network is recounted. In case the link removal disconnects the network, the two revived armies will both survive, since they cannot reach each other to compete. The algorithm is unaffected by network dynamics such as node and link addition or removal, because every node can trigger a recount in response to node or link removals.

Two concepts have to be added to BEACON to support recounts: *army revival* and *immunity*. Army revival allows a node j that is conquered by another army ($A_j \neq j$) to build its own army again, by selecting a new random strength S_j and setting itself with a zero distance as the first hop to the beacon ($P_j = j, H_j = 0$). The newly revived army of node j ($A_j = j$) will also be immune ($I_j = k$) to its previous army k. The immunity ensures that all nodes of army j will win all fights with nodes of army k. Without immunity, the recount will be suppressed if the new army j is weaker than the reigning army k.

Although GOSSIPICO is robust against dying nodes and other network dynamics using recounts, a node's count value drops during the recount process. Dips do not last long, and can even be avoided at the expense of a slight delay if nodes estimate the network size based on their estimate before the recount and the current count value. If dips are to be avoided, a node's estimate X is calculated as $X = (1 - f(t))X_{old} + f(t)C_s$, where X_{old} is the node's estimate at the time it was incorporated into a new army and $f(t)$ is a logistic function that shifts the current estimate from the previous estimate to the current count value. The logistic function is given by $f(t) = 1/(1 + \exp(-t + 2H + 5))$ where t is the number of times the node created the same IS message. The constant 5 is used to make sure that $f(t)$ starts at a value close to 0. The distance H to the beacon ensures that high diameter (and therefore slower converging) networks automatically adjust to the recount time without the need of a tuning parameter. If the value of C_s is larger than X_{old}, nodes accept C_s as the best estimate. In all simulations of GOSSIPICO in Section 6.4 we avoid dips.

6.2.4. INTERACTION BETWEEN COUNT AND BEACON

With the general outline of COUNT and BEACON in place, we will now explain in detail how the two parts of GOSSIPICO interact. The interactions between COUNT and BEACON are illustrated in Figure 6.2: COUNT uses the next hop (P) to the beacon in BEACON when an IC message is sent, and the army (A) information when receiving messages; BEACON resets the count process when a node is conquered by another army. The precise interactions will be discussed using the pseudo-codes of both COUNT and BEACON.

Algorithm 1 COUNT

Init: $M_w \leftarrow \{1,1,1\}$

Sending
1: **if** $T_w = 1$ **then**
2: select peer P
3: **else**
4: select random peer
5: **end if**
6: send message M_w to peer
7: $M_w \leftarrow \{C_s, F_s, 0\}$

Receiving
8: **if** $A_r \neq A$ **then**
9: return M_r to sender
10: **return**
11: **end if**
12: **if** $T_r = 0$ & $T_w = 0$ **then**
13: **if** $F_r > F_w$ **then**
14: $M_w \leftarrow M_r$
15: **end if**
16: **end if**
17: **if** $T_r = 1$ & $T_w = 0$ **then**
18: $M_w \leftarrow M_r$
19: **end if**
20: **if** $T_r = 1$ & $T_w = 1$ **then**
21: $M_w \leftarrow \{F_w + F_r, C_w + C_r, T_w\}$
22: **end if**
23: **if** $F_w > F_s$ **then**
24: $F_s \leftarrow F_w$
25: $C_s \leftarrow C_w$
26: **end if**

resetCount
27: $C_s \leftarrow 1$
28: $F_s \leftarrow 1$
29: $M_w \leftarrow \{1,1,1\}$

A message consists of three fields: C, F, and T, i.e. $M = \{C, F, T\}$. Subscripts r, w and s indicate received, waiting and state, respectively.

The pseudo-code for COUNT is given in Algorithm 1. Lines 1-6 ensure that if a node wants to send an IC message ($T = 1$), and it is not the beacon of its own army, it will send the message to the next hop towards the beacon of its army. In all other cases, the node will send its waiting message to a random neighbour. After sending a message, a new waiting message is created in line 7. Lines 8-11 make sure that nodes only accept messages originating from their own army. If the message is indeed from within the army, it is processed according to the four rules in Section 6.2.1: lines 12-16 execute rule 1, rules 2 and 3 are executed in lines 17-19, and lines 20-22 form rule 4. After processing the message, the node's state values are updated in lines 23-26 if the new information is fresher than the current information. Finally, lines 27-29 are used to reset a node's state

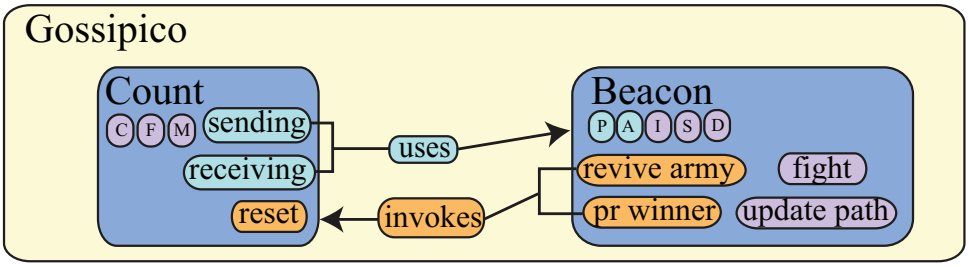


Figure 6.2: Interactions between COUNT and BEACON.

Algorithm 2 BEACON

```
    Fight(Node i,j)
 1: if I_i = A_j then
 2:     prWinner(i,j)
 3:     return
 4: end if
 5: if I_j = A_i then
 6:     prWinner(j,i)
 7:     return
 8: end if
 9: if A_i = A_j then
10:     updatePath(i,j)
11:     return
12: end if
13: if S_i > S_j then
14:     prWinner(i,j)
15: else
16:     prWinner(j,i)
17: end if

    updatePath(Node i,j)
18: if H_i < H_j + 1 then
19:     P_i ← j
20:     H_i ← H_j + 1
21: end if
22: if H_j < H_i + 1 then
23:     P_j ← i
24:     H_j ← H_i + 1
25: end if

    prWinner(Node i,j)
26: A_j ← A_i
27: S_j ← S_i
28: I_j ← I_i
29: H_j ← H_i + 1
30: P_j ← i
31: resetCount                    ▷ in Count

    reviveArmy(Node i)
32: S_i ← random integer
33: H_i ← 0
34: P_i ← i
35: I_i ← A_i
36: A_i ← i
37: resetCount                    ▷ in Count
```

and message in case it is incorporated in a new army. These lines can be invoked from BEACON.

Algorithm 2 shows the pseudo-code for BEACON. The first thing that is checked in a fight is whether one of the nodes is immune to the army of the other node, as can be read from lines 1-8. If this is the case, the node that is immune will win the fight and the losing node takes over the strength, army and immunity from the winning node (lines 26-30), sets the winning node as the first hop towards the beacon (lines 29-30), and resets its COUNT process (line 31). If both nodes are of the same army (lines 9-12), the shortest path to the beacon is updated by checking whether either node can reach the beacon quicker via the other (lines 18-25). If both nodes are from different armies, the stronger one will incorporate the weaker one (lines 13-17). The "revive army" code can be called in response to a network change. The army revival process involves determining a new random strength, setting the army id equal to the node id, setting the immunity equal to the present army, setting the distance to the beacon to zero, and resetting the COUNT process (lines 32-37).

6.3. CONVERGENCE OF GOSSIPICO

The analytical discussion of GOSSIPICO'S speed of convergence is based on the bounds on the spreading time of a single rumour in a network by Giakkoupis [104]. In order for

these bounds to be applicable, we divide the node counting process into three stages, although in reality these stages happen at the same time. In the first stage, the armies fight for dominance to establish a single beacon. In the second stage, the nodes send their IC messages along the beacon paths until they have all been combined. Finally, in the third stage, the result of combining all the IC messages spreads through the network. All three stages can be bounded by the spreading time of a single rumour in a network, which Giakkoupis [104] bounds as $O(\phi^{-1}\log(N))$, where N is the number of nodes in the network and ϕ the conductance.[3] The conductance depends both on the network type and on N. It is a variation of the isoperimetric constant and closely related to the algebraic connectivity [30].

During the first stage, the beacon information of the strongest army spreads unhindered through the network, just as a rumour in Giakkoupis' work. During the second stage, the IC messages are routed towards the beacon. Most messages combine on their way to the beacon, but in the worst case they combine at the beacon. Since the beacon information was spread in at most $\phi^{-1}\log(N)$ rounds, the longest path to the beacon cannot be longer than $O(\phi^{-1}\log(N))$ hops. During the third stage of the algorithm, a single piece of information, i.e. the count value of the last IC message, spreads to all other nodes by means of gossip; this is exactly the same process as rumour spreading. Simulations in the following section show that GOSSIPICO has an $O(\log(N))$ convergence in Erdős-Rényi random graphs and scale-free graphs.

6.3.1. CONVERGENCE DETECTION

GOSSIPICO will always converge in static networks and, by the virtue of the recount process, also in dynamic networks that display network dynamics that are slower than the convergence time. As nodes only have a local view of the network, they need a strategy to distributedly detect that there is only one IC message left. They need to know that the process has converged.

Convergence detection in GOSSIPICO relies on two observations. First, convergence can only occur when a single army dominates the entire connected component. Second, nodes can detect whether all their neighbours are within the same army.

Definition 1. *Let a border-free node be defined as a node who's neighbours all belong to the same army.*

GOSSIPICO with convergence detection consists of three layers: one BEACON layer and two COUNT layers. In the first COUNT layer the number of nodes is counted, as described in Section 6.2.1, whereas in the second COUNT layer the number of border-free nodes is counted. We will call the layer that counts nodes the count layer and the layer that counts border-free nodes the convergence layer. Since every node is initially a border node, all nodes are initialised with an IS message in the convergence layer.

In order for border-free nodes to be counted, nodes first have to find out whether they are border-free or not. As nodes exchange their army membership every time they

[3] $\phi = \min_{S\subseteq\mathcal{N},\, \sum_{i\in S} d_i \leq |\mathcal{L}|} \frac{\text{cut}(S,S^c)}{\sum_{i\in S} d_i}$, where $\text{cut}(S,S^c)$ is the set of links crossing a partition of a graph $G=(\mathcal{N},\mathcal{L})$ in S and S^c, with $\mathcal{N}$ the set of nodes, $\mathcal{L}$ the set of links, and d_i the degree of node i.

fight on the BEACON layer, they can simply keep a single flag for each neighbour, signalling that that neighbour is within the same army or in an unknown army. The improved neighbour selection discussed in Section 6.2.2 will be of great advantage in convergence detection because every node needs to have had contact with all its neighbours before it knows it is border-free and the neighbour selection prevents neighbours to be contacted twice before all others have been contacted.

When a node has discovered that all its neighbours are within the same army, it creates an IC message on the convergence layer to signal that it is border-free. The actual convergence detection is left to the beacon and is achieved by comparing the count values on both counting layers. If we assume that IC messages on both counting layers follow the same route to the beacon, the following inequality holds at the beacon $C_s^{con} \leq C_s^c$, where C_s^{con} is the state count value on the convergence layer and C_s^c is the state count value on the count layer. Equality is only reached when there are no border nodes left in the network and, hence, the counting process has converged.

Although unlikely, it is possible that a path update to the beacon occurs between the sending of an IC message on the count layer and the convergence layer for some node. To avoid false convergence detection caused by IC messages arriving out of order, the beacon will have to observe a short delay before concluding the process has converged. This delay is bounded by the maximum hop count from any node to the beacon, which can be found if nodes include this information in the IC messages.

In dynamic networks, the beacon can revoke convergence by creating an IS message that is fresher than the converged message but without the convergence flag. Nodes will now be aware that the latest information is not necessarily the final value. Nodes can also adjust their border-free status by sending an IC message with count value -1 on the convergence layer when they discover they are no longer border-free.

6.4. SIMULATION RESULTS

We have performed various simulations to test the convergence and dynamic behaviour of GOSSIPICO. Time is measured in gossip cycles. During each gossip cycle, the nodes are processed sequentially and in a random order. The count time of the network is defined as the number of gossip cycles that pass until every node in the network reaches a count value equal to the network size, and is averaged over a number of runs. When we use Erdős-Rényi random graphs, the probability p that a link exists is set to twice the critical link probability: $p = 2\log(N)/N$; scale-free graphs are generated following the preferential attachment model in Batagelj and Brandes [105], with such a number of links per new node that the total number of links is closest to that in an Erdős-Rényi graph with the same number of nodes and a link probability of twice the critical value. We will first discuss the rate of convergence in static networks. We then discuss the dynamic behaviour of GOSSIPICO.

6.4.1. COUNTING IN STATIC NETWORKS

We illustrate the impact of BEACON on the performance of COUNT, by comparing the performance of GOSSIPICO to that of COUNT alone. Since COUNT can be seen as an extension of randomised token forwarding algorithms, these simulations also illustrate by

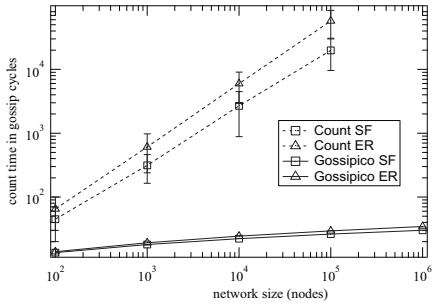


Figure 6.3: Count time for GOSSIPICO and COUNT. ER indicates an Erdős-Rényi graph, SF a scale-free graph.

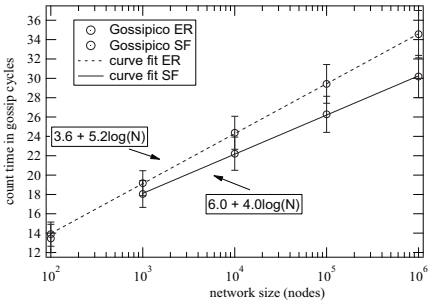


Figure 6.4: Count time for GOSSIPICO in an Erdős-Rényi graphs and scale-free graphs.

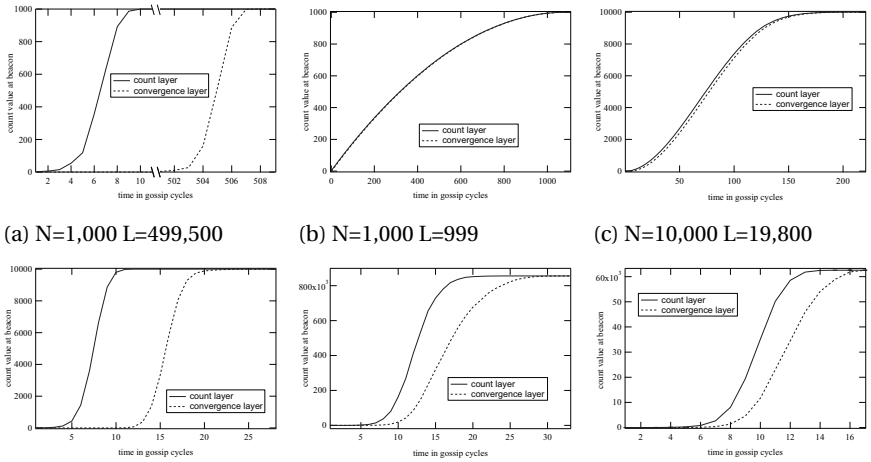
how much BEACON might speed up other algorithms. Simulated network sizes range up to 10^6 nodes, and for every size a hundred different networks were generated.

Figure 6.3 shows the count time for both GOSSIPICO and COUNT as a function of the network size for the two graph types. As expected, GOSSIPICO greatly outperforms COUNT on its own, as can be seen from Figure 6.3. The count time of COUNT scales as a power function of N, whereas the count time of GOSSIPICO scales logarithmically. The logarithmic scaling of GOSSIPICO is illustrated in Figure 6.4. This plot is based on an average over 500 runs per size and shows an error band at ± one standard deviation. The count time in GOSSIPICO can be fitted with $3.6 + 5.2\log_{10}(N)$ for the random graphs and $6.0 + 4.0\log_{10}(N)$ for the scale-free graphs. For random graphs the fit goes through all data points from $N = 100$ nodes and upwards, whereas for scale-free graphs the fit starts at $N = 1000$ nodes, because the scale-free degree distribution only emerges for larger networks. Both COUNT and GOSSIPICO perform better in scale-free graphs than in random graphs; the hub nodes in scale-free graphs function as meeting places where IC messages are combined. Because BEACON ensures that GOSSIPICO scales logarithmically in random and scale-free networks, COUNT and similar (token collecting) algorithms are made suitable for use in very large networks.

6.4.2. PERFORMANCE OF CONVERGENCE DETECTION

The performance of GOSSIPICO'S convergence detection is quantified as the delay after the actual convergence of the counting process and the detection of convergence by the beacon. Since each node has to learn the army membership of all its neighbours, it is expected that convergence detection is slower in dense networks than in sparse ones. We track the count value at the beacon on both the count layer and the convergence layer for networks with varying link densities. The values are averaged over 1000 runs for the synthetic networks and 100 runs for the real-world networks. The results for all graphs are grouped in Figure 6.5.

Figures 6.5 a and b show the count value at the beacon on both count layers for two extreme graphs in terms of density: the complete graph (a) and the path graph (b). Counting the number of nodes in the complete graph is quick, but it takes $\frac{N}{2}$ cycles be-



(a) N=1,000 L=499,500 (b) N=1,000 L=999 (c) N=10,000 L=19,800

(d) N=10,000 L=89,928 (e) N=855,802 L=4,291,352 (f) N=62,561 L=147,878

Figure 6.5: Count value at the beacon in the count layer and convergence layer for **(a)** a complete graph, **(b)** a path graph, **(c)** a rectangular grid, **(d)** a scale-free graph, **(e)** the Google web graph, and **(f)** the Gnutella graph.

fore the first IC message is created on the convergence counting layer. For the path graph the situation is completely opposite; counting is slow, but when the counting process has finished, convergence detection is almost immediate.

For other graph types similar behaviour is found. In Figure 6.5 the count values at the beacon for a Scale-Free graph (d) and Grid (c) are also shown. Again, it takes less time to count the denser graph but convergence detection suffers from a larger delay. For the grid graph the count value on the convergence layer follows the count value on the node counting layer with a small delay. We also included two real network traces in our simulations: a trace of the Gnutella P2P system [106] and a web graph released by Google as part of the Google programming contest 2002.[4] For both graphs we first remove the directionality of the links, and then extract the largest connected component to use in our simulations. For real-world networks, as shown in Figures 6.5 e and f, the delay in convergence detection is not too large, illustrating that in practical situations convergence can be detected quickly.

6.4.3. COUNTING IN DYNAMIC NETWORKS

In order to illustrate GOSSIPICO's robustness against node failure and to show that no node is indispensable, we track the average estimate of the network size while the node currently functioning as beacon is removed every $40th$ cycle. When a node is removed, all links to the node and the node's state information are also removed.

We start out with both Erdős-Rényi random graphs and scale-free graphs ranging from $N = 1000$ to $N = 100,000$ nodes. In Figure 6.6 the average estimate of the network size over time is shown. Figure 6.6 illustrates that the average estimate of the network

[4] Both datasets can be found in the large network dataset collection, part of the Stanford Network Analysis Project (http://snap.stanford.edu/)

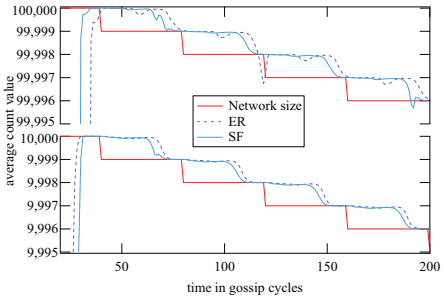


Figure 6.6: Average estimate of the network size during a targeted beacon node attack for both ER and SF graphs and two different initial sizes.

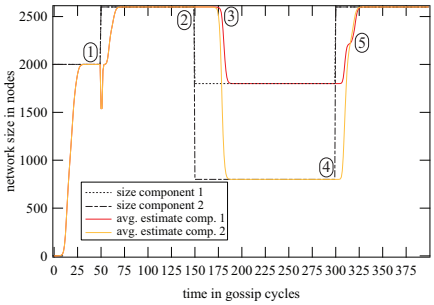


Figure 6.7: Average estimate of the network size after random node additions (1), targeted link removals leading to disconnectedness (2,3), and reconnecting (4,5).

size follows the real network size. During the 25 cycles after the beacon dies, the network is recounted and the nodes shift their estimate from the previous count value to the new one to prevent a drop in the estimate. Even when the beacon node dies before the network is recounted, the average count value does not fluctuate too much as can be seen in the case of the 100,000 node network (the upper graph in Figure 6.6). This shows that GOSSIPICO can keep up with dynamics with a rate of change close to, or marginally over, its own speed of convergence without introducing large fluctuations in the estimate.

We next show GOSSIPICO's behaviour in fully dynamic and disconnecting networks. We start by creating two random graphs, one of 1500 nodes (component 1) and one of 500 nodes (component 2), and connect them by adding 10 links between randomly selected nodes in each component. After fifty cycles we add 300 nodes to each component, all new nodes only link to nodes in their own components. After another 100 cycles the 10 links connecting the two components are removed, and are added again after another 150 cycles.

The average estimate of the network size over time is plotted in Figure 6.7. The figure shows that the maximum count time for the initial 2000-node network over all the runs in this simulations is 33 cycles, regardless of the bottleneck of 10 links connecting the two components.

The addition of the 300 nodes per component (event 1 in Figure 6.7) shows that new IC messages combine along near shortest paths. All IC messages combine, on average, within 7 cycles, while the average hopcount in these bridged networks is 4.2. After another 14 cycles, the spreading process finishes. The narrow dip (which only lasts for 1 cycle) in the estimated network size that is visible immediately after the addition of the new nodes is the result of the initial estimate of 1 of all new nodes.

Removing the bridging links between the two components (event 2) has a similar impact on the average estimate as a single dying node. The estimate falls (event 3) rapidly after an initial delay of about 25 cycles, without fluctuating. Regardless of the fact that 20 nodes trigger a recount at the same time, the estimate of the component size smoothly approaches the correct value in both components.

After reconnecting the two components (event 4), the estimate of the network size is updated just as quickly as a normal count of the network would be, as can be seen by comparing the first 25 cycles in Figure 6.7 to the first 25 cycles after the components are reconnected in cycle 300. The increase in the smaller component is continuous, while the estimate in the larger component displays a bend (event 5) half way. This bend can be explained by observing that in half of the cases the army in the smaller component will dominate the network, and in the other half of the cases the army in the larger component will do so. When the larger component assimilates the smaller, the nodes in the smaller component will immediately accept the new count value as the new estimate because it is higher than their previous count value. In the opposite case, the nodes in the larger component will hang on to their previous count value a little longer. In either case, the counting process continues without a restart as soon as the two components are connected.

Figures 6.6 and 6.7 illustrate that GOSSIPICO smoothly tracks all dynamics in the network, ranging from node additions to dying nodes and network components disconnecting and reconnecting again. Instead of relying on a periodic recount, GOSSIPICO reacts to changes in the network. This ensures that in periods of network stability the estimate of the network size is also stable, while during periods of network dynamics, the algorithm tracks the changing network size.

6.4.4. MAXIMUM COUNT VALUE OVER TIME

In this section we discuss the maximum count value in the network as a function of the normalised number of sent messages. The product of the number of nodes and the expected hopcount $E[H]$ is taken as the normalisation constant. We simulate four different network topologies, an Erdős-Rényi random graph, a scale-free graph, a square grid, and a path. For each network type the highest count value in the network after every interaction between two nodes is averaged over 500 different realisations, both for $N = 1,000$ and $N = 10,000$. As an approximation of the average hopcount of both an Erdős-Rényi graph and a scale-free graph we use $\log(N)$, while for the grid and path graph we use the approximation for a regular lattice $E[H] \simeq \frac{d}{3} N^{1/d}$, where d is the dimension of the lattice (for a grid $d = 2$, for a path $d = 1$) [107].

Figure 6.8 shows that the normalised number of sent messages before the maximum count value is reached, is of the same order of magnitude for all four networks. The main difference between the networks lies in the slope of the curves. In random and scale-free graphs, the maximum count value follows an S-curve that is more pronounced for larger networks, while in the grid and path graphs the maximum count value increases more steadily, and shows little influence of the network size. The difference between the grid and path graphs on the one hand and the random graphs on the other, is probably best explained by the speed at which the winning army can grow. The winning army in the path graph will grow at a fixed rate because the configuration at the borders of the army never changes. A growing army will always have two nodes at its endpoints that are neighbours with different armies. In the case of a square grid, the nodes in the winning army roughly form a circle, and the army grows at an increasing rate proportional to the circumference of that circle. The increase in growth rate, however, is stable.

For the random and scale-free graphs the situation is quite different. With every ad-

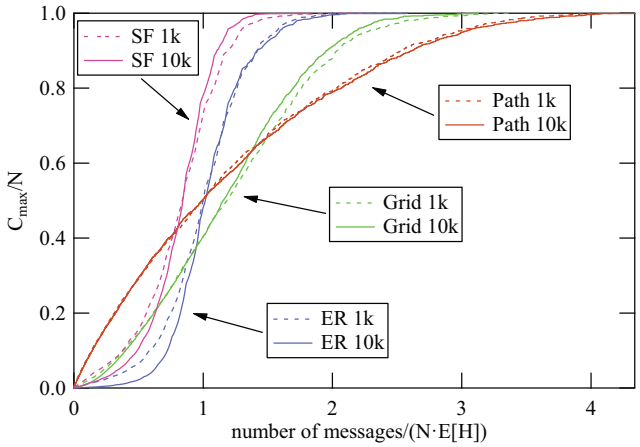


Figure 6.8: Maximum count value as a function of the normalised number of sent messages for four graph types.

dition of a node to the winning army, the percentage of nodes not in the winning army that can be reached by the winning army can increase rapidly, thus allowing the army to grow explosively. After a brief initial phase, the number of nodes in the winning army undergoes a phase-transition and covers almost the entire network. Although the average hopcount is the most important factor in the count time for all four network types, the speed at which an army can grow determines the slope of the graph.

6.5. CHAPTER SUMMARY

We propose a gossip algorithm, GOSSIPICO, to count the number of nodes in a network (or sum/average over node values). Our algorithm works by combining messages, which has an advantage over averaging-based counting algorithms in that the estimate approaches the network size quicker and more smoothly. GOSSIPICO uses only $O(\log(N))$ messages per node to count Erdős-Rényi random graphs and scale-free graphs, while randomised token forwarding based counting algorithms use $O(N^{\alpha})$ messages per node, with a power exponent $\alpha > 0$. The count time closely follows the average hopcount for grids and path graphs, which matches the lower bound $O(\text{diam}(G))$.

A major improvement, besides speed, over previous algorithms is that GOSSIPICO automatically restarts the counting process when a change is detected that could lead to disconnectedness in the network. When two components are joined, the algorithm converges to the correct count without a restart, which is impossible for algorithms based on averaging. Simulations show that GOSSIPICO is a very robust algorithm that provides nodes with a continuously updated estimate of the network size at a speed of convergence that equals that of rumour spreading, which is known to be very fast. Finally, GOSSIPICO comes with fully working convergence detection, enabling nodes to distributedly and precisely count the number of nodes, and be assured that the count value is correct.

7

NETWORK EXTRACTION, ANALYSIS AND MANIPULATION

Online Social Networking (OSN) applications such as Facebook's communication and Zynga's gaming platforms service hundreds of millions of users. To understand and model such relationships, social network graphs are extracted from running OSN applications and subsequently processed using social and complex network analysis tools. Here, we focus on the application domain of Online Social Games (OSGs) and deploy a formalism for extracting graphs from large datasets. Our formalism covers notions such as game participation, adversarial relationships, match outcomes, and allows to filter out "weak" links based on one or more threshold values. Using two novel large-scale OSG datasets, we investigate a range of threshold values and their influence on the resulting OSG graph properties. We discuss how an analysis of multiple graphs—obtained through different extraction rules—could be used in an algorithm to improve matchmaking for players.

7.1. INTRODUCTION

An increasing number of complex network studies and social network use graphs to represent data. Often, two extreme approaches for building a graph from raw data are deployed. At one extreme, graphs are straightforwardly extracted when the raw data specifies links, such as in the case of friendship relationships for Facebook data. At the other extreme, graphs are extracted by applying a single, domain-specific and usually threshold-based, rule for mapping raw data to links. The latter approach has been used, for example, for building the graphs of mail and email exchanges of various communities and organisations, of messaging in Twitter and the Microsoft Messenger Network, of hyperlinking in the Web and in the blogosphere, etc. The impact of the choice of graph extraction rules and thresholds has received much less attention and constitutes the focus of this study.

We focus on the social networks formed by the actions of players involved in online social gaming (OSG), that is, in online gaming where the social element affects positively

the gameplay experience. Online social games such as Defense of the Ancients (DotA) and League of Legends are each played by tens of millions of gamers. The game is fragmented into hundreds of thousands of non-communicating instances (matches [108]), and groups of only about ten players are involved in any instance of the game at any one time. Players can find partners for a game instance through the use of community web sites and online tools, which may include services that matchmake players to a game instance. An interesting feature of OSGs, as opposed to many online social networks, is that not only friendship relations are formed, but also adversarial relationships may manifest as useful social relationships.

Within the application domain of online social games, few studies use network metrics to divide players into different classes. For example, Kirman and Lawson [109] extract a network from an online game by creating an unweighted and undirected link between players that ever exchanged information in the game. They define three types of players based on the successive removal of the highest-degree nodes until the largest connected component falls apart. Shim et al. [110] define implicitly a network that is used to predict future player performance based on the relationship between mentors and apprentices. This analysis could be extended by looking at network-wide properties instead of only local properties. The prediction of the success in games can also be applied to real-world games as is done by Vaz de Melo et al. [111], where a complex network approach is used to predict the performance of basketball teams. The authors propose a network-based ranking of players as a replacement for current statistics such as assists and points scored to predict the future success of a team.

Other studies use different definitions of links to create different networks from the same dataset. Szell and Turner [112] study a detailed dataset of interactions and friendship and enemy relations spanning three years in the online game Pardus, which is much less popular than the communities we investigate in this study. They use the game as a substitute of the real-world and test several hypotheses in the field of social dynamics such as social balancing, network densification and triadic closure. They study three different networks extracted from the dataset: the network of communication between players, the network of friends as indicated by players in the game, and the network of enemies as indicated by players in the game. Although the authors present a detailed analysis of the networks for each type of interaction, and especially contribute to existing work by analysing the network of enemy relations, their links are either interaction based or explicitly indicated by the players.

A related study [113] on guild members in World of Warcraft also investigates the differences between networks formed based on different types of interaction. In this work social network analysis is used to explore the network structure of interactions between guild members in the online game World of Warcraft. The authors studied 76 players that formed a single guild and extracted networks by creating links between players that communicate amongst each other. Different types of interaction were classified into seven different categories such as asking for help or group management to form seven different networks. An analysis of these networks in terms of reciprocity and topological structure indicates that the different types of interaction lead to different networks. This study focusses, however, on a rather small group of players, which enables the authors to analyse and classify the communication between players. In our study, we analyse

large datasets by applying different mappings, instead of analysing the dataset to find the mappings.

In practice, it may be difficult to obtain a clear social structure from an online social game. Data that has proven economic value or private status, such as expressed friendships, are rarely shared by game operators; instead, third-parties may have access only to proxies, such as a list of players for each game instance. Furthermore, for OSG networks, traditional relationships such as expressed friendship relationships may not be a good indicator of joint activity. This may be a consequence of the classes of prosocial emotions involved in OSGs, for example vicarious pride and happy social embarrassment [114, Ch. 5], which are both derived from competition. These emotions complement other prosocial emotions that precede traditional friendship, such as admiration and devotion, and may require other tools for expression.

Without a clearly specified social structure, the analyst of OSG networks has to select and tune the *extraction rule*, that is, the rule for extracting graph links from *play relationships* recorded in the logs of completed and ongoing game instances. For example, it is common to extract OSG graphs through the simple rule of forming a link between gamers that have played at least once together. However, this simple rule overemphasises the importance of a single in-game encounter, which may have been casual or the result of an automated matchmaking mechanism. OSG networks derived using this rule have been recently investigated, for example for EverQuest [110], World of Warcraft [115], Fighters Club (a Facebook application) [116], etc. In contrast, we generalise this formalism for graph extraction.

We analyse the interplay between various concepts and thresholds that are used when defining extraction rules in OSGs, and the characteristics of the resulting OSG graphs. Based on two large OSG datasets, we study various rules and thresholds for extracting graphs. Using this approach, we are able to show that even small changes in the thresholds used for various rules, especially for the small values that have been used by previous OSG graph studies, can lead to significant changes in the structure and use of the resulting graph. Thus, our work provides the tools to obtain a broader picture of OSG graph analysis than previous work.

The results of OSG studies are useful for tuning the distributed systems on which the games operate, for improving the game experience, and for understanding the individual and group psychology of players. As an application of the proposed analysis technique, we study the quality of matchmaking players in game instances.

7.2. DATA SETS

We select for our study Defense of the Ancients (DotA), a free and popular game in which social relationships, such as same-guild membership and even friendship, can improve the gameplay experience. DotA is a 5-against-5-player game, that is, it consists of independent matches played by two contesting teams of five players each. DotA is a multiplayer online battle arena (MOBA) game, in which each player controls an in-game representation (here, the hero), and teams strive for the conquest of the opposite side's main building. The game includes many strategic elements, from team operation to the management of resources and the creation of helper troops.

DotA, which is the representative of the MOBA game sub-genre, is played today by

an estimated[1] number of players above 20 million, world-wide. DotA has featured in several tournaments with wide appeal to gamers and game-watchers, such as the World Cyber Games (WCG) and the Electronic Sports World Cup (ESWC). (To understand the phenomenon of watching DotA games, we recommend the seminal book on gaming culture of Rossignol [117].) Other successful MOBA games are Blizzard DotA, Valve's DotA2, League of Legends, and Heroes of Newerth. Each of these games is played by millions of players, which are loosely grouped into large communities. In turn, most communities operate their own game servers, maintain lists of tournaments and results, and publish information such as resulting player rankings via common websites.

The datasets we used were taken from the Game Trace Archive [118]. The DotaLeague dataset consists of 3,744,753 matches that were played between July 2006 and July 2011. The dataset contains, for each match, the names of the players (61,198 in total) on each team, the active time, the start and end times, various gameplay statistics per team, and the team that has won. The *active time* refers to the time at which the match opens and players can enter, while the *start time* refers to the time when sufficient players are present and the match begins. The *end time* refers to the time at which the match finishes. Not all intended matches take place. It can happen, for example, that not enough players join an active match to actually start it. Therefore, to make sure that the matches we analyse have indeed been played, we only use those matches that have a value start and end time. Although the match active time stamp is available for all matches, the match start and end time stamps were only available from November 2008 onwards, which corresponds to 1,470,786 played matches.

For DotAlicious, the dataset contains 625,692 played matches, which represents the complete set of matches played from April 2010 to February 2012. Each match entry in the dataset records the names of the players (62,495 in total) of that match, the countries from which they are playing, the result of the match (winning team, draw, or abort), the start and end times, various gameplay statistics per player, and the number of points obtained for each player. The DotAlicious dataset also contains matches that have not actually been played. They are identifiable by the fact that they start and end at the same time. We filtered out matches with a zero duration, obtaining 617,069 played matches.

To understand the general characteristics of our datasets, we conduct an analysis of the game activity they represent. Both datasets exhibit similar time patterns, along the lines of other games and of other typical Internet applications. The number of matches started per hour of the day shows a clear diurnal pattern in weekdays and weekends, as can be seen from Figure 7.1. In weekdays, the number of matches played per hour rises from 6AM onwards and reaches a peak around 9PM. In weekends, the number of matches played rises from 6AM and reaches a peak around 4PM, where it stabilises until 9PM, with only a small drop around dinner time. The average number of matches played per day of the week is fairly stable (approaching 1000 matches/day), with slightly more matches played in the weekends.

The two datasets are also similar with respect to the inter-arrival time of matches. The time between the consecutive matches of individual players is shown, for the DotAlicious dataset, in Figure 7.2. The time between matches shows a power-law-like behaviour, with many matches being separated by less than an hour, but also consecutive

[1] http://www.playdota.com/forums/blog.php?b=892

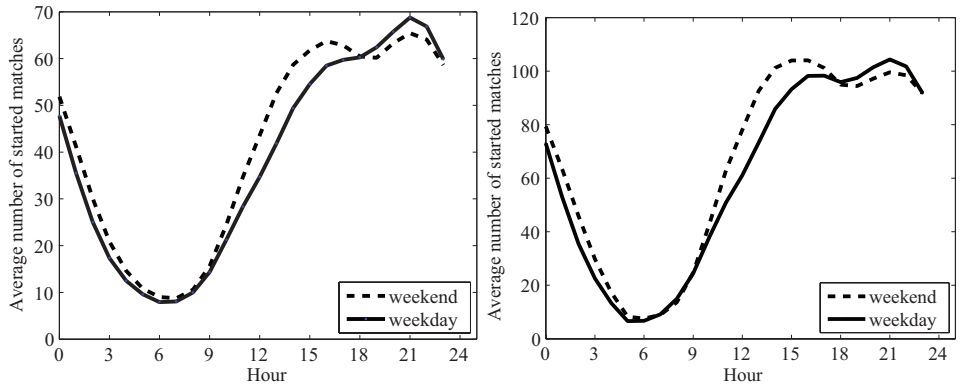


Figure 7.1: Average number of matches per hour for DotAlicious (left) and Dota-League (right).

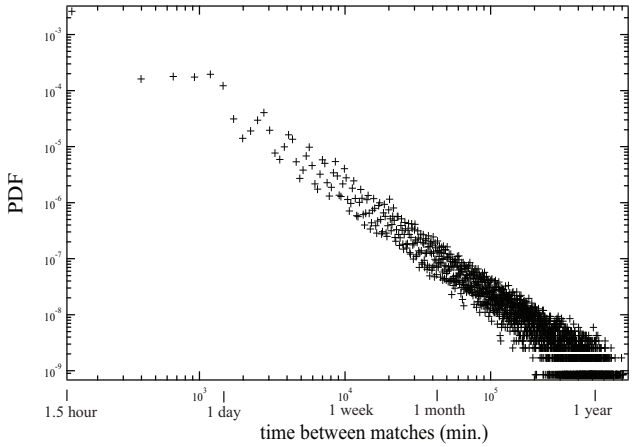


Figure 7.2: Probability density of the time between consecutive matches for each player in the DotAlicious dataset.

matches being separated by more than a year.

7.3. A FORMALISM FOR GRAPH EXTRACTION

A common approach in online social network studies is to model a dataset as a graph. Formally, a dataset D is mapped onto graph G via a mapping function $M(D)$. A simple undirected and unweighted graph $G = (\mathcal{N}, \mathcal{L})$ consists of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L links. In a *weighted* graph a link weight w is associated to every link in $\mathcal{L}$. In a directed network, a link between two nodes also has a direction.

A *mapping* is a set of rules that define the nodes and links in a graph. Entities are often mapped to nodes, while relations between entities are mapped to links. Entities are usually readily identifiable as persons, events, or objects and are therefore intuitively

mapped onto nodes. Mapping relations to links, however, is more challenging.

Entities can be related to each other in many different and often subtle ways and due care should be given to which relations produce insightful graphs. For example, some relations between entities may be the result of chance, while others have a clear origin. The difference between random and meaningful relations is often expressed by a notion of strength. Strength, represented by a link weight, adds another dimension of complexity to representing and understanding the characteristics of a network.

In the creation of many "real-world" graphs, often a single explicit mapping is used. For instance, all graphs in the Stanford Network Analysis Project's (SNAP)[2] large graphs collection are based on explicit relations in the source datasets with one exception [119], where graphs are built using the co-occurrence of phrases on news sites and blogs. For this exception, nodes (phrases) and links (co-occurrences) have to meet a set of criteria to be added to the graph, although the authors merely state which criteria they used and do not elaborate on the systematic effects of these criteria.

In this chapter we explore how the mapping influences the resulting graph. Since a dataset usually comprises different types of information, e.g. location, time, etc., mapping to only one graph might misrepresent the dataset, unless clearly put in perspective. We consider multiple graph representations. Although we use the gaming datasets as examples, the techniques developed here are generally applicable.

Presently, many proposed complex network metrics (see, for example, Section 7.4.1) only apply to unweighted graphs. As a result, relations are often only expressed as links if the strength is within a desired range by applying a threshold. *Thresholding*, therefore, is a powerful and popular tool to deal with weighted networks and has an important impact on the resulting graph. For our OSG datasets, the individual players in the dataset are always mapped to nodes. Nodes without adjacent links are removed from the extracted graph. We explore six different strategies to map play relationships to links. For each strategy, a link in the extracted graph corresponds to a different type of gaming relationship between two players, which is likely not expressed directly in the raw (input) data. Each mapping includes a threshold n; as a consequence, each mapping may produce a different graph per threshold value.

By applying different mappings and different thresholds to the dataset, we investigate how a particular mapping and/or threshold affects the resulting graph. We have used the following mappings:

SM: The number of times two players are in the *Same Match* is greater than n.

SS: The number of matches played on the *Same Side* is greater than n.

OS: The number of matches played on *Opposing Sides* is greater than n.

ML: The number of *Matches* played and *Lost* together is greater than n.

MW: The number of *Matches* played and *Won* together is greater than n.

PP: This mapping is a directed version of the other five mappings. In the PP mapping, a *directed* link exists from player A to B if player A has played at least n% of all his matches (either on the same team, or opposing team etc.) with player B.

[2] http://snap.stanford.edu/

The different mappings are related to each other. For example, applying mapping function SM to the Dota-League dataset with a (low) threshold $n = 10$ creates graphs such that for two players p_1 and p_2 a link between them is formed if p_1 and p_2 occur in at least 10 different matches, while applying the SS mapping adds the extra condition that the players played on the same side. Our formalism can support more complex mappings, e.g., players played against each other at least 10 times, during the winter, while located in the same country.

7.4. THE FORMALISM IN PRACTICE: AN ANALYSIS OF THE EXTRACTED GRAPHS

In this section, we study the structure of the graphs formed by using different mappings. As each mapping extracts a different type of relationship between players, it is interesting to know whether these relationships give rise to significantly different graph structures. The various mappings can help finding different properties of the two seemingly similar datasets (see Section 7.3). We compare the obtained graphs using a broad selection of graph metrics. A detailed explanation of (and where to download) the tools we have developed to carry out the extraction and analysis presented in this chapter can be found in Appendix A.

7.4.1. GRAPH METRICS

To compare the graphs extracted using different mappings, we calculate a number of network metrics for the largest component of each extracted graph. The selected metrics all reflect properties related to the degrees and paths between players, and allow us to study the social relations in the gaming community. We also include a spectral metric. The metrics used in this section are explained in the following. For a more in-depth explanation we refer to [120, 121].

Size(s) of the connected component(s) (N, L): The size of the largest and other connected components indicates how many fellow players a player can reach in the network.
Link density (d): The link density is obtained by dividing the number of links in the network by the possible number of links $\binom{N}{2}$, and indicates how densely connected the network is. For directed graphs the density is obtained by dividing the number of links by $2\binom{N}{2}$, as every link can be in either direction.
Degree distribution: The degree distribution characterises the number of direct neighbours a node has.
Algebraic connectivity: The algebraic connectivity is the second smallest eigenvalue of the Laplacian matrix[3]. This is a spectral graph metric that indicates how well connected a graph is.
Average hop count ($\bar{h}$): The average hop count indicates how many hops, on average, players are removed from each other.
Diameter (D): The diameter is the longest shortest path, in terms of hops, in the network.

[3]The Laplacian matrix is obtained as $L = D - A$, where D is a diagonal matrix of the node degrees and A the adjacency matrix.

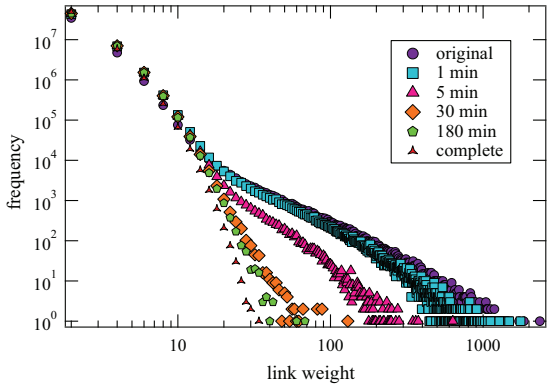


Figure 7.3: Frequency of occurrence of link weights in the reference model for five different intervals and the original data for DotA. The mapping is SM.

Average clustering coefficient ($\bar{C}$): The average clustering coefficient is a measure for how many neighbours of a node are also neighbours of each other.
Betweenness centrality (B): The betweenness centrality score of a node indicates on what fraction of shortest paths it is present and is a measure of node importance.
Coreness (c): A node of coreness k has at least k neighbours with at least k neighbours.
Assortativity (ρ): The assortativity measures to what extent nodes link to other nodes with similar degrees.

7

In addition to focussing on the largest component, we further analyse the effect of *thresholding* (both on the link weight and the play percentage for the PP mapping) on the size and number of network components. As thresholding removes random and weak relations between players; graphs extracted with a high threshold value will reveal the strongest social structures. Among the graph component representations of these social structures, many may be of similar size, without a clear largest component, while the collection of those components offers insight into strong social ties.

7.4.2. GRAPH ANALYSIS

In this section, based on different mapping functions and their thresholds, we analyse the size and social structure of the graphs extracted from the two OSG datasets. We compare the results for Dota-League and DotAlicious to indicate that, while not seemingly different for one general mapping, different mappings can reveal fundamentally different graph characteristics, and as such should be considered by the careful practitioner to fully capture the various facets of any data set.

METHODOLOGICAL REALITY-CHECK

Before analysing the extracted graphs, we have to be sure that the relationships we identify are not the result of players being simultaneously online by chance. To answer this

	DotA-League						DotAlicious					
	SM	OS	SS	ML	MW	PP	SM	OS	SS	ML	MW	PP
N	31,834	26,373	24,119	18,047	18,301	29,500	31,702	11,198	29,377	22,813	21,783	34,523
N_{lc}	27,720	19,814	16,256	6,976	8,078	33	26,810	10,262	20,971	10,795	13,382	3,239
L	202,576	85,581	62,292	30,680	33,289	53,514	327,464	92,010	108,176	43,240	54,009	125,340
L_{lc}	199,316	79,523	54,186	17,686	21,569	120	323,064	91,354	99,063	29,072	44,129	17,213
d ($\times 10^{-4}$)	4.00	2.46	2.14	1.88	1.99	0.62	6.52	14.7	0.49	1.66	2.28	1.05
d_{lc} ($\times 10^{-4}$)	5.19	4.05	4.10	7.27	6.61	1,100	8.99	17.4	2.51	4.99	4.93	16.4
μ	0.0301	0.0114	0.0060	0.0040	0.0032	-	0.0385	0.0403	0.0120	0.0095	0.0194	-
$\bar{h}$	4.42	5.40	6.30	8.09	7.67	3.70	4.24	3.97	5.3	6.80	5.95	18.45
D	14	21	24	28	26	9	17	12	19	20	22	74
$\bar{C}$	0.37	0.40	0.41	0.41	0.41	-	0.43	0.27	0.47	0.47	0.49	-
ρ	0.13	0.26	0.25	0.27	0.28	-0.10	0.08	0.01	0.25	0.27	0.29	0.20
B_m	0.04	0.09	0.09	0.17	0.12	1.21	0.03	0.05	0.04	0.06	0.06	0.37
c_m	85	55	41	19	22	3	131	68	48	16	20	7

Table 7.1: Metrics for $n = 10$: for MOBA games: the Dota-League and DotAlicious datasets. The metrics we present: number of nodes N, number of nodes in largest connected component N_{lc}, number of links L, number of links in largest connected component L_{lc}, link density d, link density of largest connected component d_{lc}, algebraic connectivity μ, average hop count $\bar{h}$, diameter D, average clustering coefficient $\bar{C}$, assortativity ρ, maximum betweenness B_m, and maximum coreness c_m.

question, we first create a reference model by randomizing, for any window of length w minutes, the interactions observed in the MOBA datasets. The randomisation of, for example, the SM mapping is done by taking the players from all matches that started within the current time window and randomly assigning them to matches. Since the SM mapping does not take team information into account, the match assignment comes down to forming random groups of 10 unique players from the entire list of players who were active in the time window.
We run the parameter w from 1 to ∞ and depict the results, together with the original data, in Figure 7.3. Whereas the results for $w = 1$ leave little room for randomisation, the results for $w = \infty$ randomises the entire dataset. In Figure 7.3, the curves for $w = 1$ and the original data have a powerlaw-like shape. The curves for various values of w follow the $w = 1$ (original) curve for link weights of up to about 15 matches played together, but take afterwards an exponential-like shape, which indicates they are more likely to be the result of chance than of intended user behaviour. The fact that curves are markedly different for small time windows shows that it is very unlikely that players are found in the same match often simply because they happen to be online at the same time.

NETWORK SIZES

We first analyse the basic sizes in terms of nodes and links of the extracted graphs, and summarise the results in Table 7.1. Under the most general mapping, the SM mapping, graphs extracted from the Dota-League and DotAlicious datasets have different sizes (in Table 7.1, rows N and L, the respective columns "SM"). The network extracted from the DotAlicious dataset contains more links, while the two networks contain an equal number of nodes. Based on this general mapping (column "SM" in Table 7.1, for each of the two datasets DotA-League and DotAlicious), the graph extracted from the DotAlicious dataset is denser ($d = 4.00 \times 10^{-4}$ and $d = 6.52 \times 10^{-4}$, respectively), although the number of matches in the DotAlicious dataset is half that in the Dota-League dataset. We conclude from this that players who play regularly together in the DotAlicious dataset do so in more diverse combinations, because they create more links with fewer matches,

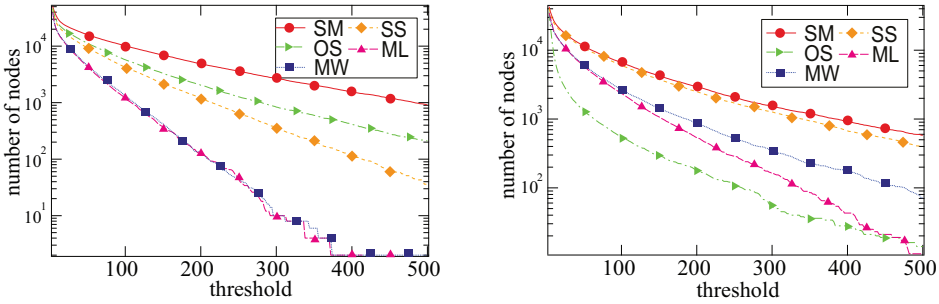


Figure 7.4: Number of nodes in the network as a function of the threshold for Dota-League (left) and DotAlicious (right).

than the players in the Dota-League dataset do.

Not only do the graphs obtained for different datasets differ, the graphs obtained using different mappings also highlight differences between the datasets (compare, for each row in Table 7.1, the respective values obtained for each dataset). In general, it seems that players from the Dota-League dataset, when they appear in the same game, play on opposing sides; in contrast, for the DotAlicious dataset they play mostly on the same side. This can be seen by contrasting the sizes of the networks extracted using the SS and OS mappings (rows N and L in Table 7.1). The DotAlicious network contains almost 3 times more nodes and links for the SS mapping than for the OS mapping. In contrast, for the Dota-League dataset, the networks extracted using the OS mapping are larger than those extracted using the SS mapping.

The tendency of DotAlicious players to play on the same side, and that of Dota-League players to play on opposing sides can be seen more clearly from Figure 7.4, where the number of nodes in the network as a function of the threshold is shown for all mappings. The Dota-League graphs are larger (have more nodes) for the OS mapping compared to the SS mapping throughout the range of thresholds. In the DotAlicious dataset, however, the OS mapping results in the smallest graph of all mappings. Moreover, forming links between players that played on the same side results in graphs that are almost as large as those produced with the SM mapping. This indicates that whenever DotAlicious players play many matches together, they almost surely play those matches on the same side. Arguably, playing together forms a stronger social bond than playing against each other; an observation that could not have been made based without studying different mappings and thresholds.

In addition to playing together, DotAlicious players also like to win together. Up from a threshold of 100, the lines for ML and MW markedly take different slopes in Figure 7.4 (right). The larger networks for MW compared to ML show that winning together forms true friendships—it leads to long-lasting relationships.

In contrast to the graphs extracted from the DotAlicious datset, in the graphs extrac-

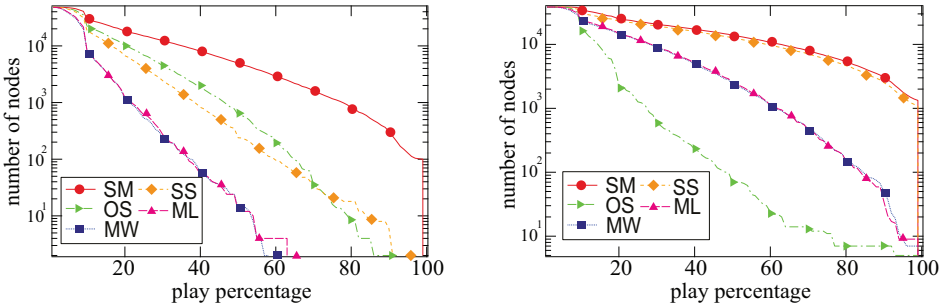


Figure 7.5: Number of nodes in the network as a function of the play percentage for Dota-League (left) and DotAlicious (right).

ted from the Dota-League dataset the network sizes for the ML and MW mappings are the same, indicating a 50-50 win ratio for matches played together. There is no indication that play relations are strengthened by winning in the Dota-League dataset. This can be explained by the fact that players in Dota-League cannot choose on which side they play; instead, the game mechanism offered in this community only allows joining the waiting queue (see Section 7.5). The 50-50 win ratio we observe for matches played together indicates that matches are generally well balanced.

By using different mappings we have revealed clear differences between the seemingly similar datasets used in this study. The most general SM mapping does not display those differences; for example, the curves for SM in both the left and right hand graph in Figure 7.4 are very similar. The PP mapping, however, can also reveal a difference between the two SM mappings, as can be observed in Figure 7.5. The number of nodes in the network for the higher percentages of games played together, is higher by an order of magnitude. This difference indicates that although both datasets contain pairs of players that play a high *number* of matches together, only the DotAlicious dataset contains players that play a high *percentage* of their matches with a select group of other players.

Although the PP mapping reveals differences between the DotAlicious and Dota-League datasets, it also clouds information that was visible using the regular mappings. In the DotAlicious dataset, the difference between the network sizes for the OS and SS mappings is no longer clear when using the PP mapping. The PP mapping highlights that players have, on average, a win ratio of 50%. Because the PP mappings produce directed graphs, the individual win ratio results in equally large networks for ML and MW.

SOCIAL NETWORK STRUCTURE

The metrics in the lower half of Table 7.1 offer a better insight into the social structure of the network. We look at three path length metrics: the average hop count ($\bar{h}$), the diameter (D), and the maximum betweenness centrality (B_m). These three metrics relate to how easily nodes can reach each other in the network and whether some nodes are more important with respect to reachability. Social networks are characterised by a low average hop count and a relatively high clustering coefficient; Watts and Strogatz [39]

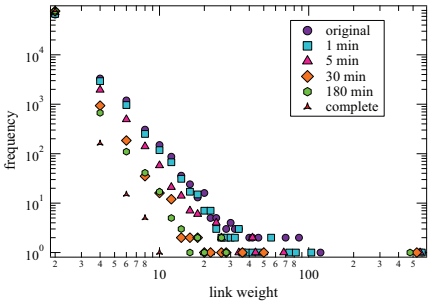
call this the small-world property. In Table 7.1, the SM mapping of both datasets shows very similar values for the metrics related to path length. Based on this general mapping function, the small average hop count and high clustering coefficient suggest that the extracted graphs indeed show small-world properties rather than the properties expected of random graphs.

The relatively high clustering coefficient that is often found in social networks is often the result of the fact that a friend of your friend is likely to also be your friend, too. For both the Dota-League and DotAlicious graphs the clustering coefficient of players that played in the same match is around 0.4 (0.37 for Dota-League and 0.43 for DotAlicious), indicating that the players you play with are also likely to play among each other. The higher clustering among DotAlicious players might be the result of the greater control players have over with whom they play, or might be the result of player behaviour.

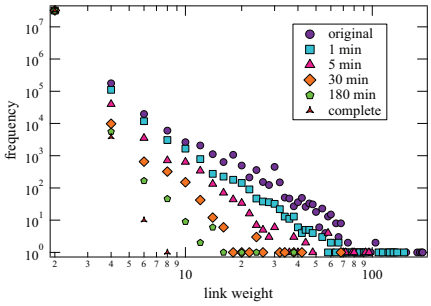
Contrary to many online social networks, the competitive element in online social games may result in foe relations, in addition to friendship relations, which can be studied via the OS and SS mappings. As can be seen in Table 7.1 for DotAlicious, the clustering coefficient for the SS mapping is higher than for the SM mapping. The increase in clustering coefficient indicates that, if we interpret being on the same side in a match as being friends, the friends of your friends are indeed likely to be friends of yours. The clustering of players increases a bit further if we use the MW mapping, showing the strengthening effect of winning matches together. Where winning together strengthens relations, losing matches does, from a clustering perspective, lead to slightly weaker relations as can be seen from the clustering coefficient for the ML mapping. As an indication that the ML and MW graphs are indeed different, the overlap in links between the ML and MW graphs is about 30%, i.e. 30% of the links in the ML graph also occur in the MW graph and vice versa.

Whereas the SS mapping results in graphs with a higher clustering coefficient for the DotAlicious dataset, the OS mapping creates graphs with a markedly lower clustering coefficient. Intuitively this ties in with the idea that although a friend of your friend is also a friend of yours, an enemy of your enemy is also your friend. Although players on opposing sides in a match instance will most likely not dislike each other outside the scope of that match, we nonetheless see a different graph structure for *foe* relations. The much lower clustering coefficient cannot be explained by other graph properties such as link density. Indeed, the link density in the OS graph is *higher* than in the SS graph.

Just as in Section 7.4.2, we find that the different mappings allow us to see differences between the two datasets. The clustering coefficients are the same for all graphs extracted from the Dota-League dataset. In particular, there is no evident difference between friend and foe relations in the Dota-League dataset. In contrast, the difference is prominent in the DotAlicious dataset. As observed previously, the PP extension of the SM mapping shows vastly different results for the two datasets (see Table 7.1, the columns "PP" corresponding to each dataset). The difference in the number of nodes is not large, but the difference in the size of the largest strongly connected component is. The Dota-League dataset shows no large strongly connected component, whereas the DotAlicious dataset shows a largest connected component of 3,000 nodes. However, the largest strongly connected component extracted from the DotAlicious dataset is poorly connected: it has a diameter of 74 hops and an average hop count of 18.45.



(a) Frequency of occurrence of link weights in the reference model for five different intervals and the original data for SC2. The mapping is SM.



(b) Frequency of occurrence of link weights in the reference model for five different intervals and the original data for WoT. The mapping is SM.

Figure 7.6: Results for methodology reality-check.

APPLICATION TO OTHER GAME GENRES

To explore whether our findings for the DOTA games are more widely applicable to other online game genres outside the multiplayer online battle arena (MOBA), we widen the scope to include two more game types: real time strategy (RTS) games and massively multiplayer online first person shooter (MMOFPS) games. RTS games are among the most popular genres today, and ask players to balance strategic and tactical decisions, often every second, while competing for resources with other players. Although faster-paced, MMOFPS games test the tactical team-work of players disputing a territory. We could expect RTS and MMOFPS games to lead to similar interaction graphs as MOBAs: naturally emerging social structures centred around highly active players. However, these game genres have different match-scales and team-vs-team balance than MOBAs. Moreover, RTS games can stimulate individualistic gameplay, while MMOFPS games may have teams that are too large to be robust.

We collect then analyse two additional datasets: for the RTS game StarCraft II (SC2) from Mar. 2012 to Aug. 2013, and for the MMOFPS game World of Tanks (WoT) from Aug. 2010 to Jul. 2013. For each of these popular games, we have collected over 75,000 matches, played by over 80,000 SC2 and over 900,000 WoT players. SC2 matches are not generally played in equally-sized teams, and 92% of the collected matches are 1v1-player. In contrast, 98% of WoT matches are 15v15-player. Such large teams can be much harder to maintain over time than the teams found in typical MOBAs, due to inevitable player-churn.

The results of the methodology reality-check for these two game genres, as shown in Figure 7.6b show similar, yet not so pronounced behaviour as for the DOTA datasets. Although players do not play nearly as often together in other genres' datasets as in the MOBA datasets, randomisation within only small time windows lowers the link weights. We conclude that it is unlikely that the relationships we identify are the result of chance encounters between players and, instead, indicate conscious, possibly out-of-

	StarCraft II					World of Tanks				
	SM	OS	SS	ML	MW	SM	OS	SS	ML	MW
N	907	611	314	95	212	4,340	477	4,251	561	1,824
N_{lc}	31	22	24	9	14	129	118	122	66	57
L	748	404	327	85	200	9,895	3,253	6,543	1,564	2,923
L_{lc}	58	21	44	13	24	2,329	1,243	1,160	519	473
d ($\times 10^{-4}$)	18	22	67	190	89	10.51	286.54	7.24	99.57	17.58
d_{lc} ($\times 10^{-4}$)	1,247	909.10	1,594	3,611	2,637	2,821	1,801	1,572	2,420	2,964
D	2	8	3	2	2	6	3	5	4	3
$\bar{C}$	0.58	0	0.70	0.65	0.65	0.79	0.10	0.78	0.88	0.87
ρ	-0.46	-0.45	-0.42	-0.58	-0.53	-0.10	-0.12	-0.06	-0.03	-0.10
B_m	0.91	0.74	0.84	0.80	0.79	0.09	0.08	0.11	0.20	0.20
c_m	53	11	32	32	32	29	15	14	14	14

Table 7.2: Metrics for $n = 10$: for other game genres: StarCraft II (RTS) and World of Tanks (MMO and FPS). The metrics we present: number of nodes N, number of nodes in largest connected component N_{lc}, number of links L, number of links in largest connected component L_{lc}, link density d, link density of largest connected component d_{lc}, algebraic connectivity μ, average hop count $\bar{h}$, diameter D, average clustering coefficient $\bar{C}$, assortativity ρ, maximum betweenness B_m, and maximum coreness c_m.

game agreements between players.

Alone or together? For SC2, the mappings lead to small graphs, with many small connected components. The majority of players participate in 1v1-player matches, but the 8% of players who do play in larger groups tend to play against each other more than together ($N = 611$ for the OS mapping, versus 314 for SS). When players do play on the same side, winning tends to strengthen the teams ($N = 212$ for the MW mapping, versus 95 for ML), just as we saw for the DotaLicious dataset. The connected components are strongly connected, yet small. The connected components of the mappings extracting same-team (SS) graphs are highly clustered. For the OS mapping, on the other hand, the largest component is a tree. The clustering coefficients observed in the various RTS graphs indicate much stronger team relationships in RTS games than in MOBAs. Because RTS games have not shown a trend of greatly increasing the number of players in the same instance, over the last decade, we hypothesise that RTS games will continue to spawn tightly-coupled teams that always play together; such teams are naturally vulnerable to player departures.

For WoT, the large team-size makes it difficult to organise teams well: the largest connected components for all mappings are not very large. Similarly to SC2 and DotaLicious, in WoT the players that do play often together do so on the same team and, again, players who play together are more likely to win rather than lose together. As modern FPS games tend to be played in increasingly larger teams, with 32v32-player games now not uncommon, we conclude that MMOFPS games will require additional mechanisms if they are to develop any form of robust social structure. Moreover, even more so than in the SC2 datasets, many players play only one or a few games: 69% of the more than 900,000 players played only once or twice. This is another area where developers could use the emerging social structures among their players to increase the number of players who keep on playing the game.

In conclusion, our formalism can be applied to other game-genres, for which it leads to new findings vs MOBAs, and suggests that even communities of popular networked games could benefit from improved mechanisms that foster denser interaction graphs

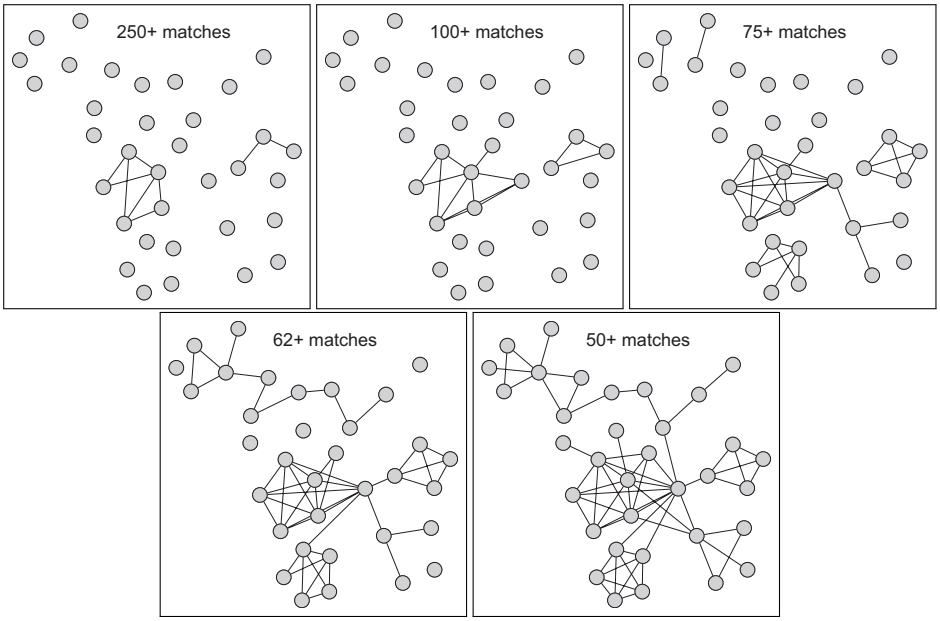


Figure 7.7: Clusters for the SS DotAlicious graph, for various values of the mapping threshold.

and a richer gaming experience.

IMPACT OF THE MAPPING THRESHOLD ON THE RESULTING NETWORK CONNECTEDNESS

An important property of networks is whether the network is connected. A disconnected network indicates that the nodes in the network cannot come into contact, either directly or via intermediate nodes. The number of connected components and the size of the largest connected component in a network are measures of how connected or fragmented the network is.

The threshold value has a large impact on the number of connected components, as depicted in Figure 7.7. For a threshold $n = 0$, the extracted graphs consists of a single connected component. For increasing values of n, the extracted networks become smaller and more indicative of clusters of players who are actively involved in relationships with other cluster members. The reverse process—lowering the threshold—makes the isolated clusters grow. We illustrate the dependency between cluster growth and threshold reduction in Figure 7.7, where plots from left to right depict the results for decreasing values of the mapping threshold. As the threshold value decreases, the few connected nodes in the left-most image grow into a single, large connected component. In practical terms, players first organise in smaller clusters before these smaller clusters all connect. The player in between the two initial clusters depicted in the left-most image of Figure 7.7 clearly functions as a hub.

Although it is possible to extract an almost fully connected network for all mappings, applying a threshold reduces the relative size of the largest component significantly. Figure 7.8 shows the number of connected components excluding isolated nodes. As the

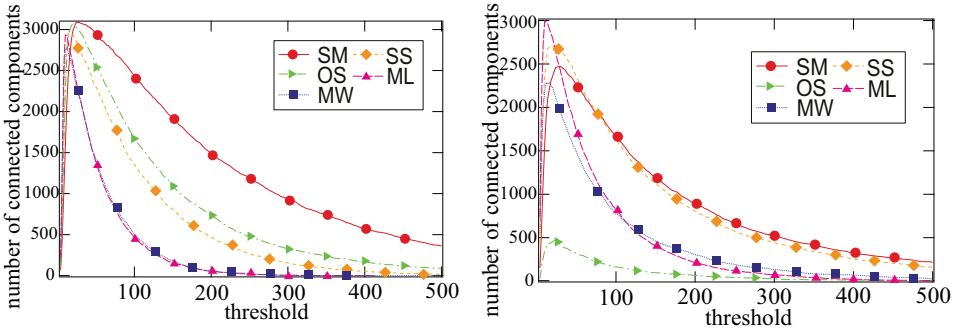


Figure 7.8: Number of connected components for Dota-League (left) and DotAlicious (right).

threshold value increases, the dominance of the largest component diminishes and the number of connected components in the graph increases rapidly. This indicates that the network falls apart quickly in many small components (players with a strong gaming relationship), rather than that nodes are stripped of the giant component or that it splits into a few large subcomponents. The small components are all between two and five nodes in size, which is consistent with the maximum size of five players per team that is typical of DotA (see Section 7.2). At a threshold value of 28, half of the nodes are located in small clusters. At a threshold value of 100, 85% of the nodes are in small clusters, yet the distribution of cluster sizes does not change much. The peak value at a threshold of 28 is in the same region where the number of nodes in the graph stops decreasing dramatically and enters a regime of more steady decrease as was shown in Figure 7.4. In Figure 7.9, the number of strongly connected components in the directed graphs ex-

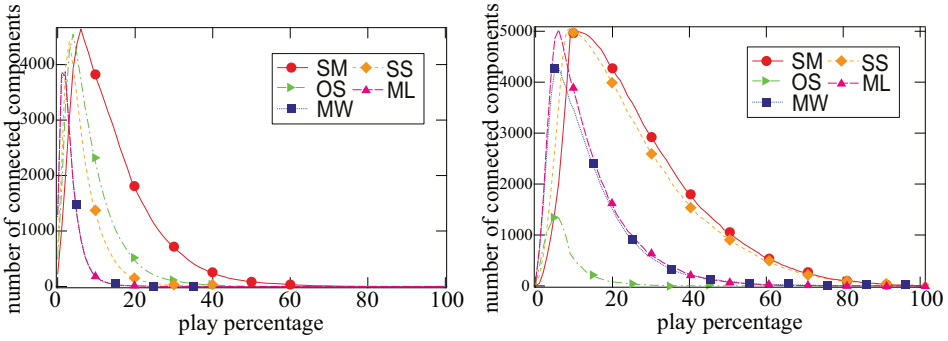


Figure 7.9: Number of strongly connected components with PP mapping for Dota-League (left) and DotAlicious (right).

tracted with the PP mapping is shown for the Dota-League dataset (left) and DotAlicious dataset (right). The number of strongly connected components has higher peaks in the networks extracted from the DotAlicious dataset than for the Dota-League dataset, and also stays much higher for higher play percentages. This indicates that although the network falls apart in many small clusters in both datasets, only in the DotAlicious dataset

these clusters are larger than 1 node. Using the largest component to characterise the network may provide valuable insight for low threshold values, but still is too crude and misses the composite nature of the network. Scaling up the range of the threshold values could, in an easier manner than most clustering algorithms, bring forth new and significant (strong) gaming relations.

The betweenness centrality (B) and coreness (c) can give clues as to whether a graph contains important nodes, such as, for example, influential spreaders [122]. For the graphs constructed with the lower thresholds, the maximum betweenness value indicates that between 4 and 9 percent of all the shortest paths cross at the most central node. This value increases with an increasing threshold while the link density increases. This indicates that, as the largest component shrinks, some players play an increasingly important role in facilitating short paths.

7.5. FORMALISM-BASED MATCH RECOMMENDATIONS

We focus in this section on the use of our formalism for extracting graphs for improving typical game functionality. We propose, as an example, an algorithm that can assist in matchmaking decisions. A good matchmaking system ensures that players in a game have matching profiles and could, therefore, take player clustering information into account. In this section, we analyse how matches are formed by players and propose a graph-inspired matchmaking algorithm that leads to much stronger social ties than random matchmaking, and is slightly better than the real algorithms used by the communities from which we collected our datasets.

The two datasets under study represent communities that deploy in practice different matchmaking algorithms. For Dota-League, players who want to play a match first join a waiting queue. When there are 10 or more players in the waiting queue, the matchmaking algorithm will form teams that are balanced in terms of the skill levels of the players. Although this matchmaking algorithm enforces balanced matches, it does not take into account the social ties of the players. Often, players synchronise themselves out-of-game via instant messaging tools to join the waiting queue at the same time and thus increase the probability to end up in the same match as their preferred players, but there is no guarantee that they will play on the same team. In contrast, for DotAlicious, each game server has a number of open matches waiting for players to join, and each arriving player can select which match to join and on which team.

To study the effect of a matchmaking algorithm, we present a scoring of matches that reflects the utility derived by players from a game. Following McGonigal [114], we assume that matches played by players with strong social ties are enjoyed to a higher extent than those played amongst players that have weak or no social ties. As we have shown in Section 7.4.2, a large part of the DotA players are grouped into relatively small, high link-weight clusters. This indicates that these players enjoy their gaming experience best when they play with or against a small set of other players. Since there are multiple ways to rank matches, our goal is not to propose a unique scoring methodology, but rather to show how to compare and possibly improve matchmaking based on one such socially-aware scoring system.

For each match, we assign a score to a match based on the connected components, referred to as *cluster*, to which the players of the match belong. Cluster membership

is derived for the different mappings (excluding PP) described in Section 7.3 as follows: first, a threshold value of $n = 100$ is used to filter out weaker links between players. Then, we identify each connected component (cluster) in the extracted graph. The resulting clusters are then numbered, and each cluster member (player) is labelled with its cluster number. In our analysis, we assign a *score* to every match based on the overlap of cluster membership amongst players. Intuitively, we design the score to reward those matches in which many players from the same cluster participate. Specifically, a match receives one point for every player in clusters that are represented in the game by two or more players.

Team A

Player	Cluster
A_1	1
A_2	2
A_3	1
A_4	4
A_5	5

Team B

Player	Cluster
B_1	2
B_2	5
B_3	3
B_4	6
B_5	3

Figure 7.10: Example of a match with cluster numbers associated to the players.

Consider the match schematically represented in Figure 7.10. Team A consists of players "A_1" to "A_5", as can be seen in the column labelled "Player"; team B consists of players "B_1" to "B_5". The column labelled "Cluster" records the cluster number for each player. In total, this match is assigned 7 points, as follows. First, 2 points are given for each of cluster 1, which is represented in this game by players "A_1" and "A_3", and cluster 2 (players "A_1" and "B_1"). Then, 3 points are given for players "A_4","B_3", and "B_5" (cluster 3). Players "A_5","B_2", and "B_4" have no fellow cluster members in the match and will be assigned 0 points.

To favour the small clusters that lead to novel human emotions [114], and which are shown to be prevalent in our datasets in Section 7.4.2, we design the scoring system such that the largest cluster is not considered when assigning points. This effectively avoids biasing the matchmaking algorithm to the dominant set of links, which is akin to recommending a well-known tune, not because it is similar to the requester's tastes, but because it is present in most other people's playlists.

The socially-aware matchmaking algorithm which works as follows. First, for each 10-minute time interval (sliding windows), the algorithm builds a list of all the players who are online. In practice, the algorithm can build this list from the online information provided by the waiting rooms of each DotA community; to obtain this list from the real (raw) datasets (see Section 7.2), we define a player as being online during an interval if the player has joined at least one match during the interval. Second, the algorithm computes the cluster membership for each player. Third, from the largest online players' cluster to the smallest, all online players from the same cluster are assigned to new matches if size permits; otherwise, the cluster will be divided into two parts and players from one part will be assigned into new scheduled matches. In practice, the third step can be executed whenever the number of players exceeds the number needed for a game (10), at the end of the 10-minute interval, or periodically every few minutes; The rationale behind obtaining all the online players in a time interval is that players who

are aware that their friends are online are likely prepared to wait for a short duration in order to play together.

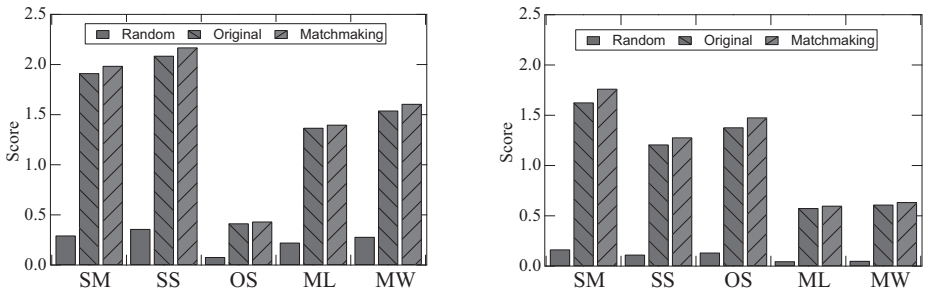


Figure 7.11: Average match scores when performing matchmaking for DotAlicious (left) and Dota-League (right).

The matchmaking results are shown in Figure 7.11; higher scores are better. For comparison purposes, we have also scored the matches obtained via randomly matching players that are online during each time interval (the "Random" matchmaking algorithm in Figure 7.11) and the matches observed in the real datasets (the "Original" matchmaking algorithm). The "Random" algorithm indeed scores the lowest, as it does not take into account social ties. Our proposed matchmaking algorithm (the "Matchmaking" algorithm in Figure 7.11) improves the scores of matches, by a large margin when compared with the "Random" algorithm and by a small but non-negligible margin when compared to the "Original" algorithm. We attribute the better scores achieved by our admittedly simple "Matchmaking" algorithm to the difficulty of seeing whether friends are online in the studied systems, due to shortcomings in the offered service. Without proper information, some players may not be aware that some of their friends are online and thus join other matches. Indeed, we find that the improvement is smaller for DotAlicious than for Dota-League, and attribute this to players in DotAlicious having more freedom and tools for selecting whom to play with. By using a graph perspective instead of manual off-line synchronisation, even a simplistic matchmaking algorithm can reach higher match utility scores by leveraging cluster information obtained after a fairly high threshold, that is, after processing only a small part of the original graph. By using a more complex multi-faceted graph extraction rule that, for instance, would include information on friend and foe relationships, we expect that an even better matchmaking could be achieved.

7.6. DECREASING THE SPECTRAL RADIUS BY LINK REMOVALS

The second topic of this chapter is that of graph manipulation. In addition to extracting graphs from datasets and using those graphs to improve the performance of an online gaming server, we can ask whether it is possible to change the topology of the graph in such a way that it is less vulnerable to spreading processes.

The largest eigenvalue $\lambda_1(A)$ of the adjacency matrix A, called the spectral radius of

the graph, plays an important role in dynamic processes on graphs. As we have seen in chapter 2, the steady-state fraction of infected nodes in a network in the SIS model is determined by a phase transition at the epidemic threshold $\tau_c^{(1)} = \frac{1}{\lambda_1(A)}$: when the effective infection rate $\tau > \tau_c$, the network is infected, whereas below τ_c, the network is virus-free. Beside virus spread, the same type of phase-transition threshold [123] in the coupling strength $g_c \sim \frac{1}{\lambda_1(A)}$ occurs in a network of coupled oscillators.

Motivated by a $\frac{1}{\lambda_1(A)}$ threshold separating two different phases of a dynamic process on a network, we want to change the network in order to enlarge the network's epidemic threshold τ_c, or, equivalently, to lower $\lambda_1(A)$. Removing nodes is often too drastic, and, therefore, we concentrate here mainly on the problem of removing m links from a graph G with N nodes and L links. We are searching for a strategy so that, after removing m links, λ_1 is minimal. Earlier, Restrepo *et al.* [124] have initiated an instance of this problem: "How does λ_1 decrease when links are removed?" They introduced a new graph metric, called the dynamical importance $I_x = \frac{\lambda_1(A)-\lambda_1(A\setminus\{x\})}{\lambda_1(A)}$, where x either denotes the removal of a link $x = l$ or of a node $x = n$. The dynamical importance was further investigated by Milanese *et al.* [125]. Both Restrepo *et al.* and Milanese *et al.* have approached the problem by using perturbation theory. However, they did not consider optimality of their removal strategy.

The objective of removing links from a graph is to reduce the spectral radius, and by that, the epidemic threshold of a virus in the graph. A precise formulation of the optimal removal of links from a graph is given as follows:

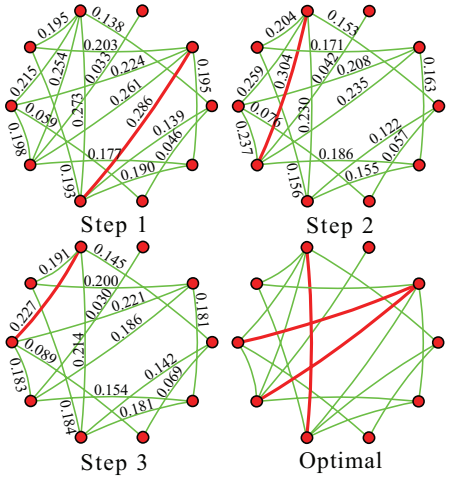
Problem 1 (Link Spectral Radius Minimisation (LSRM) problem)**.** *Given a graph $G(\mathcal{N},\mathcal{L})$ with N nodes and L links, spectral radius $\lambda_1(G)$, and an integer number $m < L$. Which m links from the graph G are to be removed, such that the spectral radius of the reduced graph G_m of $L-m$ links has the smallest spectral radius out of all possible graphs that can be obtained from G by removing m links?*

7

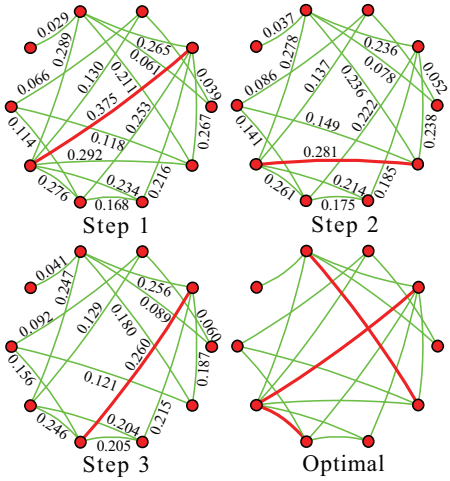
The LSRM problem is NP-hard [126], as can be shown by reducing the problem to an equivalent problem, namely finding a Hamiltonian path in a graph, that is known to be NP-complete [127]. As a path graph has the lowest possible $\lambda_1(A)$ of $2\cos(\frac{\pi}{N+1})$, optimally removing $m = L - N + 1$ links either results in a Hamiltonian path (if $\lambda_1(A) = 2\cos(\frac{\pi}{N+1})$), or not.

As an example to illustrate the NP-completeness of the LSRM problem, Figure 7.12a-7.12c show, in a topology of $N = 10$ nodes and $m = 3$ link removals, that the "best single step strategy" is not always optimal in the end. The "best single step strategy" consists of removing the link that lowers $\lambda_1(A) - \lambda_1(A_1) = y_1$ most in the first step. Next, in the second step, the link that lowers $\lambda_1(A_1) - \lambda_1(A_2) = y_2$ most is removed and finally, in the third step, the link that lowers $\lambda_1(A_2) - \lambda_1(A_3) = y_3$ most is removed. The optimal situation depicts the removal of $m = 3$ for which $\lambda_1(A) - \lambda_1(A_3) = y^*$ is maximal. Hence, $y_1 + y_2 + y_3 \leq y^*$.

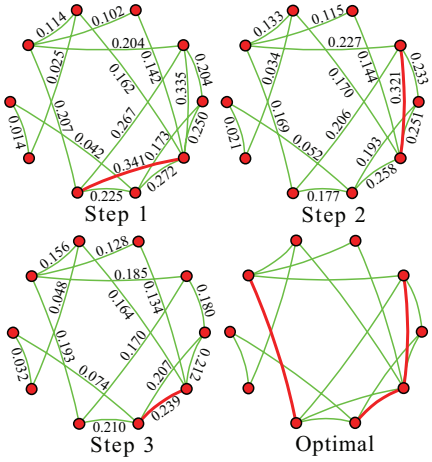
In addition, 10^6 instances of Erdős-Rényi (ER) random graphs with $N = 10$ and link density $p = \frac{2\ln N}{N}$ have been generated. In each instance, the "best single step strategy" and the global optimum have been computed. In 63185 (6,3%) instances, there was no overlap in links, in 332262 (33,2%) ER graphs, there was one link in common, in 97944 (9,8%) ER graphs, we found 2 links in common and in the remaining 506609 (50,7%) ER



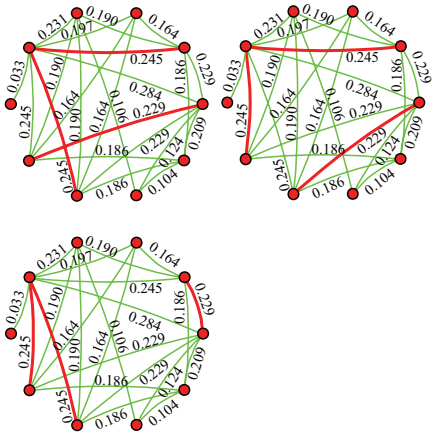
(a) An example of a graph with $N = 10$ nodes, where *none* of the links in the greedy approach appears in the optimal set of links. The numbers indicate the change in largest eigenvalue $\lambda_1(A) - \lambda_1(A \setminus \{l\})$ after removal of link l.



(b) An example of a graph with $N = 10$ nodes, where only 1 link in the greedy approach appears in the optimal set of links.



(c) An example of a graph with $N = 10$ nodes, where *two* links in the greedy approach appear in the optimal set of links.



(d) An example where the global optimum is not unique. The original $\lambda_1(A) = 5.065310$ and, after removal of three links, the smallest largest eigenvalue is $\lambda_1(A_3) = 4.312414$. Consequently, the largest $\lambda_1(A) - \lambda_1(A_3) = 0.752896$ is obtained after removal of the three red links.

Figure 7.12: Examples of the results of the greedy approach, ranging from no correct links in the greedy solution to the greedy solution giving the optimal solution.

graphs, all 3 links in the "best single step strategy" were the same as in the global optimum. Moreover, Figure 7.12d illustrates that the global optimum is not always unique. The global optimum may not be unique, as it is possible that the removals of different sets of m links will lead to cospectral or even isomorphic smaller graphs, as indicated in Figure 7.12d.

The maximum number m of links that have to be removed from G to ensure that λ_1 in G_m is lowered below some given value ξ is

$$m \leq L - \frac{\xi^2 + N - 1}{2}$$

which is derived from the bound [30, (3.48) on p. 54], due to Yuan Hong [128],

$$\lambda_1 \leq \sqrt{2L - N + 1} \tag{7.1}$$

for connected graphs, else $\lambda_1 \leq \sqrt{2L\left(1 - \frac{1}{N}\right)}$.

7.6.1. STRATEGIES FOR LINK REMOVAL

Since the LSRM problem is NP-complete, we will resort to a greedy heuristic to remove links. In each step, the link that is expected to reduce the largest eigenvalue the most is taken from the network. Because it is computationally expensive to compute the real effect of removing a link on the largest eigenvalue for each possible link, we compare eight different strategies of selecting the best link to remove. The strategies for greedily removing a link are all based on bounds for the largest eigenvalue. We will denote a strategy by S followed by the appropriate node properties involved.

The first strategy is based on the bounds for the difference between the largest eigenvalue of the adjacency matrix A of graph G and that of adjacency matrix A_m of graph G with m links removed [126]:

$$2 \sum_{l \in \mathcal{M}_m} (w_1)_{l^+} (w_1)_{l^-} \leq \lambda_1(A) - \lambda_1(A_m) \leq 2 \sum_{l \in \mathcal{M}_m} (x_1)_{l^+} (x_1)_{l^-}$$

where x_1 and w_1 are the eigenvectors of A and A_m corresponding to the largest eigenvalues $\lambda_1(A)$ and $\lambda_1(A_M)$, respectively, and where a link l joins the nodes l^+ and l^-. Removing the link for which $l = i \sim j$ for which $(x_1)_i (x_1)_j$ is maximal will increase the upper bound of the decrease in spectral radius. This strategy is denoted by $S = x_i x_j$ instead of $S = (x_1)_i (x_1)_j$ to simplify the notation.

A lower bound of the largest eigenvalue of the adjacency $\lambda_1 \geq \frac{N_3}{N_2}$ has been proved in [45], where N_k is the total number of walks of length k. A walk in a graph is a sequence of links where each next link is adjacent to at least one of the endpoints of the previous link. The lower bound $\frac{N_3}{N_2}$ appeared earlier as an approximation of λ_1 in [129], and it is a perfect linear function of assortativity the ρ [45].

Let us first look at the decrease of $\frac{N_3}{N_2}$ by a link removal. We know [45] that

$$\frac{N_3}{N_2} = \frac{\sum_{i=1}^{N} d_i^3 - \sum_{i \sim j} (d_i - d_j)^2}{\sum_{i=1}^{N} d_i^2}$$

We denote N_3 and N_3' as the number of 3 hop walks in the original graph G and in the graph $G\backslash\{l_{ij}\}$ with one link $l = i \sim j$ less, respectively. Then, we have that

$$\begin{aligned}\Delta_3 &= N_3 - N_3' \\ &= d_i^3 + d_j^3 - (d_i-1)^3 - \big(d_j-1\big)^3 - (d_i-d_j)^2 - \sum_{l\in\mathcal{N}(i), l\neq j} (d_l-d_i)^2 - (d_l-d_i+1)^2 \\ &\quad - \sum_{l\in\mathcal{N}(j), l\neq i} \big(d_l-d_j\big)^2 - \big(d_l-d_j+1\big)^2\end{aligned}$$

where d_i is the degree of node i in the original graph G and $\mathcal{N}(i)$ is the set of the neighbours of node i. The decrease Δ_3 can be simplified as

$$\begin{aligned}\Delta_3 &= 2 - 3(d_i+d_j) + 3(d_i^2+d_j^2) - (d_i-d_j)^2 + \sum_{l\in\mathcal{N}(i), k\neq j} (2d_l - 2d_i + 1) + \sum_{l\in\mathcal{N}(j), l\neq i} \big(2d_l - 2d_j + 1\big) \\ &= 2(d_i^2+d_j^2) + 2d_id_j + 2 - 3(d_i+d_j) + \big(d_i+d_j-2\big) - 2d_i(d_i-1) - 2d_j(d_j-1) \\ &\quad + \sum_{l\in\mathcal{N}(i), k\neq j} 2d_l + \sum_{l\in\mathcal{N}(j), k\neq i} 2d_l \\ &= 2d_id_j + \sum_{l\in\mathcal{N}(i), k\neq j} 2d_l + \sum_{l\in\mathcal{N}(j), k\neq i} 2d_l \\ &= 2d_id_j + 2(s_i+s_j) - 2(d_i+d_j)\end{aligned}$$

where

$$s_i = \sum_{l\in\mathcal{N}(i)} d_l \tag{7.2}$$

is the total degree of all the direct neighbours of a node i. Similarly, the decrease in the number of two hop walks is denoted as

$$\Delta_2 = N_2 - N_2' = 2(d_i + d_j - 1)$$

Note that Δ_2 and Δ_3 are only functions of a local property, i.e. the degree d_i and d_j of the two end nodes of a link l_{ij}. The complexity of computing Δ_3 or Δ_2 for all linked node pairs is $O(N^2)$ in a dense graph, which is the worst case.

This hints that the spectral radius is possibly decreased the most by a link removal that either reduces $S = \frac{N_3}{N_2}$ or the assortativity $S = \rho_D$ the most. Strategy $S = \frac{N_3}{N_2}$ will remove the link such that $\frac{N_3-\Delta_3}{N_2-\Delta_2}$ is minimised.

The other considered strategies $S = d_id_j$ and $S = d_i + d_j$ remove that link $l = i \sim j$ with largest sum or product of the degrees of the link's end points, whereas the strategies $S = s_i + s_j$ and $S = s_is_j$ remove the link with the largest sum or product of the total degree s_i of the neighbours at both end points. Finally, we also considered the strategy $S =$ betweenness, that removes the link with highest link betweenness, i.e. the number of shortest paths between all node pairs that traverse the link.

A final strategy is to remove the link that possibly disconnects the graph G into two disjoint graphs G_1 and G_2. However, this strategy is not always optimal as illustrated in Figure 7.13. Only when both G_1 and G_2 are the same, we found that the removal of the connecting link induces the largest decrease in $\Delta\lambda_1$. Since this strategy cannot be applied always, we have further ignored this strategy.

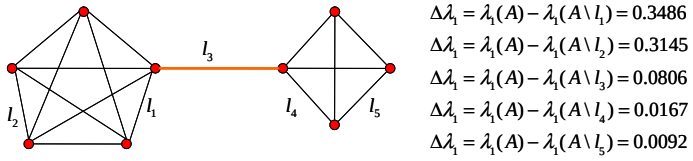


Figure 7.13: All possible $\Delta\lambda_1$ are computed when one link is removed.

7.7. PERFORMANCE OF THE DIFFERENT LINK REMOVAL STRATEGIES

We define the performance measure Ξ_S of a particular link removal strategy S as

$$\Xi_S = (\lambda_1(A) - \lambda_1(A_1))_{\text{optimal}} - (\lambda_1(A) - \lambda_1(A_1))_{\text{Strategy } S}$$

Figure 7.14 compares the strategies introduced in the previous section. Figure 7.14 confirms that strategy $S = x_i x_j$ is superior to all other strategies. There is a very small difference between the strategies $S = d_i + d_j$ and $S = d_i d_j$ and between $S = s_i + s_j$ and the corresponding product $S = s_i s_j$. In both cases the product strategy is slightly better (but the difference is not observable from Figure 7.14).

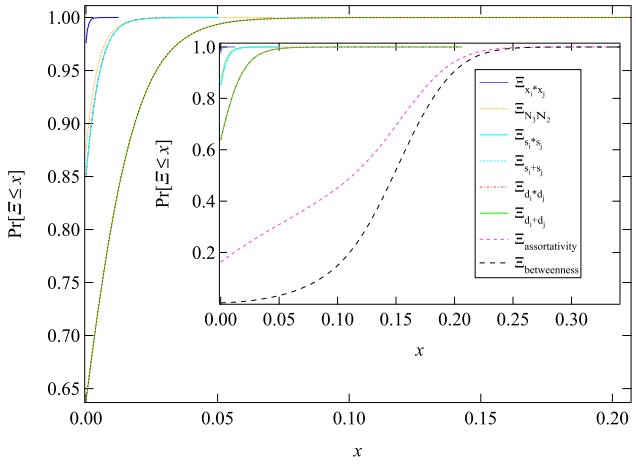


Figure 7.14: Various strategies applied to 10^6 instances of ER graphs with $N = 20$ and $p = 2\ln N/N$. The insert shows two additional strategies "assortativity" and "betweenness" that are clearly worse than the others.

7.7.1. REMOVING $m > 1$ LINKS

In this section, we investigate the behavior of the several strategies when more than one link is removed. We generated 10^4 Erdős-Rényi graphs with $N = 10$ nodes and $L = 20$ links, of which about two percent are disconnected. From each of the generated graphs, all the links are removed one by one following the different "greedy" strategies. We compare the decrease in λ_1 for each strategy to the optimal solution found by removing all

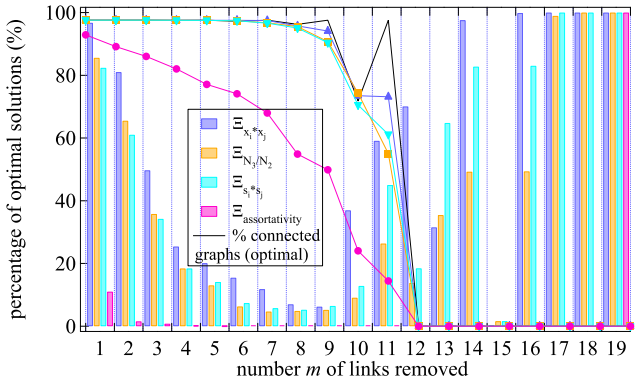


Figure 7.15: Four strategies compared with the global optimum as a function of the number m of removed links in ER random graphs with $N = 10$ nodes and $L = 20$ links, where 10^4 instances are generated. The lines show the percentage of connected graphs per strategy after the removal of m links.

possible combinations of m links. In Figure 7.15, the percentage of agreement between the greedy strategies and the optimal strategy is shown.

Figure 7.15 illustrates that strategy $S = \max_{1\le(i,j)\le N}(x_1)_i\,(x_1)_j$ is nearly always (except for $m = 13$) superior to strategy $S = N_3/N_2$ and $S = s_i s_j$. Figure 7.15 exhibits a regime change from $m = 10$ on, where the connectivity of the graphs starts to decrease rapidly.

The peculiar regime for $m > 10$ can be understood as follows. The optimal solution for $m = 10$ removals is a circuit, if the original graph contains a single connected circuit on $N = 10$ nodes. If strategy $S = \max_{1\le(i,j)\le N}(x_1)_i(x_1)_j$ finds the optimal solution for $m = 10$ removals, the only possible solution for $m = 11$ removals is to cut the circuit to form a path. This is also the optimal solution. The eigenvector components of a path graph are symmetrical around the node(s) in the middle of the path and are maximal for the centre node(s). Strategy $S = \max_{1\le(i,j)\le N}(x_1)_i(x_1)_j$ will, for the next link removal, cut the path in the middle. The resulting graph is also the optimal solution. In the next step, however, the strategy will cut one of the paths in two, resulting in three paths of lengths one, two and four links, respectively. The optimal solution for $m = 13$ link removals consists of a graph with three paths of lengths two and one of length three. This graph can never be formed by strategy $S = \max_{1\le(i,j)\le N}(x_1)_i(x_1)_j$ starting from a circuit. The optimal solution for $m = 14$ consists of two paths of length two and two paths of length one, which can be obtained in many different ways, including cutting the longest path of the solution for $m = 13$. In almost 98% of the cases this solution is found by strategy $S = \max_{1\le(i,j)\le N}(x_1)_i(x_1)_j$. The high success rate means, at the same time, that the optimal solution for $m = 15$ is almost never found because it cannot be reached from the optimal solution of $m = 14$ by another link removal, regardless of the followed strategy. The weaker performance of strategy $S = s_i s_j$ for $m = 12$ can be explained by considering the optimal solution for $m = 11$ which is a path of nine links. Strategy $S = s_i s_j$ removes the link that has the maximum product of the one hop neighbours of its endpoints. Since a path has an even degree distribution, except for the endpoints, the five links that form

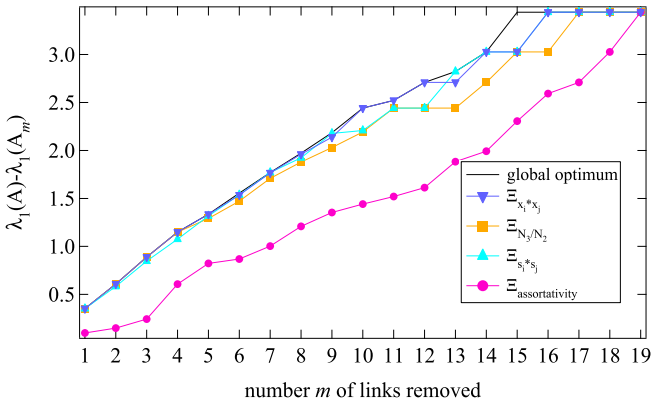


Figure 7.16: The performance $\lambda_1(A) - \lambda_1(A_m)$ of four strategies versus m link removals in a typical instance of a graph with $N = 10$ and $L = 20$ links.

the centre of the path have an equal probability of being removed. Consequently, the optimal solution for $m = 11$ will result in the optimal solution for $m = 12$ only one in five times. The other four possibilities lead to a graph with either a combination of a path of length two and a path of length six or a combination of a path of length three and a path of length five. Both these graphs will give the optimal solution for $m = 13$ link removals, which explains the increased success rate for $m = 13$.

At $m = 15$, the graph consists of 5 links and $N = 10$ nodes, configured in separated "cliques" K_2 (i.e. line segments) and the largest eigenvalue is minimal at $\lambda_1 = 1$. For $m > 15$, the strategies are all the same: a clique K_2 (i.e. disjoint link) is removed.

Figure 7.16 illustrates four strategies on a typical instance of a network with $N = 10$ and $L = 20$ links. While the strategy $S =$ assortativity clearly under performs, the three other strategies $S = x_i x_j$, $S = N_3/N_2$ and $S = s_i s_j$ are competitive: for small m, the strategy $S = x_i x_j$ excels (as shown in Figure 7.15), but for larger m the others can outperform. Again, this phenomenon is characteristic for an NP-complete problem, where the whole previous history of links removals affects the current link removal. The considered strategies (except for the global optimum one) are greedy and only optimise the current link removal, irrespective of the way in which the current graph G_m is obtained previously.

7.8. SCALING LAW OF $(\lambda_1(A) - \lambda_1(A_m))_{\text{OPTIMAL}}$

Another observation from Figure 7.16 is that

$$\Delta\lambda_m|_{\text{optimal}} = \lambda_1(A) - \lambda_1(A_m)|_{\text{optimal}} = O\left(m^{\beta}\right) \tag{7.3}$$

where $\beta \leq 1$. In other words, we conjecture that the scaling of $\lambda_1(A) - \lambda_1(A_m)$ with m is sublinear in m (for non-regular graphs) and that the coefficient β is likely a function of the type of graph. Obviously, $\Delta\lambda_m = 0$, when $m = 0$. Applying the upper bound (7.1) to

$\lambda_1(A_m)$ shows that

$$\begin{aligned}\Delta\lambda_m &\geq \lambda_1(A)-\sqrt{1-\frac{1}{N}}\sqrt{2L-2m}\\ &\geq \lambda_1(A)-\sqrt{1-\frac{1}{N}}\sqrt{2L}+\sqrt{1-\frac{1}{N}}\sqrt{2m}=O\left(m^{1/2}\right)\end{aligned}$$

On the other hand, if G_m is a regular graph, then

$$\Delta\lambda_m=\lambda_1(A)-\frac{2L-2m}{N}=O(m)$$

In particular, if G and G_m are regular graphs, then

$$\Delta\lambda_m=\frac{2m}{N} \tag{7.4}$$

These arguments illustrate that $\frac{1}{2}<\beta\leq 1$. Figure 7.16 shows that $\lambda_1(A)-\lambda_1(A_m)|_{\text{optimal}}$ is likely close to $\beta=1$, which suggests that the optimal way to remove m links is to make G_m as regular as possible, because the lowest possible $\lambda_1(A_m)$ with given N and $L-m$ is obtained for a regular graph (as follows from the Rayleigh inequality $\lambda_1(A)\geq\frac{2L}{N}$).

While the law (7.3) is difficult to prove in general, we provide evidence by computing the decrease in λ_1 when m random links are removed in the class of Erdős-Rényi random graphs $G_p(N)$. For sufficiently large Erdős-Rényi random graphs $G_p(N)$, it is known [30] that

$$E[\lambda_1]=(N-2)\,p+1+O\left(\frac{1}{\sqrt{N}}\right)$$

When m random links are removed from $G_p(N)$, we again obtain an Erdős-Rényi random graph with link density

$$p^*=\frac{L-m}{\binom{N}{2}}$$

Hence,

$$\begin{aligned}E[\Delta\lambda_m]&=E\left[\lambda_1\left(G_p(N)\right)\right]-E\left[\lambda_1\left(G_{p^*}(N)\right)\right]\\ &=(N-2)\left(p-p^*\right)+R_p(N)\end{aligned}$$

where the error term $R_p(N)$ is unknown. Assuming that $R_p(N)$ is negligibly small, we find, for sufficiently high N,

$$E[\Delta\lambda_m]\simeq\frac{(N-2)\,m}{\binom{N}{2}}=\frac{2m}{N}-\frac{2m}{N(N-1)}$$

Thus, the average decrease in $\lambda_1(A)-\lambda_1(A_m)$ after removing m random links in $G_p(N)$ is approximately, for large N,

$$E[\Delta\lambda_m]\simeq\frac{2m}{N} \tag{7.5}$$

which is close to (7.4) for regular graphs.

For $m = 1$, simulations on various types of graphs in Figure 7.17 and Figure 7.18 suggest the scaling law

$$(\lambda_1(A) - \lambda_1(A_1))_{\text{optimal}} = \frac{\alpha}{N} \tag{7.6}$$

where α is graph specific. In other words, $N\Delta\lambda_1 = \alpha$ is independent of the size of the graph.

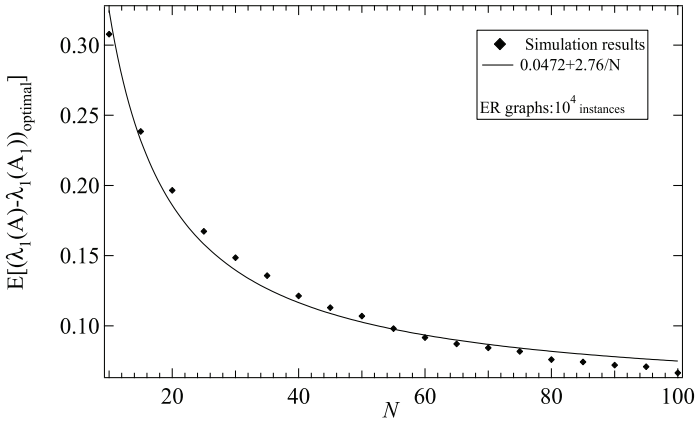


Figure 7.17: The scaling law of $(\lambda_1(A) - \lambda_1(A_1))_{\text{optimal}}$ for ER random graphs as a function of N.

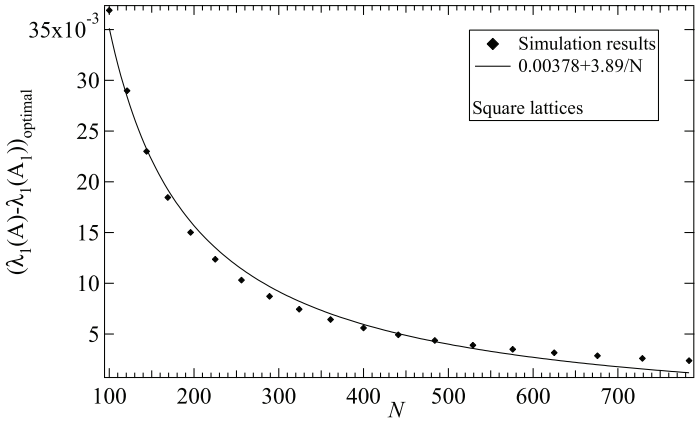


Figure 7.18: The scaling law of $(\lambda_1(A) - \lambda_1(A_1))_{\text{optimal}}$ for square lattices as a function of N.

Ignoring the asymptotic nature of the analysis that led to (7.5), we observe that, for $m = 1$, a maximum occurs at $N = 2$. Figure 7.19 shows the pdf of $\Delta\lambda$ for Erdős-Rényi random graphs, where for each curve 10^6 ER graphs have been created in which one random link was removed. The simulations agree with $E[\Delta\lambda] \simeq \frac{2}{N}$ and indicate that $\text{Var}[\Delta\lambda] \simeq 13 + 2E[\Delta\lambda]$. Since a random link removal is inferior to the removal of the optimal link, Figure 7.17 indeed illustrates that the coefficient of the inverse N scaling

law $\alpha_{G_p(N)} \simeq 2.75 > 2$. Figure 7.18 shows that $\alpha_{lattice} \simeq 3.9 > \alpha_{G_p(N)} \simeq 2.75$, which may indicate that deviations from regularity causes λ_1 to decrease more.

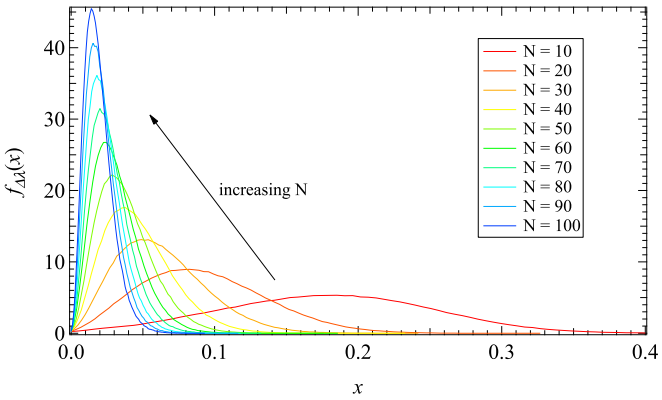


Figure 7.19: The probability density function of $\Delta\lambda_1$ for ER random graphs of several sizes N.

7.9. CHAPTER SUMMARY

Extracting graphs from a dataset usually comprises mapping entities in the dataset to nodes and relations between the entities to links. We develop a formalism to extract the implicit social relations from two datasets containing match instance of two very popular online games. By analysing the implicit social networks we show that the two gaming communities differ in whether players play on the same team or on opposing teams. By varying the threshold in the link mapping it becomes clear that the social network is composed of many small but strongly connected components that are themselves again connected by weaker links. Moving from graph extraction to manipulation we study the minimisation of the spectral radius by link removal. The spectral radius is both fundamental in graph theory as well as in many dynamic processes in complex networks such as epidemic spreading, synchronisation and reaching consensus [30, p. 200]. We show that the spectral radius minimisation problem is an NP-hard problem, which opens the race to find the best heuristic. In particular, in large infrastructures such as transportation networks, where removing links can be very costly, a near to optimal strategy is desirable. We show that an excellent strategy is $S = x_i x_j$: on average, this strategy outperforms most other heuristics, but it does not beat them all the times. Beside graph theoretic bounds and arguments that underline the goodness of the heuristic $S = x_i x_j$, two scaling laws (7.3) and (7.6) are found: these laws may help to estimate the decrease in spectral radius as a function of the number N of nodes and/or the number m of link removals.

8

CONCLUSION

This thesis describes the research efforts to understand dynamic processes on graphs; specifically the SIS process on graphs. The main topics are that of temporal behaviour and non-Markovian behaviour. In the case of epidemic algorithms, the SIS model is no longer applicable, but the temporal properties of such a process, i.e. the convergence time, are equally important. The convergence time is a key performance indicator of any algorithm. In this chapter we summarise the main contributions this thesis makes in the general field of dynamic processes on complex networks.

8.1. MAIN CONTRIBUTIONS

In chapter 2 we introduce one of the most studied (and also simplest) models to describe the behaviour of an infection spreading over a network: the Susceptible-Infected-Susceptible model, or SIS for short. Since a complete analytic solution to the SIS model in a general graph using Markov theory is possible, yet not feasible due to the exploding number of states as the network size increases, approximations have been proposed in literature over the years. We discuss two mean-field approximations from literature: the N-Intertwined Mean-Field Approximation (NIMFA) and the Pastor-Satorras & Vespignani HMF approximation, and benchmark the performance of these two approximations in various graph types against the steady-state of the ε-SIS model which we use to define the meta-stable state of the SIS process. The two approximations that we compare are part of a large research interest in the meta-stable state of the SIS epidemic process, that mainly focusses on finding both the epidemic threshold and the average fraction of infected nodes in the meta-stable state.

Although the epidemic threshold is loosely defined as a temporal property of the SIS process —it separates a region where the virus dies out "quickly" from a region where it stays active "for a very long time"—, the temporal properties of the SIS process are less well known. In chapter 3 we discuss the survival time of an SIS epidemic. We derive exact expressions for the average survival time in the complete and the star graph. In the case of the complete graph, we show that the average survival time equals the first term

in the Lagrange expansion of the second largest eigenvalue of the infinitesimal generator. In general, the survival time depends on the initial state of the epidemic. When an epidemic starts in a single node, it will have a shorter average survival time than when it starts with all nodes infected. We show that, for the complete graph and the star graph, the ratio between the survival time starting in a single node, and the survival time starting with all nodes infected peaks close to the NIMFA epidemic threshold. This allows us to exactly *define* the epidemic threshold as the effective infection rate that maximises the ratio between the spreading time starting from a single infected node and the spreading time starting from all nodes infected. Interestingly, the worst-case survival time distribution does not undergo a dramatic change around the epidemic threshold, the exponential tail gradually grows less steep.

In chapter 4 we discuss another temporal property of the SIS process, this time in the situation where two SIS processes compete for the same healthy nodes. It is known that when two mutually immune viruses are active in a single network, one of them will die out. The average fraction of infected nodes, disregarding with which virus they are infected, is the same as in a single SIS process, provided that both viruses have the same effective infection rate. If both viruses are equally strong, i.e. they have the same effective infection rate, each virus has a 50-50 percent chance of pushing the other out. However, if we prevent the viruses from dying out completely by reinfection the last curing node of each type, the dominating virus alternates. Even when one of the two viruses is pushed down to a single surviving infected node, it will claw back the advantage and push the other virus down. The domination time, defined as the duration of a period where one virus continuously has more nodes infected than the other, is shown to be exponentially distributed and to depend on the total number of infected nodes at the start of the domination period. As a consequence, when two viruses start with a single infected node, the first domination period will generally be longer than when a domination period starts from the meta-stable state. Even when one virus is stronger or quicker than the other the weaker or slower virus can still dominate the other virus. The domination time will, however, by asymmetrical. When the difference in strength becomes too large, the weaker virus no longer has the strength to dominate.

In chapter 5 we let go of the Poissonian infection and curing processes. The fact that in the SIS model the curing and infection processes are modelled as Poisson processes makes the SIS process a Markovian process, which puts a number of mathematical tools at our disposal to compute properties of the process such as the survival time and the domination time. However, the infection and curing processes that we encounter in the real world are unlikely to be Poissonian. Various measurements suggest infection processes that are log-normal or Weibullian. We illustrate the effects of a non-Markovian infection process by taking a Weibull inter-arrival time distribution for the per-link infection process. We use the Weibull distribution because it has the exponential distribution as a special case. Via the shape parameter of the distribution we can gradually steer it away from the exponential distribution while keeping the average inter-arrival time constant. Steering the inter-arrival time distribution of the infection events away from exponential towards a more heavy-tailed distribution has a dramatic effect on the epidemic threshold. We then show that keeping the average number of infection attempts during an infectious period constant, instead of the average inter-arrival time of

infection events, the NIMFA equations also hold for non-Markovian SIS. The quality of the NIMFA solution in a non-Markovian setting, however, depends on both topological properties and the distribution of inter-arrival times. Also, NIMFA is no longer an upper-bound of the fraction of infected nodes in the meta-stable state in non-Markovian SIS. Not only the average fraction of infected nodes in the meta-stable state is affected by a non-Poissonian infection process, also the survival time of the SIS process changes. This is caused by the distribution of the spreading attempts over the infectious period of a node. In the case of Weibullian inter-arrival times of infection events, the spreading attempts can be either concentrated towards the beginning of the infectious period or towards the end. Both extremes will lead to shorter survival times, probably as a result of a form of synchronised behaviour.

Moving even further from the classic SIS epidemic model, we enter the realm of epidemic algorithms in chapter 6. Epidemic algorithms use information spreading techniques that mimic those of viruses or gossip spreading in order to diffuse information or, in our case, gather information. We develop two basic algorithms, COUNT and BEACON, that employ a periodic random communication paradigm. COUNT can be used to compute aggregate functions (sum, average, or find minima and maxima) over node specific values; in our specific test case we use COUNT to count the number of nodes in a decentralised dynamic network. To speed up COUNT and other token collecting algorithms, we develop BEACON. In BEACON nodes compete amongst each other to become a beacon node. The beacon node diffuses its presence throughout the network and by that, attracts the piles of tokens that are on random walks through the network. We show that the convergence time of GOSSIPICO, which, in its simplest form, consists of a BEACON layer and a COUNT layer, scales logarithmically in Erdős-Rényi random graphs and scale-free graphs, while the convergence time in high hopcount graphs, such as path graphs and grids, is in the order of the diameter. An automatic restart mechanism makes GOSSIPICO suitable to be used in highly dynamic networks, even when the network falls apart in multiple connected components, the algorithm can adapt.

Since all the different forms of epidemic processes and algorithms take place in networks, it is fitting that in chapter 7 we turn our attention to the networks themselves. We develop a framework to extract graphs from datasets. Our framework is based on the concept of mappings. The mapping decides which relationships or commonalities between entities in the dataset are mapped onto links. We apply our formalism to various large online gaming datasets. The different mappings reveal different kinds of relations between the players in different games and/or communities. An especially sensitive part of the link mapping is shown to be the threshold for how many times a certain condition has to be met before a link between two links is created. Removing links has an influence on the largest eigenvalue of the adjacency matrix. Because the largest eigenvalue is an approximation of the epidemic threshold, we show which greedy algorithm to remove a set of links to minimise the epidemic threshold works best. A heuristic algorithm is needed as finding the optimal solution is an NP-hard problem. All the software we have developed for the graph extraction and analysis is combined in a toolkit as described in Appendix A, and available online.

8.2. FUTURE WORK

Despite the fact that epidemic models have been studied for a while now, the research field of epidemics on graphs still contains large parts of uncharted territory. We roughly divide the future work on SIS epidemics in three directions: improving our approximations, understanding the applicability of the (non-Markovian) SIS model to real-world spreading phenomena, and expanding the SIS model.

To start in the first direction, we still lack a proper understanding of why mean-field approximations are accurate in one graph and far off the mark in others. The short, or perhaps naive, answer is that if there is a correlation between the infection probability of nodes, mean-field will be inaccurate. The longer, and more complex, answer is that we do not know how the topology and inter-arrival time distributions contribute to the correlation between infection probabilities. A quantitative spectral or topological network property (or set of properties) should be found that can be a measure for the expected accuracy of mean-field approximations. At the very least we should be able to take a general graph and decide whether mean-field approximations to describe the SIS process will work well or not. In addition to approximations to the average fraction of infected nodes in the meta-stable state, we need approximations to the survival time of an outbreak in general graphs. Work in this direction would extend the results presented in chapters 2 and 3.

In chapter 5 we showed that replacing the exponential inter-arrival time distribution with a Weibullian one has a great impact on the SIS process. The move away from Markovian SIS is needed to better model real-world spreading phenomena such as content propagation in social networks. Research has shown that (online) human activity is bursty: long periods of inactivity are followed by short periods of intense activity. This kind of infection processes cannot be accurately modelled by Poission processes. Work has to be done to better understand the actual infection and curing processes of real spreading processes, both in terms of distributions and typical rates. At the same time, we need to understand the properties of the non-Markovian SIS process on various graph types to model the reach of content, and to learn how to manipulate networks to improve or counter spreading processes.

The third direction of future work on SIS models is to add detail to both the process and the network. The classical SIS model assumes that all nodes and links are equal in terms of infection strength and curing speed. A heterogeneous SIS model is needed to treat the more realistic case of weaker and stronger nodes. Research in coupled networks and different inter and intra community infection rates is now becoming hot. Even more challenging is the situation where the nodes and links become aware of the infection and start to adapt: healthy nodes break links with infected nodes. Finally, we think that the topic of virus competition, including mutating viruses and super infection (virus populations live inside other populations of infected nodes), will receive a lot of attention in future work on epidemic processes on graphs, expanding the work presented in chapter 4.

Just as epidemic spreading in graphs is an established research field, epidemic algorithms have received solid attention over time. One of the main challenges still remains to develop an efficient way of detecting convergence of the algorithm. The convergence detection of GOSSIPICO, for example, can lead to a big gap between actual

convergence and detection. In the specific setting of epidemic algorithms designed to compute graph metrics, it remains unclear which metrics can be computed efficiently using a decentralised epidemic algorithm. Such algorithms will be very beneficial in the characterisation of very large distributed (social) networks. Our algorithms developed in chapter 6 and the toolkit we presented in Appendix A is only a very limited start to such an approach. Ultimately, if the behaviour of the SIS model can be described or accurately approximated in terms of (spectral) graph properties, a simple SIS simulation on a network could be enough to compute properties of the graph. However, before the SIS model can serve as an algorithm in itself, all the problems stated above would have to be solved.

A

A Toolkit for Real-time Analysis of Dynamic Large-Scale Networks

Networks are used in many research domains to model the relationships between entities. We present a publicly available toolkit to extract graphs from datasets or data streams and to analyse their properties. The graph extraction is based on a set of rules that define the links between entities in a set or stream of self-contained events involving sets of entities. As the extracted graph is dynamic and, moreover, can be spread over multiple machines, we include the class of gossip algorithms to analyse them. In addition, the toolkit also contains algorithms to compute metrics of static snapshots of the dynamic graph.

A.1. Introduction

Many research domains use networks as modelling tools. Networks can represent vastly different objects including physical infrastructures such as rail, road and waterways (e.g., [2]), flight routes and shipping lanes (e.g., [3, 4]), but also sewage systems and power grids (e.g., [5]), and, of course, the internet. Networks can be constructed from financial transactions [6], friendship or collaboration relations among individuals [7], (P2P) peers exchanging data (e.g., [130]), sports players or online gamers that have played on the same team [8], and more abstract things such as functional brain networks where the nodes in the network are brain regions that share a link when they show correlated activity [9], or co-purchase networks where nodes are items in a shop that share a link when they were purchased together [10]. In short, everything that can be represented as a collection of entities that have a relation can be modeled as a network.

The study of complex networks currently faces two main challenges: 1) The size of the studied networks has grown enormously. Especially online communities such as social networking sites have exploded in size and easily contain hundreds of millions of nodes. In addition, the availability of machine-readable data has enabled companies

and researchers to construct increasingly large networks.

Very large networks are a problem for two reasons. First, in some cases the networks are so big that it is no longer possible to process them in a single machine. Often, the networks are not even stored in a single location, but are distributed over multiple servers in multiple locations. Second, the time complexity of the algorithms used to calculate certain properties does not scale well with the network size.

Simple network properties that are limited in scope to single nodes are not challenging to calculate. When our scope encompasses the entire network, however, the real challenges begin. Calculating how many of the shortest paths in a network go via a particular link (link betweenness), for example, is computationally expensive; other properties, such as the isoperimetric constant or the facility location problem, are even NP-hard to determine.

2) The second challenge that network science faces is that of network dynamics. Real networks are not fixed in time; they are constantly changing. In an online social network, for example, new users sign up and existing users leave, thereby changing the number of nodes. At the same time users make new friends or break old ties, which changes the topology. If we are not just interested in the links between user profiles, but only consider the users to be present in the network when they are logged in, the network changes are quick and substantial. Since a social network is very large, it is currently impossible to accurately track changes in network metrics as a function of time. Traditional methods of calculating properties of networks break down for very large and dynamic networks.

Before graph analysis can begin, there is the challenge of extracting a graph from a dataset or a data stream. We have developed a suite of tools, which is publicly available [131], that can be used for all steps needed to analyse the relationships between entities in datasets or data streams. Although other tools exist, such as Jung, Igraph, Pajek, and Cytoscape for static network analysis, Gephi and NetLogo for visualisation, DeSiNe [132] for traffic dynamics, and PeerSim for simulating gossip algorihtms, the strength of our suite is that all the components are in one place, using the same underlying data structures, and that we consider dynamic networks.

We will describe our toolkit in four sections: Section A.2 will provide an overview of the elements in the toolkit and how they can be combined for real-time analysis of dynamic graphs. In Section A.3, the graph extraction from either a dataset or data stream is introduced in detail, after which in Section A.4 we include the class of gossip algorithms as a way to compute properties of the extracted graph. Finally, in Section A.5, we briefly describe the available tools in the toolkit to do static graph analysis on snapshots of the dynamic graph, and Section A.6 discusses some future work and conclusions.

A.2. OVERVIEW TOOLKIT

The main idea behind the toolkit is to combine three parts, graph extraction, static and dynamic graph analysis, to analyse data streams. The analysis is illustrated in Figure A.1. The data stream that we want to analyse is passed through a filter that extracts a dynamic graph based on a set of rules. This set of rules determines what will become nodes in the graph and what will become links. We call this mapping a dataset onto a graph. The details of how to extract graphs from a dataset and how different mappings influence the resulting graph are explained in more detail in Section A.3. The novel application

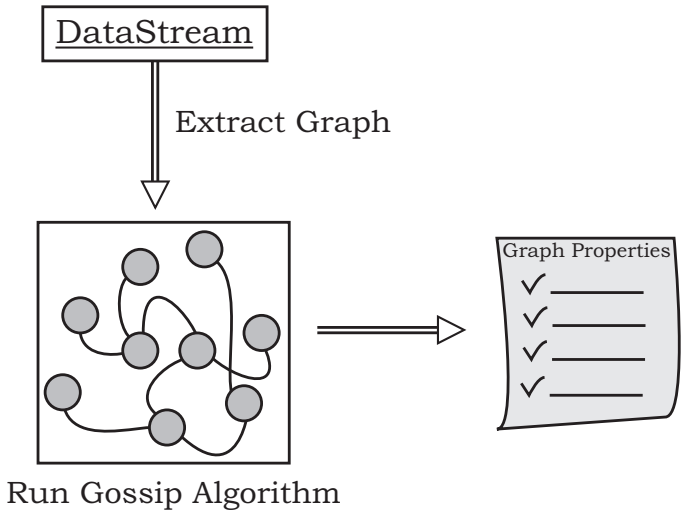


Figure A.1: Main steps in the data stream analysis. The data stream is passed through a filter to extract a dynamic graph containing the relations between entities of interest in the data stream. On this large and dynamic graph a gossip (or alternative) algorithm is ran to compute properties of interest.

here is that instead of mapping a fixed or static dataset onto a static graph, we apply these techniques to a data stream. The result of mapping a data stream onto a graph is not a static graph but a dynamic one. The filter continuously outputs graph dynamics: nodes and links are joining or leaving the graph depending on the data that is streamed through the filter.

We use a gossip algorithm to continuously compute or update those graph properties that we are interested in, although other dynamic algorithms could also have been used. Gossip algorithms have proven to be ideal candidates to perform computations in large, dynamic and distributed systems. In Section A.4, we will introduce gossip algorithms in more detail and explain how we simulate gossip algorithms. We will give an example algorithm to compute averages or to find minima and maxima over node values.

A.3. GRAPH EXTRACTION

Graphs are commonly used as models to study the properties of various different physical or non-physical problems. A communication network, for example, can consist of a group of computers connected via Ethernet links. For such a network we can then compute how many links can be removed without losing connectivity. In this example, the step from the physical world of computers and cables to the more abstract graph model is very intuitive. The computers are our nodes, and the cables are our links.

When we study less tangible networks, such as social networks or implicit networks in datasets, we have to decide what are going to be the nodes in the graph and what are going to be the links. More formally, a dataset D is mapped onto graph G via a mapping function $M(D)$. A simple undirected and unweighted graph $G = (\mathcal{N}, \mathcal{L})$ consists of a set $\mathcal{N}$ of N nodes and a set $\mathcal{L}$ of L links. In a weighted graph, a link weight w is associated

to every link in $\mathscr{L}$. In a directed network, a link between two nodes also has a direction.

A.3.1. EXTRACTION RULES

A *mapping* is a set of rules that define the nodes and links in a graph. Entities are often mapped to nodes, while relations between entities are mapped to links. Entities are usually readily identifiable as persons, events, or objects and are therefore intuitively mapped onto nodes.

Mapping relations to links, however, is more challenging. Entities can be related to each other in many different and often subtle ways and due care should be given to which relations produce insightful graphs. For example, some relations between entities may be the result of chance, while others have a clear origin. The difference between random and meaningful relations is often expressed by a notion of strength. Strength, represented by a link weight, adds another dimension of complexity to representing and understanding the characteristics of a network.

Our tool to extract graphs from datasets was developed in the context of identifying implicit social networks in the game data from online social games [8], but is not restricted to be used in the gaming or social network setting. Graphs can be extracted from every dataset that consists of a collection of self-contained events that involve a group or groups of entities. In [8], the self-contained events were matches played in an online game and the groups of entities were two (not necessarily equally sized) teams of players. The data was crawled from a website and then parsed into a *Match* object containing meta-data such as the start and end times of the match, and score-related information.

Graphs were extracted from the *Match* objects based on two different sets of rules: global rules and local rules. Global rules apply to the dataset as a whole: for example, a link exists between two players that were present in the same game. Local rules apply to individual potential links. For example, while a global rule dictates that two players have a link if they have been in a match together, a local link can dictate that that match must have had a duration of at least 20 minutes.

Both local and global rules can either be simple, or compound. Simple rules consist of a single requirement such as “in the same game together” or “played in the afternoon”, whereas compound rules are logical combinations of simple rules. The rules can be specified in a configuration file in bracket notation, for example ([link won] & [duration > 20]) | ([link lost] & [duration < 10]).

Although all our examples are in the setting of online games, the same machinery can be used in different settings. For example, the self-contained events could be items in a shopping cart, or people on a bus, etc. The flexibility of our tools for graph extraction enabled us to study the effect of different mappings and thresholding on the social networks extracted from gaming data.

A.3.2. LINK SET

Every time a link is formed as a result of the extraction rules, it is stored in a link set. However, we might want a link to be formed a few times before we want to consider it as a valid link in the graph. To achieve this, we count how many times a link is formed (we increment the link weight every time it is formed) and only add it to the graph when the link occurred more than a threshold value. In this case, the link set can be seen as an in-

termediate step between the data stream and the graph. The link set is implemented as a balanced search tree, which is a data structure that is designed to have approximately the same number of nodes to the left and right of each node. The fact that a tree is balanced guarantees that a search in the tree will be of $O(\log n)$, where n is the number of entries in the tree. The balanced search tree that we use in the link set is an implementation of the red and black tree [133].

Each link that is added to the tree has a unique value associated with it that serves as an identifier for the link but also enables links to be ordered lexicographically. The link id is simply the binary concatenation of the ids of the source and destination nodes. If links are bidirectional, the link id is the concatenation of the bigger and smaller node ids of the endpoints.

Every time a link is added to the link set, its link id is compared to that of the root node. If it is greater than that of the root node, it is compared to the right child of the root node, otherwise to the left child. This continues until either (i) a node is found that has the same value as the link to be inserted, or (ii), it is smaller (larger) than the current node but larger (smaller) than its left (right) child. In the first case, the link weight of that link will be incremented; in the second case, the link is added to the tree in the current position and the tree is rebalanced.

Since the link set can take up quite some memory, especially when it contains hundreds of millions of links, it is not unlikely that the tree runs out of memory. The solution to this problem depends on whether the tree is used to store the links from a static dataset or a data stream. For the static case, the link set can be stored to disk when it reaches a certain size. After storing the link set to disk it is cleared and can be filled again until it reaches the predefined size again, after which it is merged with the tree on disk. In this fashion, the stored link set can grow very large. The stored link set can be read from disk once to extract the graph, possibly while applying a threshold on the link weight. By writing the link set to disk it is also easier to create graphs from a link set using different threshold settings later.

If the link set is the result of streaming data through the filter, writing an intermediate file to disk if memory consumption grows too large is no longer an option because of the long disk access time which makes merging the tree in memory with the tree on the disk too slow. Also, since we want to have exact knowledge of when a link is added to the network, we would have to merge the tree in memory and the one on disk after each link addition. This effectively reduces the link set to exist on disk only. As the entire link set has to be kept in memory, we either have to use a machine with a very large amount of memory, or we have to spread the link set over a number of machines in a computation cluster.

We have implemented a tree structure of partial link sets that allows the link set to be distributed over multiple machines. Each individual partial link set is used to store a part of the links. The simplest way to spread the possible links over the link sets is to divide the space between the highest and lowest known link evenly between the number of link sets. However, since it is not known beforehand how many links will be stored in the link set, the distribution over the partial link sets might become skewed over time. If the links are too unevenly spread over the partial link sets, they have to be rebalanced to make sure that all link sets grow at approximately the same rate.

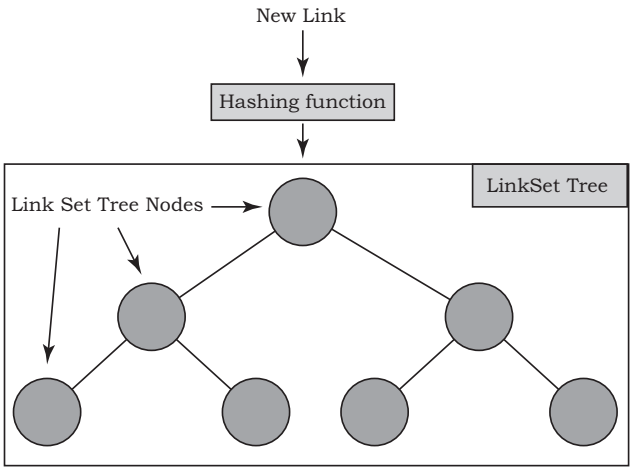


Figure A.2: Schematic overview of inserting links in the distributed link set. A new link is first passed through a hashing function to get a hash of the link. The link and hash are then offered to the centre node of the LinkSet tree. Based on the hash, the link will end up in one of the partial link sets in the tree.

Our approach to spread links over the partial link sets is centred around a hashing function. An ideal hashing function has a uniform output probability for every input. If each of the partial link sets is responsible for a part of the output range of the hashing function, a new link has equal probability of being assigned to any of the partial link sets. In this way, the distribution of the links over the partial link sets will be uniform. The implementation of our tree of partial link sets is based on the *LinkSetTreeNode* as illustrated in Figure A.2. First the link id of a new link is hashed. The link and its hashed id are offered to the central node in a LinkSetTree. From the central node the link is passed on until the *LinkSetTreeNode* where the link should be stored is found.

The pseudo-code in Algorithm A.1 illustrates how the correct *LinkSetTreeNode* is found. Each *LinkSetTreeNode* has a lower and upper limit of link ids that it is responsible for. It also has a pointer to a smaller and larger *LinkSetTreeNode*. When a link is offered to the node, it checks whether the (hashed) id is smaller or larger than the lower and upper limit, and, if it is, it passes the link on to the smaller or larger node. If the hashed link id is within the range of the current *LinkSetTreeNode*, it will send the link to its partial link set via a TCP connection.

The distributed link set is essentially a client-server application where the server uses the LinkSetTree to find out which client should store which link. The clients use an ordinary link set as described above to store the links. Clients do not all have to have the same amount of memory available. Each client can indicate the maximum number of links it thinks it can store when it contacts the server. The server can use this information to distribute the output range of the hash function in such a way that each client is responsible for a part of the range proportional to its storage capacities.

In addition to a distributed implementation of the link set, the partial link sets can also be exploited to speed up the storage in a single large memory machine. Figure A.3 shows a measurement of the running time of adding a billion links to the link set in three

```
Object LinkSetTreeNode()
    Fields:
    int lowerlimit, upperlimit
    LinkSetTreeNode() smaller, larger
Method insert(link)
  if link.id < lowerlimit then
    smaller.insert(link)
    return
  end if
  if link.id > upperlimit then
    larger.insert(link)
    return
  end if
  send link to link-set
return
```

Algorithm A.1: Link insertion in distributed link set.

different scenarios as a function of the number of partial link sets. Our test machine featured 16 logical cores and 48 GB of memory. In this experiment the added links are randomly generated. The source and destination nodes are drawn uniformly from a set of integers between 0 and $N-1$, where N is the number of nodes in the graph and is chosen to range from 10^4 to 4×10^4. Clearly, the larger N is, the larger the number of possible unique links. In our experiment the number of recorded unique links was approximately 5×10^7, 2×10^8 and 5.7×10^8. Also shown in Figure A.3 is the rate of link additions. A link addition in this case is simply an offered link, of which there were 10^9 in every experiment.

Figure A.3 shows that using more partial link sets leads to lower running times. The reduction in running time is larger when the number of unique links in the tree is larger. This is caused by the computational complexity of finding the right insert point in the search tree for larger trees. Figure A.3 also shows that for a small number of unique links in the tree, the overhead of sending the links over TCP outweighs the computational efficiency of smaller trees. For example, there is virtually no benefit of using 16 in stead of 8 partial link sets for a graph with 50 million unique links.

A.3.3. GRAPH DYNAMICS

Going from a link set to an evolving graph involves two opposing processes: growing and shrinking. The growing dynamics are more easily extracted from the link set. Every time a link is found in the streamed data, it is inserted into the link set as described above. If the link was not yet in the link set, it is added, otherwise, the weight of the link already present in the link set is incremented by one. While incrementing the link weight, it is easy to check whether it goes from just below the threshold to over the threshold, and, if it does, to add the link to the graph. If either of the nodes adjacent to the link is not present in the graph yet, it is also added. This process will, however, lead to an ever growing graph. It is probably desirable to include a mechanism to remove links that are

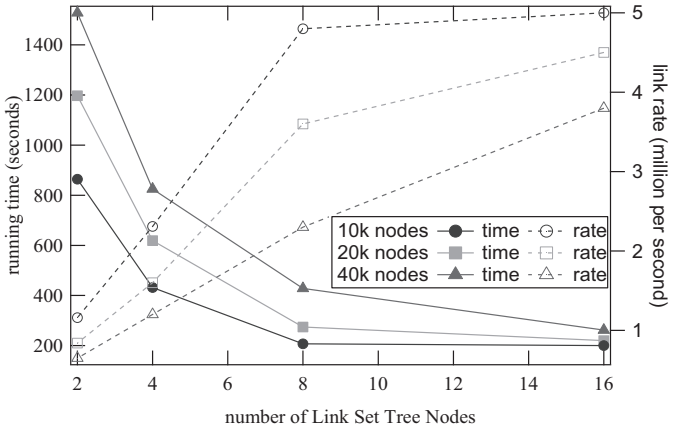


Figure A.3: Running time for the addition of 10^9 links to the distributed link set as a function of the number link set tree nodes for three different random graphs. The number of unique links in the tree is approximately 5×10^7, 2×10^8 and 5.7×10^8 for the three curves.

no longer deemed important.

There are various options available for shrinking the graph and the link set. First of all it is possible to periodically remove all the links from the link set whose link weight is smaller than a certain threshold value. A periodic cleaning of the link set reduces the amount of memory that is used by links that were formed only once or twice, probably as the result of a chance encounter. If the cleaning threshold is lower than the threshold used to create the graph, however, the periodic cleaning does not lead to links being removed from the graph.

Another way to introduce shrinking is to use the time information of link creation. It is possible to use (i) full time information: each increment has a time stamp, (ii) partial time information: only the time stamp of increments within a fixed window from the last increment are kept, and possibly the maximum number of increments ever recorded in that interval, or (iii) minimal time information: only the time stamp of the last increment is used. By keeping this time information, it is possible to apply global rules such as "a link exists between players that played more than 10 times together within any 24 hour period." Obviously, the more time information is kept, the more storage is needed. Even when storing time information, shrinking of the link set and graph will have to be done periodically and involves variations on the theme of removing links that have not been formed the last x time units.

A.4. DYNAMIC GRAPH ANALYSIS

In order for the dynamic graph resulting from the graph extraction to offer insight in the relationships between entities in the data stream, graph properties should be computed. Ideally, algorithms that update a metric following graph dynamics, instead of recomputing the metric, are used. However, this is not always possible. Moreover, if the

```
Data: array order containing node ids
while true do
    shuffle order
    for node n in order do
        activate n
    end for
end while
```

(a) Synchronous simulation

```
Data: TimeLine T
while true do
    take ticket from T
    activate owner
    set new ticket time
    insert ticket in T
end while
```

(b) Asynchronous simulation

Algorithm A.2: Synchronous and Asynchronous simulation mode.

link set (and graph) is spread out over multiple machines, both distributed and dynamic algorithms are needed. Gossip algorithms have proven to be useful to determine properties of large, distributed and dynamic graphs, which makes them ideal to analyse the graph extracted from a data stream.

In a gossip algorithm, nodes typically select one of their neighbours periodically to send, request or exchange information with. Depending on whether nodes send, request or exchange information, the process is called Push, Pull, or Push-Pull. The timing of nodes becoming active can either be synchronous or asynchronous. In the synchronous case, each node becomes active once during a time window. Nodes will need to be able to loosely synchronise to prevent too much misalignment of time windows. In the asynchronous case, a node typically becomes active when an exponentially distributed timer expires. In other words, the node's activity follows a Poisson process.

In a gossip algorithm, nodes send a limited amount of information and keep very little state information. Nonetheless, these algorithms are capable of delivering global network information to all nodes without central control while being robust against message delivery failure. Data aggregation is a particularly fruitful application of gossip algorithms. The goal of aggregation algorithms is to inform every node in the network about some global property without the aid of a central authority. In the setting of a sensor network, this could be the average reading over all sensors, or the maximum or minimum reading. The work of Boyd [134] contains a wealth of references on distributed averaging. Also, see [135] for applications of gossip algorithms in signal processing.

The simple format of gossip algorithms makes them very suitable to be deployed in a general setting. Our toolkit offers such a setting where a set of data structures and functions is offered to simulate the behaviour of the algorithm in a somewhat idealised setting. The simulator offers both a synchronous and asynchronous mode of operation. The pseudo-code of the synchronous simulation mode is given in Algorithm A.2a. In the synchronous mode, the node ids are stored in an array which is shuffled each iteration. During the iteration, the nodes are processed in the order of which they appear in the array. The shuffling and iteration-wise operation of the simulation ensures that every node will be active only once during an operation window and at the same time it mimics the behaviour of loosely synchronised clocks at the nodes. Without shuffling the nodes will be processed in the same pattern every iteration. Alternatively, the node array can be shuffled after a particular number of cycles.

In the asynchronous case, the simulator has to keep track of the expire times of every

node's internal timer. These times are stored as tickets in a time line. A ticket consists of a time value and an owner value. The owner value indicates which node issued the ticket and the time value indicates when the timer of that node expires. The time line is implemented as a balanced search tree. Storing the expire times in a tree makes it easy to chance the distribution of the waiting times. The simulator continuously takes the ticket with the smallest time value from the tree and makes the node perform its action. When the node is done, it issues a new ticket to be inserted in the time line with a time value that is an exponentially distributed random value removed in time from the current moment. The tree structure ensures that insert and remove operations take $O(\log N)$ time, where N is the number of tickets in the time line.

The network structure that is used in the gossip simulation is designed to make it easy to facilitate network dynamics. All nodes are stored in a linked list, making additions easy and quick. Removals are slower since the node to be removed has to be located first. In case of heavy churn, nodes could also be stored in a search tree, just like the tickets in the asynchronous simulation mode. Each node internally also stores a linked list with pointers to its neighbours. Just as node dynamics is optimised for growing networks, the linked lists for the links between nodes also favours adding links in terms of running time. For heavy link removal it could also be better to use a tree structure. Every time a node tries to send a message over a link, it has to check whether the node the link points to is still present. If not, it should remove the link.

In addition to the timing and network dynamics the simulator also offers a structured way of implementing a gossip algorithm. Every class implementing the node behaviour interface can be loaded as a gossip algorithm. The simulator offers two ways of performing the actual contact between nodes: the direct method and the message method. In the direct method, the active node can directly interact with the procedures and fields of the behaviour code of its neighbours. This makes the interaction between nodes quicker, but it requires all the nodes to be in a single machine. In the message method, the active node prepares a message containing all the values that it wants to exchange with its neighbour and passes that message to the simulator. The simulator then delivers the message to the target neighbour and takes care that the reply of the target neighbour is again delivered to the active node. Although this requires a few intermediate steps, this method allows nodes to be spread over different machines. Spreading nodes over different machines can be beneficial when the network grows very large.

A basic example of a gossip algorithm that can be used to compute properties of the large dynamic graph that is the output of the graph extraction filter is Gossipico [136]. Gossipico can be used to sum, average or find the minima and maxima over node related values. A typical example function of Gossipico is to count the number of nodes and links in the graph and to compute the maximum degree. These functions can be combined in Gossipco, but it is also possible with the current simulation framework to run multiple gossip algorithms in the same network.

A.5. ADDITIONAL FEATURES

An alternative to the dynamic graph analysis described in Section A.4 is to take snapshots of the dynamic graph and perform traditional static graph analysis. Depending on the metric of choice, a dynamic/gossip algorithm may not realise a significant speed-

up compared to a static algorithm. To minimise the need of additional programs, our toolkit contains a set of implemented algorithms to compute various network metrics. The spectral metrics (eigenvector centrality, algebraic connectivity, spectral gap) rely on the eigenvalue decomposition of the CERN Colt libraries in some cases. In other cases an implementation of the power method is used.

The hop-count related metrics, such as the average shortest path or the eccentricity can be computed individually or as a by-product of computing the node betweenness. The betweenness code is an implementation of Brandes' work [137] on betweenness algorithms. The coreness algorithm is an implementation of Batagelj [138]. Other supported operations are the computation of the number of connected components and various ways of exporting networks including Cytoscape and Pajek formats.

Another feature that is offered by the graph analysis tools is the ability to generate synthetic networks according to numerous different network models. The code to generate Erdős-Rnyi random graphs and preferential attachment graphs is based on work by Batagelj and Brandes [105].

A.6. CONCLUSION

We have presented a toolkit that offers all necessary tools to analyse datasets or data streams using network science. The links between entities in the dataset or data stream can be extracted using different types of link rules. These links are stored in a link set that can be spread out over multiple machines if the link set becomes large. When a data stream is analysed, the link set and extracted graph will be dynamic. In order to analyse the properties of a large dynamic graph that is possibly spread out over multiple machines, we have offered the class of gossip algorithms. The toolkit comes with a simulation environemnt for gossip algorithms as well as traditional metrics to study snapshots of a dynamic graph.

Although the dynamic and distributed link set is fully implemented in the toolkit, as well as a sample gossip algorithm, more work is needed to determine which graph metrics can be computed using gossip algorithms and which cannot. For those metrics that cannot be computed using gossip algorithms, new (preferably dynamic) algorithms are needed.

We have performed basic measurements of the rate at which links can be inserted in the link set as a function of how many partial link sets are used, but how large the link set can realistically grow remains unexplored. Future work is needed to determine the absolute performance limits of our toolkit.

B

SISS

SISS is an event driven simulator of the susceptible-infected-susceptible (SIS) spreading process on a network. In the standard, Markovian SIS process, each node can be in one of two states: susceptible and infected. Infected nodes spread the infection to their susceptible neighbours. Spreading of the infection between two nodes is modelled as a Poisson process with infection rate β. For each node pair containing at least one infected node, an independent Poisson process modelling the arrival times of spreading events exist. A spreading event simply means that the infection tries to spread from an infected node to a neighbour. Such an attempt can either be successful, if the neighbour was susceptible, or unsuccessful, if the neighbour was a infected itself. The simulator will simulate the arrival of every spreading attempt, regardless of the state of the network; if two adjacent nodes are both infected, the arrivals of spreading attempts of both nodes (so two Poisson processes on the link, one in either direction) will be processed in the simulator.

When a node is infected, it will cure with rate δ. Curing events also follow a Poisson process. When a curing event happens the node becomes healthy, yet susceptible to the infection. As the curing process stops when the node is cured, it reduces to an exponential infection time per node: when a node becomes infected it will become healthy again after spending an exponentially distributed time infected.

The spreading and curing events are the drivers in the simulation.

B.1. Simulation

The heart of SISS is formed by a balanced search tree called TimeLine. In the TimeLine all future events are stored. As explained in Section B events in the SIS process can be spreading events or curing events. The TimeLine can also contain simulation events used to perform measurements and administrative tasks in the simulator itself, these events will be treated in Section B.1.2. Events are stored in the TimeLine in the form of tickets. Tickets are continually take from the TimeLine and processed by ticket handlers. During the processing of the tickets new events may be generated and stored as tickets in the TimeLine again. When there are no tickets left in the TimeLine the simulation ends.

B.1.1. TICKETS

A ticket is a simple object containing four fields that completely describe the event. The four fields are:

1. Time: the time at which this event will occur.

2. Owner: the entity that generated the event. This can either be a node (identified by its ID) or the system. System tickets are used to perform all kinds of tasks within the simulator as explained in Section B.1.2.

3. Action: indicates what kind of action has to be performed. Spreading events have a positive action where the action corresponds to the node id of the node that has to be infected, curing events have a value of -1. In the case of a system ticket the a negative (non -1) action encodes a specific action to be performed.

4. Level: the level of simulation. The simulator allows loosely synchronised simultaneous simulations of multiple outbreaks on the same network. All outbreaks have the same spreading and curing characteristics.

B.1.2. TICKET LISTENERS

As mentioned in Section B.1, at the heart of the simulation process tickets are continuously taken from the timeline and processed. Spreading and curing tickets are always processed by a ticket handler, the most common system tickets are processed by the simulator directly. It is, however, possible to register custom system ticket handlers as explained in Section B.1.2Using ticket handlers makes it very easy to implement behaviour different from the classic SIS process. Spreading and curing tickets are processed by different ticket handlers.

SPREADTICKETHANDLERS

Spreading ticket handlers must implement the SPREADTICKETHANDLER interface. The standard SIS behaviour is implemented in the SISCURETICKETHANDLER and the SISSPREADTICKETHANDLER classes. For a spread ticket, the owner is usually the node that is infected, however, it is possible that a system ticket is used to infect nodes. The system can infect nodes to set the initial configuration or to otherwise spontaneously infect nodes.

In the standard SISSPREADTICKETHANDLER the processing of a spreading tickets involves four steps.

1. The infectious state of the owner is checked (if the owner is not the system) as a sanity check of the system. If the owner is not infected the simulator throws an error and halts to ensure that no spreading occurs from a healthy node.

2. The node that will be infected by the owner, the target node, is encoded in the action field. If the target node is infected, nothing more has to be done. If the target node is healthy, however, it will become infected. The infectious state of all nodes is kept in the SIMULATOR as an array of booleans where true indicates infected and false indicates healthy. When a node becomes infected a random

cure time is drawn for that node. Cure times are drawn in the *getCureTime* method of the SIMULATOR from the cureDist RANDOMVARIABLEGENERATOR. The cure time for each node is stored in the SIMULATOR class in the cureTime array. After the cure time for the newly infected node has been determined, a spread time for each link is generated. The spread times are drawn in the *getSpreadTime* method of the SIMULATOR from the spreadDist RandomVARIABLEGENERATOR. If the spread time for a particular neighbour is smaller than the cure time of the infected node, a ticket is added to the TimeLine to infect the neighbour at that point in time. If the spread time is larger than the cure time of the infected node nothing will happen.

3. If the target node was healthy before the ticket was processed, the target has become infected as described in step 2. The resulting change in the viral state of the network will be processed by calling the *updateNetworkState* method of the SIMULATOR class. The details of logging and state updates are described in more detail in Section B.1.5.

4. The final step in processing a spreading ticket is to draw a new spreading time for the target. Because the spreading process per link in the normal SIS model is a Poisson process, spreading attempts arrive continuously with an exponential inter arrival time. The new spreading time is again compared to the cure time of the ticket owner and a ticket is inserted in the TimeLine if the spreading time is smaller than the cure time.

CURETICKETHANDLERS

Cure ticket handles must implement the CURETICKETHANDLER interface. The standard SIS cure ticket handler is implemented in the SISCURETICKETHANDLER class and consists, in its simplest form, of only two lines of code:

1. The viral state of the cure tickets' owner is set to healthy.

2. The *updateNetworkState* method in the SIMULATOR class is called.

In addition to these two steps the SISCURETICKETHANDLER also performs a sanity check to verify that the owner of the cure ticket is really infected. Also, if the absorbing state of the SIS model is removed, the node is reinfected if it was the last infected node.

SYSTEM TICKETS

A few system tickets are processed directly inside the SIMULATOR class. These tickets represent events that are likely to occur in many different simulation settings.

EpsilonEvent Epsilon events are spontaneous infection events that occur in the ϵ-SIS model. An EpsilonEvent is processed by infecting a random node in the network. After infecting a random node, a new EpsilonEvent ticket is added to the TimeLine.

StartLogging The StartLogging event initiates the logging of the viral state of the network. Only one StartLogging event should be processed per simulation and an error will be thrown when a StartLogging event occurs while logging has already started. Care should be taken that a StartLogging event is put in the TimeLine

either at the start of the simulation or during the simulation, as logging does not start automatically. See Section B.1.5 for more information about logging of the network state.

SampleEvent A sample event will cause the simulator to fire a SimulationEvent of the type SampleEvent. All registered SimulationEventListeners will be called in the order in which they are registered and the SampleEvent ticket is passed on. At the return from the SimulationEventListeners a new SampleEvent ticket is created with a ticket time equal to the current time incremented with the sampleInterval. For simulations that need sampling, the first SampleEvent ticket should be added by the NETWORKINITIALISER. See Section B.1.4 for more details about the NETWORKINITIALISER.

VisualisationEvent A visualisation event signifies that the the visualisation has to be redrawn. See Section B.1.9 for further details on the visualisation capabilities.

DeathRay The DeathRay event immediately terminates the simulation in a similar fashion as a stopping rule will terminate the simulation. For more on stopping rules see Section B.2.1.

CUSTOM SYSTEM TICKET HANDLERS

In addition to the standard system described in Section B.1.2 it is possible to add custom system tickets to the TimeLine. All system tickets should, however, have a unique action field. To ensure this, the action field of a custom system ticket is determined when the custom ticket handler is registered. A custom ticket handler has to implement the CustomTicketHandler interface and be registered in the simulator class by passing it to the RegisterCustomSystemTicketAction method. This method will return an integer action id that should be used for all tickets that are to be handled by this CustomTicketHandler.

B.1.3. RANDOM NUMBER GENERATION

Random number generators are all implementations of the abstract RandomVariableGenerator class. The RandomVariableGenerator class contains the Mersenne Twister random engine from the colt library as the basis for all random number generation. The Mersenne Twister is initialised in the constructor of the abstract class and takes as its seed the current system time. After initialisation, the thread is paused for a random number of milliseconds to avoid multiple instances of the RandomVariableGenerator to share the same seed value. Any class extending the abstract RandomVariableGenerator class can be used to generate the spreading and curing event times in the simulator. Investigating non-Markovian SIS processes is as simple as changing the spread or cure time generator. The following distributions are currently implemented.

- Exponential
- Weibull
- Normal (Gaussian)
- Uniform
- Discrete Uniform

B.1.4. NETWORK INITIALISER

The network initialiser is the first piece of code called after the simulation is started in the run method of the simulator class, if a class implementing the NETWORKINITILISER interface is registered. The NETWORKINITIALISER has only one required method: setInitialState. In this method, which requires the Simulator object as an argument, anything that has to be set at the beginning of the simulation can be set. The most common things to set in the network initialiser are things like the initially infected nodes but also more elaborate things can be done efficiently from this point in the simulator. Especially the combination of a custom result processor (see Section B.1.6) and network initialiser, possible with an event listener (see Section B.1.7) can be very powerful. The network initialiser can also be used to reset the simulation to its initial state during the simulation. Currently implemented network initialisers are:

- SimpleInitialiser
- SimpleSampledInitialiser
- WeibullInitialiser
- CustomDistributionInitialiser

B.1.5. LOGGING AND STATE UPDATES

After each network change, that is, after a successful infection attempt or a curing event, the updateNetworkState method is called. This makes sure that the infected node counter is updated accordingly and that the event counter is incremented. If logging is enabled, the time the network has spent in the previous state is logged. The network state in this respect is nothing more than the number of infected nodes.

After these task have been taken care of, the additionalStateUpdater is called. Any additional analysis or date logging after a state change can be done in this place. An additionalStateUpdater has to implement the additionalStateUpdater interface. For the classical SIS process, the only implementing class is the SIS_ASU. This state updater fires a stateChange event and, if the number of infected nodes is zero, an extinction event.

B.1.6. RESULT PROCESSOR

If, after processing a ticket, the TimeLine is empty, no more events are scheduled for the current simulation. In short, the simulation has finished. Before returning from the run method, the result processor (if one has been registered) will be called. A result processor has to implement the SimulationResultProcessor interface and can be registered with the Simulator via the setSRP method. The SimulationResultProcessor will be called with the simulator object as a parameter so that any kind of processing on the internal state of the simulator can be done. A simulation result processor is especially handy when operating the simulator in GUI mode (see B.2.1). In other settings, the internal state of the simulator object will be available to the calling code after the run method terminates. Although possible, it is not recommended to call the run method of the simulator from within the result processor since this will lead to nested calls of the run method and, eventually, to a stack overflow.

B.1.7. Simulation Events and Listeners

Simulation events are the standard way in the simulator to let custom parts of code know that a particular event has happened. Ticket handlers, as well as additional state updaters can use simulation events to trigger a reaction. The ticket containing the event that triggered the firing of a simulation event is always passed on with the simulation event. Currently, a simulation event can be of the following types:

Node Infected This event can be fired to indicate that a node became infected. Usually this event is used to track the state of a single node, or a small number of nodes. Otherwise the State Change event is more convenient.

Node Cured This event can be fired to indicate that a node has cured. Just as the Node Infected event, this event is usually used to track the state of a single node, or a small number of nodes. To track the state of the entire network, the State Change event is more convenient.

State Change This event can be fired to indicate that a node has either become infected or has cured. It is fired by the SIS_ASU (if required).

Reinfection This event can be fired to indicate that a node was reinfected to prevent the virus from dying out. It is fired by the SISCureTicketHandler.

Sample Event This event is fired by the standard system ticket handler and indicates that a sample ticket has been processed. To conveniently sample the state of the simulation at fixed intervals, a sample ticket in combination with a simulation listener is recommended. See section B.1.2 for more information on sample tickets.

In order to react to the SimulationEvents fired during a simulation, a class implementing the SimulationEventListener should be registered via the addSimulationEventListener method of the Simulator class. To fire an event from, for example, an additional state updater, a call to the fireSimulationEvent method of the Simulator class should be made. Whenever such as call is made, the Simulator calls all registered SimulationEventListeners in order of their registration.

B.1.8. Timer Listener

Timer events are a special type of events that are triggered periodically in simulation time, not simulated time. This means that a timer event that is set to be triggered every 5 seconds is triggered every 5 seconds of running time. The timer interval (measured in milliseconds) can be set by changing the timerInterval in the Simulator class.

A TimerEvent contains both the simulation time and the simulated time, and offers access to the Simulator object. They can be reacted to by registering a class that implements the TimerEventListener interface via the addTimerEventListener method of the Simulator class.

B.1.9. Visualisation

As a part of the GUI mode of the simulator it is possible to visualise the network state.

B.2. USING THE SIMULATOR

The simulator can be used both as a part of existing code or as a stand alone simulator. In stand-alone mode, the simulator can be used either from a command line or by using a graphical user interface. The following subsections give short examples of these use cases.

B.2.1. RUNNING SIMULATIONS

The following example shows how to run a simulation of the classical SIS process on a graph that is given as a simple adjacency list.

```
import siss.core.ExponentialRV;
import siss.core.SIS_ASU;
import siss.core.SimpleInitialiser;
import siss.core.Simulator;
import siss.listeners.SimpleTimerListener;
import siss.tickethandlers.SISCureTicketHandler;
import siss.tickethandlers.SISSpreadTicketHandler;

public class example {

public void simulate()
{
        // create simulator with 1 level for network in file network.txt
            (adjacency list)
        Simulator S = new Simulator("network.txt",1,2);
        // Infection is Poisson process with rate 0.125
        S.setSpreadDist(new ExponentialRV(0.125));
        // Curing is Poisson process with rate 1
        S.setCureDist(new ExponentialRV(1));
        // Initial state has 1 infected node
        S.NI = new SimpleInitialiser(1);
        // Simulation will run for 1000 time units
        S.timeLimit = 1e3;
        S.stoppingRules = new int[]{Simulator.TIME_LIMITED};
        // Additional State Updater fires network change events
        S.setASU(new SIS_ASU(true,false));
        // Regular SIS ticket handlers
        S.setSTH(new SISSpreadTicketHandler());
        S.setCTH(new SISCureTicketHandler());
        // Add a TimerEventListener to output average number of infected
            nodes
        S.addTimerEventListener(new SimpleTimerListener(true,false));
        // Run simulation
        S.run();
}}
```

Stopping Rules

The stopping rule specified in the code above is used to stop terminate the simulation. Two stopping rules are defined:

- TIME_LIMITED as the name suggests, the simulation stops after a specified simulated time. The simulated time limit is set by the timeLimit variable of the Simulator class.
- EVENT_LIMITED this stopping rule monitors the number of times the network state has changed. An event in this setting is a change in the network state, not a processed ticket.

If multiple stopping rules are requested, the simulation only terminates when all stopping rules have been met.

GUI

Instead of writing some start code or embedding the simulator in existing code, a GUI is provided to run simulations. The GUI consists of a dialog screen that acts as a read-only console for both standard messages and error messages and various input fields and tabs to set all the needed parameters. On different tabs, it is possible to specify which implementations of the different interfaces (such as the ticket handlers, network initialiser, additional state updater, simulation listeners and timer listeners) should be used. The class-path is scanned automatically and all implementing classes of the various interfaces found on the class-path can be selected in the GUI.

The current settings can be saved to a configuration file using via a menu item. This configuration file can also be used to start the simulator via de command prompt without using the GUI.

Command Line

As mentioned in Section B.2.1, all required settings for a simulation can be saved into a configuration file which can be used to start a simulation from the command prompt. In addition to passing the location of the configuration file, it is possible to overwrite certain parameters from the command line using the -overwrite argument. This feature is especially handy when multiple nearly identical simulations have to be run on a computation grid to perform, for example, parallel parameter sweeps.

SUMMARY

Local interactions on a graph will lead to global dynamic behaviour. In this thesis we focus on two types of dynamic processes on graphs: the Susceptible-Infected-Susceptilbe (SIS) virus spreading model, and gossip style epidemic algorithms. The largest part of this thesis is devoted to the SIS model. We first introduce the SIS model in chapter 2. Even though the SIS model is a Markov process, and good mathematical tools exist to analyse Markov processes, the exploding state space of the SIS model make many of the standard approaches infeasible. Mean-field approximations and simulations are usually our best bet when trying to solve for the average fraction of infected nodes during an outbreak. We compare two of the mean-field approximations commonly used in literature, and benchmark them against a modified version of the SIS model, the ε-SIS model. We show that especially around the epidemic threshold, which separates a regime where the virus dies out quickly from a regime where it stays in the network for a very long time (called the meta-stable state), the two approximations can deviate from the true value. In general, the NIMFA approximation has the upper hand in terms of accuracy.

Although the time behaviour of the SIS model is used loosely in the description of the meta-stable state and the epidemic threshold, it has received less attention in research than the average fraction of infected nodes in meta-stable state. In chapter 3 we show that the survival time of an SIS process is exponentially distributed, and derive exact expressions for the average survival time in the complete and the star graph. We also propose a definition of the epidemic threshold using the survival time, and show in the star graph and the complete graph that it is accurate.

In general, viruses do not live in isolation. In chapter 4 we show that two viruses competing for the same healthy nodes in a network form a rich dynamic system. If the viruses are prevented from dying out, one will completely dominate the other in terms of the number of infected nodes, but this domination will not last. Even if the dominated virus is down to only a few infected nodes, it will regain strength and eventually dominate the other virus. We compute the average domination time and show that it depends on the number of infected nodes at the start of a domination period. Also, we show that weaker or slower viruses can still dominate their stronger or quicker competitors, but for shorter periods of time.

Real world spreading phenomena generally have infection and curing processes that are not well modelled by Poisson processes. In the context of content propagation, for example, it has been shown that human (online) behaviour is bursty: long periods of inactivity are followed by short periods of intense activity. In chapter 5, we analyse the effect of non exponential inter-arrival time distributions on the behaviour of an SIS process using a Weibull distribution. As the exponential distribution is a special case of the Weibull distribution, we can gradually steer the distribution away from exponential. The effects on the average fraction of infected nodes in the meta-stable state for the same average inter-arrival time is dramatic. However, by keeping the expected number of in-

fection attempts during an infectious period constant, the NIMFA equations still hold in the non-Markovian setting. The fact that spreading attempts are no longer uniformly distributed over the infectious period of a node leads to shorter or longer survival times and can lead to a higher or lower fraction of infected nodes. When infection attempts tend to be concentrated at either the beginning or the end of the infectious period, the survival time is shorter. Also, the accuracy of the NIMFA approximation depends on the inter-arrival time distribution.

The class of epidemic or gossip algorithms mimic the spreading of an infection in its communication process. In chapter 6 we develop COUNT, an epidemic algorithm that can count, average and find minima and maxima in a large distributed dynamic network. COUNT can be sped up dramatically by using BEACON, an algorithm where nodes compete amongst themselves to become the beacon node in a network towards which the message in COUNT can be forwarded. The two algorithms together form GOSSIPCO, a much faster version of COUNT that also includes convergence detection. GOSSIPICO is robust against network dynamics by the virtue of an automatic recount process that each node can trigger in response to changes in the network. The algorithm converges in logarithmic time in random graphs and scale-free graphs, and in a time proportional to the average hopcount in graphs such as lattices and paths.

Finally, in chapter 7 we develop a framework to extract a network from a dataset based on a mapping. The interactions between entities that are recorded in the dataset, in our case a record of millions of instances of played matches in the online game Defence of the Ancients (DOTA), are mapped to links in the graph. We show that different mappings lead to different graphs, highlighting different implicit social interactions amongst players. We show that thresholding has a large effect on which social structures are revealed by the mappings, and can be used in these graphs as a crude but fast community detection scheme. Also in chapter 7, we investigate how to adapt networks to make them less vulnerable to viruses. The underlying idea is that by reducing the largest eigenvalue of the adjacency matrix, which is an approximation of the epidemic threshold, the real epidemic threshold is increased. As the minimisation problem is an NP-hard problem, we compare various strategies in a greedy optimisation approach.

In appendix A we describe most of the software that we developed during the course of our research, while in appendix B we provide an overview of the SIS simulator we developed to perform all the simulations in the first five chapters.

SAMENVATTING

Lokale interacties in een graaf leiden tot globaal dynamisch gedrag. In dit proefschrift richten we ons op twee type dynamische processen in grafen: het Susceptible-Infected-Susceptible (SIS) virusverspreidingsmodel , en gossip-achtige epidemische algoritmes. Het grootste gedeelte van dit proefschrift is geweid aan het SIS model. We introduceren het SIS model in hoofdstuk 2. Alhoewel het SIS model een Markov proces is, en er goede wiskundige methodes bestaan om Markov processen te analyseren, de exploderende toestandsruimte van het SIS proces maken de standaard methodes onpraktisch. Gemiddeldveld benaderingen en simulaties zijn vaak de beste weg om het gemiddeld aantal geïnfecteerde knopen in een uitbraak te bepalen. We vergelijken twee gemiddeldveld benaderingen uit de literatuur, en bepalen hoe goed ze zijn door ze te meten aan een aangepaste versie van het SIS model, het e-SIS model. We tonen aan dat vooral rond de epidemische drempelwaarde, die een regime waar een virus snel uitsterft scheidt van een regime waar het virus voor zeer lange tijd in het netwerk blijft (de meta-stable state), de twee benaderingen ver van de werkelijke waarde af kunnen liggen. In het algemeen is de NIMFA benadering beter qua nauwkeurigheid.

Hoewel het tijdsgedrag van het SIS model losjes gebruikt wordt in de beschrijving van de meta-stable state en de epidemisch drempelwaarde, krijgt het minder aandacht in de literatuur dan het gemiddeld aantal geïnfecteerde knopen in de meta-stable state. In hoofdstuk 3 tonen we aan dat de overlevingstijd van een SIS proces exponentieel verdeeld is, en leiden we exacte uitdrukkingen af voor de gemiddelde overlevingstijd in de complete graaf en de ster graaf. Ook stellen we een definitie van de epidemische drempel voor die berust op de overlevingstijd, en laten in de complete graaf en de ster graaf zien dat die definitie precies is.

Over het algemeen komen virussen niet in isolatie voor. In hoofdstuk 4 laten we zien dat twee virussen die strijden om dezelfde gezonden knopen in een graaf, een rijk dynamisch systeem vormen. Als het onmogelijk gemaakt wordt voor de twee virussen om uit te sterven, zal één van de twee de ander compleet domineren wat betreft het aantal geïnfecteerde knopen, maar deze dominantie zal tot een einde komen. Zelfs al is het gedomineerde virus gereduceerd tot slechts enkele geïnfecteerde knopen, dan nog zal het weer aansterken en uiteindelijk het andere virus domineren. We berekenen de gemiddelde dominante tijd en tonen aan dat die afhangt van het aantal geïnfecteerde knopen aan het begin van de dominante periode. Daarnaast tonen we aan dat zwakkere of tragere virussen nog altijd hun sterkere of snellere competitie kunnen domineren, maar voor kortere tijdsperiodes. De verspreidingsprocessen die in de echte wereld voorkomen hebben over het algemeen infectie en helingsprocessen die niet goed beschreven worden door een Poisson proces. Zo is aangetoond, bijvoorbeeld, dat in informatie verbreiding menselijk (online) gedrag wisselvallig is: lange periodes van rust worden afgewisseld met korte periodes van intense activiteit. In hoofdstuk 5 analyseren we het effect van niet exponentiele tussenaankomsttijdverdelingen op het gedrag van een SIS

proces aan de hand van een Weibull verdeling. Aangezien de exponentiele verdeling een speciaal geval is van de Weibull verdelingen, kunnen we de verdelingen geleidelijk van exponentieel laten afwijken. Het effect op het gemiddeld aantal geïnfecteerde knopen in de meta-stable state voor dezelfde gemiddelde tussenaankomsttijd is groot. Maar, als het gemiddeld aantal infectiepogingen tijdens een geïnfecteerde periode constant gehouden wordt, blijven de NIMFA vergelijkingen geldig in het niet Markoviaanse proces. Dat infectiepogingen nu niet langer uniform verdeeld zijn over een geïnfecteerde periode leidt tot een kortere of langere overlevingstijd, en kan leiden tot een groter of kleiner gemiddeld aantal geïnfecteerde knopen. Wanneer infectiepogingen in het begin of juist aan het eind van de infectieperiode geconcentreerd zijn, is de gemiddelde overlevingstijd korter. Ook is de nauwkeurigheid van de NIMFA benadering afhankelijk van de tussenaankomsttijdverdeling.

De klasse van epidemische of gossip-achtige algoritmes bootsen de verspreiding van een virus na in het communicatieproces. In hoofdstuk 6 ontwikkelen we COUNT, een epidemisch algoritme waarmee het mogelijk is om zowel te tellen, het gemiddelde, en minimum of maximum waardes te bepalen in grote gedistribueerde en dynamische netwerken. COUNT kan aanzienlijke versneld worden door gebruik te maken van BEACON, een algoritme waarin knopen elkaar beconcurreren om een baken te worden waarheen de berichtjes in COUNT gestuurd kunnen worden. De twee algoritmes samen vormen GOSSIPICO, dat een veel snellere versie van COUNT is, en ook convergentie kan detecteren. GOSSIPICO is robuust in een dynamisch netwerk door een automatisch hertel mechanisme dat alle knopen kunnen initiëren in reactie op veranderingen in het netwerk. Het algoritme convergeert in logaritmische tijd in random grafen en scale-free grafen, en in een tijd die proportioneel is aan de gemiddelde afstand in grafen zoals het rooster en pad.

Tenslotte ontwikkelen we in hoofdstuk 7 een methode om een netwerk uit een dataset te onttrekken gebaseerd op een mapping. De relaties tussen entiteiten die in de dataset opgeslagen zijn, in ons geval miljoenen gespeelde potjes in het online spel Defence of the Ancients (DOTA), worden op zijden afgebeeld. We tonen aan dat verschillende mappings tot verschillende grafen leiden die verschillende impliciete sociale interacties tussen spelers weergeven. Het toepassen van een drempelwaarde in het maken van zijden heeft een grote invloed op welke sociale structuren door de mappings zichtbaar gemaakt worden, en kan gebruikt worden om grof maar snel groepen te vinden. In hoofdstuk 7 onderzoeken we ook hoe netwerken veranderd kunnen worden om ze minder vatbaar te maken voor virussen. De achterliggende gedachte is dat door de grootste eigenwaarde van de bogenmatrix, wat een benadering is van de epidemische drempelwaarde, te verkleinen, de werkelijke epidemische drempelwaarde groter wordt. Aangezien het minimaliseringsprobleem een NP-hard probleem is, vergelijken we verschillende greedy optimalisatie technieken. In appendix A beschrijven we het merendeel van de software die we ontwikkeld hebben in de loop van ons onderzoek, en in appendix B geven we een overzicht van de SIS simulator die we ontwikkeld hebben om alle simulaties uit de eerste vijf hoofdstukken te doen.

BIBLIOGRAPHY

[1] E. Jenelius, T. Petersen, and L. Mattsson, *Importance and exposure in road network vulnerability analysis,* Transportation Research Part A: Policy and Practice **40**, 537 (2006).

[2] P. Angeloudis and D. Fisk, *Large subway systems as complex networks,* Physica A: Statistical Mechanics and its Applications **367**, 553 (2006).

[3] P. Kaluza, A. Kölzsch, M. Gastner, and B. Blasius, *The complex network of global cargo ship movements,* Journal of the Royal Society Interface **7**, 1093 (2010).

[4] R. Guimera, S. Mossa, A. Turtschi, and L. Amaral, *The worldwide air transportation network: Anomalous centrality, community structure, and cities' global roles,* Proceedings of the National Academy of Sciences **102**, 7794 (2005).

[5] R. Albert, I. Albert, and G. Nakarado, *Structural vulnerability of the north american power grid,* Physical Review E **69**, 025103 (2004).

[6] H. Wang, E. van Boven, A. Krishnakumar, M. Hosseini, H. van Hooff, T. Takema, N. Baken, and P. Van Mieghem, *Multi-weighted monetary transaction network,* Advances in Complex Systems **14**, 691 (2011).

[7] M. E. J. Newman, *The structure of scientific collaboration networks,* Proceedings of the National Academy of Sciences **98**, 404 (2001).

[8] R. van de Bovenkamp, S. Shen, A. Iosup, and F. A. Kuipers, *Understanding and recommending play relationships in online social gaming,* in *Proc. of the 5th International Conference on COMmunication Systems and NETworkS (COMSNETS 2013)* (2013).

[9] C. J. Stam, A. Hillebrand, H. Wang, and P. Van Mieghem, *Emergence of modular structure in a large-scale brain network with interactions between dynamics and connectivity,* Frontiers in Computational Neuroscience **4** (2010).

[10] G. Oestreicher-Singer and A. Sundararajan, *Recommendation networks and the long tail of electronic commerce,* MIS Quarterly **36**, 65 (2012).

[11] R. Anderson and R. May, *Infectious Diseases of Humans: Dynamics and Control* (Oxford University Press, 1991).

[12] P. Van Mieghem, J. Omic, and R. Kooij, *Virus spread in networks,* IEEE/ACM Transactions on Networking **17**, 1 (2009).

[13] P. Van Mieghem, *Epidemic phase transition of the sis type in networks,* EPL (Europhysics Letters) **97**, 48004 (2012).

[14] R. Pastor-Satorras and A. Vespignani, *Epidemic dynamics and endemic states in complex networks,* Physical Review E **63**, 066117 (2001).

[15] N. T. J. Bailey, *The mathematical theory of infectious diseases and its applications* (Charles Griffin & Company, 2nd edition, 1975).

[16] A. Barrat, M. Barthelemy, and A. Vespignani, *Dynamical processes on complex networks* (Cambridge University Press, Cambridge, U.K., 2008).

[17] C. Castellano and R. Pastor-Satorras, *Thresholds for epidemic spreading in networks,* Physical Review Letters **105**, 218701 (2010).

[18] D. J. Daley and J. Gani, *Epidemic modelling: an introduction* (Cambridge University Press, Cambridge, U.K., 2001).

[19] J. O. Kephart and S. R. White, *Directed-graph epidemiological models of computer viruses,* in *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy* (IEEE, 1991) pp. 343–359.

[20] R. Pastor-Satorras and A. Vespignani, *Epidemic spreading in scale-free networks,* Physical review letters **86**, 3200 (2001).

[21] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, *Epidemic spreading in real networks: an eigenvalue viewpoint,* in *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on* (2003) pp. 25–34.

[22] S. Gómez, A. Arenas, J. Borge-Holthoefer, S. Meloni, and Y. Moreno, *Discrete-time markov chain approach to contact-based disease spreading in complex networks,* EPL (Europhysics Letters) **89**, 38009 (2010).

[23] E. Cator and P. Van Mieghem, *Second-order mean-field susceptible-infected-susceptible epidemic threshold,* Physical Review E **85**, 056111 (2012).

[24] A. L. Hill, D. G. Rand, M. A. Nowak, and N. A. Christakis, *Emotions as infectious diseases in a large social network: the sisa model,* Proceedings of the Royal Society B: Biological Sciences **277**, 3827 (2010).

[25] H. Hinrichsen, *Non-equilibrium critical phenomena and phase transitions into absorbing states,* Advances in physics **49**, 815 (2000).

[26] P. Van Mieghem and E. Cator, *Epidemics in networks with nodal self-infection and the epidemic threshold,* Physical Review E **86**, 016116 (2012).

[27] P. Van Mieghem, *Performance Analysis for Communications Networks and Systems* (Cambridge University Press, 2006).

[28] M. Boguñá and R. Pastor-Satorras, *Epidemic spreading in correlated complex networks,* Physical Review E **66**, 047104 (2002).

[29] P. Van Mieghem, *The N-intertwined SIS epidemic network model,* Computing **93**, 147 (2011).

[30] P. Van Mieghem, *Graph Spectra for Complex Networks* (Cambridge University Press, 2011).

[31] P. Van Mieghem, J. Omic, and R. Kooij, *Virus spread in networks,* IEEE/ACM Transactions on Networking **17**, 1 (2009).

[32] M. M. de Oliveira and R. Dickman, *How to simulate the quasistationary state,* Physical Review E **71**, 016129 (2005).

[33] S. C. Ferreira, R. S. Ferreira, and R. Pastor-Satorras, *Quasistationary analysis of the contact process on annealed scale-free networks,* Physical Review E **83**, 066113.

[34] H. Kesten, *The critical probability of bond percolation on the square lattice equals 1/2,* Communications in mathematical physics **74**, 41 (1980).

[35] R. T. Smythe and J. C. Wierman, *First-passage percolation on the square lattice* (Springer-Verlag Berlin: Springer, 1978).

[36] D. J. A. Welsh, *Percolation and related topics,* Science progress **64**, 65 (1977).

[37] P. Erdős and A. Rényi, *On random graphs,* Publicat. Mathematicae Debrecen **6**, 290 (1959).

[38] A.-L. Barabási and R. Albert, *Emergence of Scaling in Random Networks,* Science **286**, 509 (1999).

[39] D. J. Watts and S. H. Strogatz, *Collective dynamics of small-world networks,* Nature **393**, 440 (1998).

[40] M. E. J. Newman and D. J. Watts, *Renormalization group analysis of the small-world network model,* Physics Letters A **263**, 341 (1999).

[41] C. J. Stam, B. F. Jones, G. Nolte, M. Breakspear, and P. H. Scheltens, *Small-world networks and functional connectivity in alzheimer's disease,* Cerebral Cortex **17**, 92 (2007).

[42] Y. Kuramoto, *Chemical oscillations, waves, and turbulence* (Springer, Berlin, 1984).

[43] C. Li, R. van de Bovenkamp, and P. Van Mieghem, *Susceptible-infected-susceptible model: A comparison of n-intertwined and heterogeneous mean-field approximations,* Physical Review E **86**, 026116 (2012).

[44] M. E. J. Newman, *Mixing patterns in networks,* Physical Review E **67**, 026126 (2003).

[45] P. Van Mieghem, H. Wang, X. Ge, S. Tang, and F. A. Kuipers, *Influence of assortativity and degree-preserving rewiring on the spectra of networks,* The European Physical Journal B **76**, 643 (2010).

[46] C. Li, H. Wang, W. de Haan, C. J. Stam, and P. Van Mieghem, *The correlation of metrics in complex networks with applications in functional brain networks,* Journal of Statistical Mechanics: Theory and Experiment **2011**, P11018 (2011).

[47] P. Van Mieghem, *Decay towards the overall-healthy state in sis epidemics on networks,* arXiv:1310.3980 (2013).

[48] R. Parshani, S. Carmi, and S. Havlin, *Epidemic threshold for the susceptible-infectious-susceptible model on random networks,* Physical Review Letters **104**, 258701 (2010).

[49] E. Cator and P. Van Mieghem, *Susceptible-Infected-Susceptible epidemics on the complete graph and the star graph: Exact analysis,* Physical Review E **87**, 012811 (2013).

[50] J. R. Artalejo, *On the time to extinction from quasi-stationarity: A unified approach,* Physica A: Statistical Mechanics and its Applications **391**, 4483 (2012).

[51] J. A. Fill, *The passage time distribution for a birth-and-death chain: Strong stationary duality gives a first stochastic proof,* Journal of Theoretical Probability **22**, 543 (2009).

[52] L. Miclo, *On absorption times and dirichlet eigenvalues,* ESAIM: Probability and Statistics **14**, 117 (2010).

[53] P. Van Mieghem, *Performance Analysis of Complex Networks and Systems* (Cambridge University Press, 2014).

[54] J. Norris, *Markov Chains*, Cambridge series on statistical and probabilistic mathematics (Cambridge University Press, 2005).

[55] E. Cator and P. Van Mieghem, *Nodal infection in markovian susceptible-infected-susceptible and susceptible-infected-removed epidemics on networks are non-negatively correlated,* Physical Review E **89**, 052802 (2014).

[56] M. Boguñá, C. Castellano, and R. Pastor-Satorras, *Nature of the epidemic threshold for the susceptible-infected-susceptible dynamics in networks,* Physical Review Letters **111**, 068701 (2013).

[57] T. Mountford, J.-C. Mourrat, D. Valesin, and Q. Yao, *Exponential extinction time of the contact process on finite graphs,* arXiv preprint arXiv:1203.2972 (2012).

[58] M. Draief and L. Massouli, *Epidemics and rumours in complex networks* (Cambridge University Press, 2010).

[59] C. Doerr, N. Blenn, and P. Van Mieghem, *Lognormal infection times of online information spread,* PLoS ONE **8**, 1 (2013).

[60] J. Leskovec, L. A. Adamic, and B. A. Huberman, *The dynamics of viral marketing,* ACM Trans. Web **1** (2007).

[61] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, *Information diffusion through blogspace,* in *Proceedings of the 13th international conference on World Wide Web*, WWW '04 (ACM, New York, NY, USA, 2004) pp. 491–501.

[62] J. Leskovec, M. Mcglohon, C. Faloutsos, N. Glance, and M. Hurst, *Patterns of cascading behavior in large blog graphs,* in *Proceedings of the 2007 SIAM International Conference on Data Mining* (2007) pp. 551–556.

[63] B. A. Prakash, A. Beutel, R. Rosenfeld, and C. Faloutsos, *Winner takes all: competing viruses or ideas on fair-play networks,* in *Proceedings of the 21st international conference on World Wide Web*, WWW '12 (ACM, New York, NY, USA, 2012) pp. 1037–1046.

[64] C. Castillo-Chavez, W. Huang, and J. Li, *Competitive exclusion and coexistence of multiple strains in an SIS STD model,* SIAM Journal on Applied Mathematics **59**, 1790 (1999), http://epubs.siam.org/doi/pdf/10.1137/S0036139997325862 .

[65] A. Ackleh and L. Allen, *Competitive exclusion in SIS and SIR epidemic models with total cross immunity and density-dependent host mortality,* Discrete and Continuous Dynamical Systems Series B **5**, 175 (2005).

[66] A. Beutel, B. A. Prakash, R. Rosenfeld, and C. Faloutsos, *Interacting viruses in networks: can both survive?* in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12 (ACM, New York, NY, USA, 2012) pp. 426–434.

[67] B. Karrer and M. E. J. Newman, *Competing epidemics on complex networks,* Physical Review E **84**, 036106 (2011).

[68] C. Poletto, S. Meloni, V. Colizza, Y. Moreno, and A. Vespignani, *Host mobility drives pathogen competition in spatially structured populations,* PLoS Comput Biol **9**, e1003169 (2013).

[69] X. Wei, N. Valler, B. Prakash, I. Neamtiu, M. Faloutsos, and C. Faloutsos, *Competing memes propagation on networks: A network science perspective,* Selected Areas in Communications, IEEE Journal on **31**, 1049 (2013).

[70] F. Shahneh and C. Scoglio, *May the best meme win!: New exploration of competitive epidemic spreading over arbitrary multi-layer networks,* preprint arXiv:1308.4880v2 .

[71] V. Marceau, P.-A. Noël, L. Hébert-Dufresne, A. Allard, and L. J. Dubé, *Modeling the dynamical interaction between epidemics on overlay networks,* Physical Review E **84**, 026105 (2011).

[72] S. A. Myers and J. Leskovec, *Clash of the contagions: Cooperation and competition in information diffusion,* Data Mining , 539 (2012).

[73] F. Gebali, *Analysis of Computer and Communication Networks* (Springer US, 2008).

[74] F. D. Sahneh, C. Scoglio, and P. Van Mieghem, *Generalized epidemic mean-field model for spreading processes over multi-layer complex networks,* IEEE/ACM Transactions on Networking (2013).

[75] C. Li, R. van de Bovenkamp, and P. Van Mieghem, *Susceptible-infected-susceptible model: A comparison of N-intertwined and heterogeneous mean-field approximations,* Physical Review E **86** (2012).

[76] A.-L. Barabasi, *The origin of bursts and heavy tails in human dynamics,* Nature **435**, 207 (2005).

[77] J.-P. Eckmann, E. Moses, and D. Sergi, *Entropy of dialogues creates coherent structures in e-mail traffic,* Proceedings of the National Academy of Sciences of the United States of America **101**, 14333 (2004).

[78] B. Gonçalves and J. J. Ramasco, *Human dynamics revealed through web analytics,* Physical Review E **78**, 026123 (2008).

[79] F. Radicchi, *Human activity in the web,* Physical Review E **80**, 026118 (2009).

[80] J. Leskovec and E. Horvitz, *Planetary-scale views on a large instant-messaging network,* in *Proceedings of the 17th international conference on World Wide Web* (ACM, 2008) pp. 915–924.

[81] J. Candia, M. C. González, P. Wang, T. Schoenharl, G. Madey, and A.-L. Barabási, *Uncovering individual and collective human dynamics from mobile phone records,* Journal of Physics A: Mathematical and Theoretical **41**, 224015 (2008).

[82] E. Limpert, W. A. Stahel, and M. Abbt, *Log-normal distributions across the sciences: keys and clues.* Bioscience **51**, 341 (2001).

[83] C. Castellano and R. Pastor-Satorras, *Competing activation mechanisms in epidemics on networks.* Nature Scientific Reports **2** (2012).

[84] O. Givan, N. Schwartz, A. Cygelberg, and L. Stone, *Predicting epidemic thresholds on complex networks: Limitations of mean-field approaches.* Journal of Theoretical Biology **288**, 21 (2011).

[85] P. L. Simon, M. Taylor, and I. Kiss, *Exact epidemic models on graphs using graph-automorphism driven lumping,* Journal of Mathematical Biology **62**, 479 (2011).

[86] T. E. Harris, *The Theory of Branching Processes* (Springer-Verlag, 1963).

[87] A. N. Startsev, *On the distribution of the size of an epidemic in a non-Markovian model.* Theory of Probability Applications **41**, 730 (1997).

[88] H. Hinrichsen, *Non-equilibrium phase transtions with long-range interactions.* Journal of Statistical Mechanics: Theory and Experiment , P07006 (2007).

[89] G. Streftaris and G. J. Gibson, *Non-exponential tolerance to infection in epidemic systems: modeling, inference, and assessment.* Biostatistics **13**, 580 (2012).

[90] F. Chung, L. Lu, and V. Vu, *Spectra of random graphs with given expected degrees.* Proceedings of the National Academy of Sciences of the United States of America **100**, 6313 (2003).

[91] M. Krivelevich and B. Sudakov, *The largest eigenvalue of sparse random graphs.* Combinatorics, Probability and Computing **12**, 61 (2003).

[92] E. Cator, R. van de Bovenkamp, and P. Van Mieghem, *Susceptible-Infected-Susceptible epidemics on networks with general infection and cure times,* Physical Review E. **87**, 062816 (2013).

[93] P. Van Mieghem and R. van de Bovenkamp, *Non-Markovian infection spread dramatically alters the susceptible-infected-susceptible epidemic threshold in networks,* Physical Review Letters **110**, 108701 (2013).

[94] G. Yang, *Empirical study of a non-Markovian epidemic model,* Mathematical Biosciences **14**, 65 (1972).

[95] A. Vazquez, B. Rácz, A. Lukács, and A.-L. Barabási, *Impact of non-Poissonian activity patterns on spreading processes.* Physical Review Letters **98**, 158702 (2007).

[96] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, *Epidemic algorithms for replicated database maintenance,* in *Proc. of the 6th annual ACM Symp. on Principles of Distributed Computing* (1987) pp. 1–12.

[97] E. Le Merrer, A.-M. Kermarrec, and L. Massoulie, *Peer to peer size estimation in large and dynamic networks: A comparative study,* in *Proc. of the Int. High Performance Distributed Computing Symp.* (2006) pp. 7–17.

[98] D. Kostoulas, D. Psaltoulis, I. Gupta, K. Birman, and A. Demers, *Decentralized schemes for size estimation in large and dynamic groups,* in *Proc. of the Int. Network Computing and Applications Symp.* (2005) pp. 41–48.

[99] L. Massoulié, E. Le Merrer, A.-M. Kermarrec, and A. Ganesh, *Peer counting and sampling in overlay networks: random walk methods,* in *Proc. of the 25th annual ACM Symp. on Principles of Distributed Computing* (2006) pp. 123–132.

[100] A. Montresor and A. Ghodsi, *Towards robust peer counting,* in *Proc. of the 9th IEEE Int. Conf. on Peer-to-Peer Computing* (2009) pp. 143–146.

[101] M. Jelasity and A. Montresor, *Epidemic-style proactive aggregation in large overlay networks,* in *Proc. of the 24th Int. Distributed Computing Systems Conf.* (2004) pp. 102–109.

[102] D. Kempe, A. Dobra, and J. Gehrke, *Gossip-based computation of aggregate information,* in *Proc. of the 44th IEEE Symp. on Foundations of Computer Science* (2003) pp. 482–491.

[103] A. Guerrieri, I. Carreras, F. De Pellegrini, D. Miorandi, and A. Montresor, *Distributed estimation of global parameters in delay-tolerant networks,* Computer Communications **33**, 1472 (2010).

[104] G. Giakkoupis, *Tight bounds for rumor spreading in graphs of a given conductance.* in *Proc. of the 28th Symp. on Theoretical Aspects of Computer Science* (2011).

[105] V. Batagelj and U. Brandes, *Efficient generation of large random networks,* Physical Review E **71**, 036113 (2005).

[106] M. Ripeanu, I. T. Foster, and A. Iamnitchi, *Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design,* CoRR (2002).

[107] P. Van Mieghem, G. Hooghiemstra, and R. van der Hofstad, *A Scaling Law for the Hopcount in Internet*, Tech. Rep. (Delft University of Technology, 2000).

[108] Y. Guo, S. Shen, O. Visser, and A. Iosup, *An analysis of online match-based games,* in *Proceedings of the International Workshop on Massively Multiusers Virtual Environment (MMVE)* (2012).

[109] B. Kirman and S. Lawson, *Hardcore classification: Identifying play styles in social games using network analysis,* in *ICEC* (2009) pp. 246–251.

[110] K. J. Shim, K.-W. Hsu, and J. Srivastava, *Modeling player performance in massively multiplayer online role-playing games: The effects of diversity in mentoring network,* in *ASONAM* (2011) pp. 438–442.

[111] P. O. Vaz de Melo, V. A. Almeida, and A. A. Loureiro, *Can complex network metrics predict the behavior of nba teams?* in *Proc. of the Int. Conf. on Knowledge Discovery and Data Mining (KDD)* (2008) pp. 695–703.

[112] M. Szell and S. Thurner, *Measuring social dynamics in a massive multiplayer online game,* Social Networks **32**, 313 (2010).

[113] C. Ang, *Interaction networks and patterns of guild community in massively multiplayer online games,* Social Network Analysis and Mining , 1 (2011).

[114] J. McGonigal, *Reality is Broken: Why Games Make us Better and How They Can Change the World* (Jonathan Cape, 2011).

[115] N. Ducheneaut, N. Yee, E. Nickell, and R. J. Moore, *The life and death of online gaming communities: a look at guilds in world of warcraft,* in *CHI* (2007) pp. 839–848.

[116] A. Nazir, S. Raza, and C.-N. Chuah, *Unveiling facebook: a measurement study of social network based applications,* in *IMC* (2008) pp. 43–56.

[117] J. Rossignol, *This Gaming Life: Travels in Three Cities* (U. Michigan Press, 2008).

[118] Y. Guo and A. Iosup, *The game trace archive,* in *ACM Annual Workshop on Network and Systems Support for Games (NETGAMES)* (2012) pp. 1–6.

[119] J. Leskovec, L. Backstrom, and J. M. Kleinberg, *Meme-tracking and the dynamics of the news cycle,* in *KDD* (ACM, 2009) pp. 497–506.

[120] Costa, F. Rodrigues, G. Travieso, and V. Boas, *Characterization of complex networks: A survey of measurements,* Adv. Phys. **56** (2007), 10.1080/00018730601170527.

[121] M. E. J. Newman, *The Structure and Function of Complex Networks,* SIAM Review **45**, 167 (2003).

[122] M. Kitsak, L. K. Gallos, S. Havlin, F. Liljeros, L. Muchnik, H. E. Stanley, and H. A. Makse, *Identification of influential spreaders in complex networks,* Nature Physics **6**, 888 (2010).

[123] J. Restrepo, E. Ott, and B. Hunt, *Onset of synchronization in large networks of coupled oscillators,* Physical Review E **71**, 036151 (2005).

[124] J. G. Restrepo, E. Ott, and B. R. Hunt, *Characterizing the dynamical importance of network nodes and links,* Physical Review Letters **97**, 094102 (2006).

[125] A. Milanese, J. Sun, and T. Nishikawa, *Approximating spectral impact of structural perturbations in large networks,* Physical Review E **81**, 046112 (2010).

[126] P. Van Mieghem, D. Stevanovic, F. A. Kuipers, C. Li, R. van de Bovenkamp, D. Liu, and H. Wang, *Decreasing the spectral radius of a graph by link removals,* Physical Review E **84**, 016101 (2011).

[127] M. R. Garey and D. S. Johnson, *Computer and intractability,* A Guide to the NP-Completeness. Ney York, NY: WH Freeman and Company (1979).

[128] H. Yuan, *A bound on the spectral radius of graphs,* Linear Algebra and its Applications **108**, 135 (1988).

[129] J. G. Restrepo, E. Ott, and B. R. Hunt, *Approximating the largest eigenvalue of network adjacency matrices,* Physical Review E **76**, 056119 (2007).

[130] Y. Lu, J.D.D. Mol, F.A. Kuipers, and P. Van Mieghem, *Analytical Model for Mesh-based P2PVoD,* in *Proc. of the 10th IEEE International Symposium on Multimedia (IEEE ISM 2008)* (December 15-17, 2008).

[131] R. van de Bovenkamp, *Network analysis toolkit,* .

[132] T. Kleiberg, B. Fu, F.A. Kuipers, S. Avallone, B. Quoitin, and P. Van Mieghem, *DeSiNe: a flow-level QoS simulator of Networks,* in *Proc. of the first International Workshop on the Evaluation of Quality of Service through Simulation in the Future Internet (QoSim)* (Marseille, France, March 3, 2008).

[133] R. Bayer, *Symmetric binary b-trees: Data structure and maintenance algorithms,* Acta informatica **1**, 290 (1972).

[134] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, *Randomized gossip algorithms,* IEEE Transactions on Information Theory **52**, 2508 (2006).

[135] A. Dimakis, S. Kar, J. Moura, M. Rabbat, and A. Scaglione, *Gossip algorithms for distributed signal processing,* Proceedings of the IEEE **98**, 1847 (2010).

[136] R. van de Bovenkamp, F. A. Kuipers, and P. Van Mieghem, *Gossip-based counting in dynamic networks,* in *Proc. of IFIP Networking 2012* (2012).

[137] U. Brandes, *On variants of shortest-path betweenness centrality and their generic computation,* SOCIAL NETWORKS **30** (2008).

[138] V. Batagelj and M. Zaversnik, *An o(m) algorithm for cores decomposition of networks,* CoRR **cs.DS/0310049** (2003).

ACKNOWLEDGMENTS

Four years is a short time to do research in. And although it is one of the things that Piet tells his students whenever he gets the chance, it is only when you are finishing your Ph.D. thesis that the full truth of the statement becomes clear. Now that I have come to the end of my four years of research, I know that at least part of the reason that time goes by so quickly is because doing research in the NAS group can be a lot of fun. Not in the least part because of the nice bunch of diverse people that occupy the 13th floor of EWI. I would like to thank them all for the fun we have had and the wisdom they have shared with me.

I want to start by thanking my promotor Piet Van Mieghem. Four years ago, Piet challenged me to, instead of accepting a job in industry, "devote my time to science and test my brain against the world". His passion for research has been a constant inspiration to me, and his incredible output, although a bit intimidating, a great motivation. I am glad I took up his challenge.

Also, a special thanks to Fernando Kuipers, with his sharp eye for detail. I have always envied the speed with which Fernando can proof read papers, and have often wished he would take a day or two more, instead of an hour, to give his feedback.

Now to my office mates. When I first arrived at the last office down the hall, I was greeted by the three best officemates you could ask for. Anteneh, full of advice and always calm. I don't think I have ever seen Anteneh stressed out, and can only picture him with his broad smile. Cong had started only two months earlier than me, and together we learned the ropes of doing a Ph.D. We went to our first conference together, and managed to attend three more together. I am proud to have been one of your paranymphs. And then there was Xavi. Xavi has a special place in my TU Delft memories. Together we have invented numerous office games. But, in addition to arguing about the robustness of complex networks and having fun, we also explored the darker thoughts that come with doing a Ph.D. I could always discuss my doubts and questions about why we do research with him. Even though he never disagreed with me, he always knew how to put things in a brighter perspective. I am proud to have been one of your paranymph as well. Of course officemates come and go. When Anteneh left, Dongchao arrived, and after Xavi came Marcus. Marcus has done a great job replacing Xavi. We share many interests, ranging from poetry to space flight. Xiangrong arrived in our office, and even Cong's place was taken by Hale. You have all made my days in the office enjoyable. Thanks!

I should also thank the "Cluster Support Team". First Wynand, Norbert and Christian taught me how to use the computation cluster, and then how to fix it when it breaks down. You guys taught me a lot of useful skills. After Niels joined I learned even more tricks. Norbert has been a great help and friend throughout my years in NAS. I have enjoyed our discussions when our research topics overlapped. But most of all, there was always time for a beer.

There have been so many people that have helped me in every way, shape or form to write about here. Stojan, thanks for your mathematical insights into my “trivial” problems. And Jil, thank you for all the enjoyable chats, drinks and gossip! Niels, thanks for the morning coffees. And a big thanks to all the rest too: Dajie, Ebisa, António, Edgar, Nico, Rob, Song, Farabi, Rogier, Bo, Annalisa, Evangelos, Martijn, Yakup, Wendy, Remco, Negar, and Shruti.

Finally, I want to thank my parents for their love and support, and for providing me with all the opportunities to study and get where I am today. And my brothers and sisters, Henk, Marja, Bart, Bart, and Sylvia for always being there. Lastly, Natasha, thank you for your support and encouragement over the past four years. I could not have done it without the comfort of knowing that at the end of each day I got to see you again. You have no idea how much you have put into this thesis.

Ruud van de Bovenkamp
Amersfoort, December 2014

CURRICULUM VITÆ

Ruud VAN DE BOVENKAMP

26-11-1981	Born in Heemskerk, The Netherlands.

EDUCATION

1994–2001	Secondary school, Gymnasium Felisenum, Velsen-Zuid
2002–2003	Propaedeutic English Language and Culture, Leiden University
2002–2006	B. A. English Language and Culture, Leiden University
2005–2006	Propaedeutic Electrical Engineering, Delft University of Technology
2005–2008	B. Sc. Electrical Engineering, Delft University of Technology
2006–2010	M. A. English Language and Culture, Leiden University
2008–2010	M. Sc. Electrical Engineering, Delft University of Technology
2010–2014	Ph. D. research at Delft University of Technology Network Architectures and Services group
Thesis:	Epidemic Processes on Complex Networks: Modelling, Simulation and Algorithms
Promotor:	Prof. dr. ir. P. F. A. Van Mieghem

LIST OF PUBLICATIONS

15. **P. Van Mieghem and R. van de Bovenkamp**, *Accuracy criterion for the mean-field approximation in SIS epidemics on networks*, under submission.

14. **M. Märtens, R. van de Bovenkamp and P. Van Mieghem**, *Superinfection in networks*, under submission.

13. **R. van de Bovenkamp and P. Van Mieghem**, *Survival time of the SIS infection process on a graph*, under submission.

12. **A.L. Jia, S. Shen, R. van de Bovenkamp, A. Iosup, F. Kuipers and D. H. J. Epema**, *Socializing by Gaming: Revealing Social Relationships in Multiplayer Online Games*, under submission.

11. **R. van de Bovenkamp and P. Van Mieghem**, *Time to meta-stable state in SIS epidemics on graphs*, Third International IEEE Workshop on Complex Networks and their Applications, November 23-27 2014, Marrakesh, Morocco.

10. **R. van de Bovenkamp, F. Kuipers and P. Van Mieghem**, *Domination-time dynamics in Susceptible-Infected-Susceptible virus competition on networks*, Physical Review E, vol. 89, no. 4, p. 042818, 2014.

9. **A. Iosup, R. van de Bovenkamp, S. Shen, A.L. Jia, and F.A. Kuipers**, *An Analysis of Implicit Social Networks in Multiplayer Online Games*, IEEE Internet Computing, special issue May/June 2014.

8. **R. van de Bovenkamp and F.A. Kuipers**, *A Toolkit for Real-time Analysis of Dynamic Large-scale Networks*, Proceedings of the 20th Annual Symposium on Communications and Vehicular Technology (IEEE SCVT 2013), Namen, Belgium, November 2013.

7. **Guo, D., S. Trajanovski, R. van de Bovenkamp, H. Wang and P. Van Mieghem**, *Epidemic threshold and topological structure of Susceptible-Infected-Susceptible epidemics in adaptive networks*, Physical Review E, vol. 88, no. 4, p. 042802, 2013.

6. **Cator, E. , R. van de Bovenkamp and P. Van Mieghem**, *Susceptible-Infected-Susceptible Epidemics on Networks with General Infection and Curing Times*, Physical Review E, vol.87, no. 6, p. 062816, 2013.

5. **P. Van Mieghem and R. van de Bovenkamp**, *Non-Markovian infection spread dramatically alters the Susceptible-Infected-Susceptible epidemic threshold in networks*, Physical Review Letters, vol. 110, no. 10, p. 108701, 2013.

4. **R. van de Bovenkamp, S. Shen, A. Iosup, and F.A. Kuipers**, *Understanding and Recommending Play Relationships in Online Social Gaming*, proceedings of the 5th international conference on communication systems and networks (COMSNETS 2013), Bangalore, India, January 7-10, 2013.

3. **Li, C., R. van de Bovenkamp and P. Van Mieghem**, *Susceptible-Infected-Susceptible model: A comparison of N-intertwined and heterogeneous mean-field approximations*, Physical Review E, vol.86, no. 2, p. 026116, 2012.

2. **R. van de Bovenkamp, F.A. Kuipers, and P. Van Mieghem**, *Gossip-based counting in dynamic networks*, IFIP Networking, Prague, Czech Republic,May 21-25, 2012.

1. **P. Van Mieghem, D. Stevanovic, F.A. Kuipers, C. Li, R. van de Bovenkamp, D. Liu and H. Wang**, *Decreasing the spectral radius of a graph by link removals*, Physical Review E, vol.84, no. 1, p. 016101, 2011.

RELATION TO THIS THESIS

The content of this thesis is published in most of the afore mentioned papers. The following table indicates in which chapters material from which publications is used.

Publication	Ch. 2	Ch. 3	Ch. 4	Ch. 5	Ch. 6	Ch. 7	App. A
1						x	
2					x		
3	x						
4						x	
5				x			
6				x			
8							x
9						x	
10			x				
11		x					
12						x	
13		x		x			

Table B.1: Relation between publications and chapters in this thesis. The contents of publications 7, 14, and 15 do not feature in this thesis.