Epidemic Spreading in Technological Networks

Jasmin Leveille
Information Infrastructure Laboratory
HP Laboratories Bristol
HPL-2002-287
October 23rd , 2002*

computer security, viruses, worms, infection models, scale-free networks, small world networks, epidemiology, BICAS

Recent computer worms pose a major threat to large computer networks, and it is a general belief that understanding their means of propagation will help to devise efficient control strategies. This dissertation proposes a new epidemiological model to account for particular characteristics of computer worm epidemics. This new model, termed the Progressive Susceptible-Infected-Detected-Removed (PSIDR) epidemiological model, incorporates new aspects related to the availability of antivirus signatures, to the existence of direct immunization, and to the presence of a curing phase. Various costs are incorporated in the model, which allow us to determine the best strategies to fight worms. The model undergoes an extensive series of validation tests, its properties being evaluated mostly numerically. The model shows good agreement with empirical data. The paper then investigates current response strategies as well as the effect of virus throttling. The model yields both practical recommendations and new insights about the observed low prevalence of worms over the Internet.

* Internal Accession Date Only
Approved for External Publication

Epidemic Spreading in Technological Networks

Jasmin Leveille*

September 3, 2002

* This thesis was submitted as partial fulfilment of the requirements of the Master of Science degree in Evolutionary and Adaptive Systems from the School of Cognitive and Computing Sciences within the University of Sussex at Brighton, U.K.

ACKNOWLEDGEMENTS

I would like to thank my advisor Matthew Williamson, without whom this would not have been possible, for his continuous support and advices, and in general for bringing me to a better understanding of science. I am also indebted to him for pushing my Matlab programming skills to a level close to *perfection*.

I am also thankful to Dave Cliff and Inman Harvey for their very enlightening suggestions throughout the project.

Special thanks are due to John Brawn and Mike Wonham from HP Labs. Without their joyful help and extensive knowledge of computer networks and security, this work could not have existed.

I am also grateful to Andrew Byde for helping me with the maths, and Mark Butler for useful advice and also for the Matlab book.

More special thoughts to my family, who helped me from afar to go through this difficult year.

Finally, I wish to thank Brian Truong for his thorough knowledge of computer networks, and for the good times we've had in Brighton, Bristol and Penzance.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Lately, computer worms have become a major problem for large computer networks, causing considerable amounts of resources and time to be spent recovering from large-scale attacks [31]. It is believed that understanding the factors influencing worm propagation in technological networks (such as the Internet, the World Wide Web, phone networks, IP networks, etc.) will suggest useful ways to control them. So far, a few studies have employed simple epidemiological models to understand general characteristics of virus[1] spreading.

Epidemiological models have traditionally been used to understand and predict the outcome of virus outbreaks in human [35] or animal populations [11]. However, the same models were recently applied to the analysis of computer virus epidemics [26]. For example, using a simple model it has been shown that networks that have a topology similar to the Internet are highly vulnerable to viral attacks [44].

This dissertation introduces a new model that accounts for important characteristics of technological outbreaks. Indeed, a new model had to be invented because older models incorporate false assumptions about the basic dynamics of technological epidemics. The new model, based on the typical course of a worm infection, captures several important aspects not previously mentioned:

- *Direct immunization* Whenever a user installs an antivirus software (or updates it) on a machine, this machine is automatically immunized to a certain group of viruses. In previous models, machines could become immune to a virus only by first being infected by it. In the real world, this would imply that users wait to become infected, and then (when its too late), install the antivirus. The new model allows for machines to become immune before they are infected by a virus. Simulations of this model show that the outcome of the epidemic event differs from what had been predicted by older models.

- *Antivirus availability* Most of the times, the antivirus is not available when the epidemic event starts. It is only after a certain period of time that an efficient method to deal with the worm is made available. Simulation results indicate that the duration of this period (called the *response time*) can have drastic consequences on the magnitude of the damage caused by a worm.

[1] In this dissertation, the terms *virus* and *worm* are used interchangeably unless mentioned otherwise. See [13] and [36] for definitions of virus and worm respectively.

- *Curing process* Once it is realised that a particular machine is infected, the machine is disconnected from the network, cleaned up, equipped with a new antivirus and then is reintroduced in the network. Note that the machine is isolated from the network to ensure that it does not contaminate other computers. In a theoretical model, it means that computers do not jump directly from being infected to being cleaned up and good to go: there is a period in the middle where the computer is in the process of being cured. Normally, cleaning up a computer involves manual labour, which consumes significant amount of time and other resources (especially in large corporations). Therefore, the new model includes a transitionary state between that of being *infected* and being *cleaned up*. By calculating how many machines are in this transition state at any instant in time, it is possible to estimate the cost involved in fixing the computer. Previous models, because of their very fundamental structure, cannot estimate this cost.

The new model was tested on a series of experiments in order to answer three kinds of questions.

First, is the model a good account of real epidemics? Very little data exists on the epidemics of computer worms. Nonetheless, simulations of the model agree with existant data. Moreover, the model can account for the generally low prevalence of worms on the Internet, an observation that has puzzled researchers[44].

Second, is the new model significantly better than previous models, and is the increase in complexity necessary? Explicit comparisons with an older model suggests that this is the case. Moreover, the fact that various costs are easily estimated with the model allows it to define what the best antiviral response is in terms of costs. In this respect, simulations indicate that the best strategy to reduce various costs is to keep the response time as low as possible. Note that previous models could only evaluate the efficacy of a curing strategy by looking at the number of infected machines over time.

Finally, the model also yields interesting predictions about new strategies to control computer worms. In this dissertation, the effect of virus throttling (slowing the speed at which the virus can spread to other computers) [53, 54] is shown to have a positive effect on the outcome of an outbreak.

In addition, the basic dynamics of the model were validated by an extensive series of simulations on simpler models. Confirmations of earlier results and predictions suggests that the basic dynamics of the simulations are not corrupted by implementation artifacts.

The main contribution of this study is the introduction of a new—more realistic—epidemiological model meant to target technological outbreaks. Not only is this work important for a proper evaluation of current methods used to fight against worms, but also for the design of new control strategies.

The dissertation starts with a quick review of epidemiological models and computer networks. The new epidemiological model is presented in Chapter 3. Following a series of validation experiments in Chapter 4, the results for various

simulations are then reported in Chapter 5. The dissertation ends with a general discussion about implications of the results.

2. EPIDEMIOLOGICAL MODELS IN NETWORKS

This chapter separately reviews epidemiological models and network topologies. A more detailed review of the various models and results concerning the different topologies is presented in Appendix A.

2.1 Elements of epidemiological models

Epidemic models study the propagation of a virus in a population of individuals (hosts) [35]. One fundamental assumption of all models is that the time-scale of the viral infection is much smaller than the normal lifespan of hosts [34]. This means that the size of the population of hosts is taken to be constant.

In general, epidemic models assume that individuals go through a series of states at a certain constant set of rates. Therefore, the elaboration of a model requires the definition of a set of possible states and of a set of transition rates. The simplest model, referred to as the *SIS* model (for *Susceptible-Infected-Susceptible*) and illustrated in Figure 2.1, is taken here as an example:

- *Set of states of the SIS model.* In this simple model composed of two different states, each individual can be in either the *Susceptible* (S) or the *Infectious* (I) state. Susceptible individuals are simply healthy individuals that can potentially be infected by a virus. Infectious individuals are thoses that have contracted the virus and can now infect the remaining susceptible ones (by direct contact with susceptible individuals, for example). After a variable period of time, *infected* (≡infectious) individuals may naturally recover from the disease, and then go back to the susceptible state. Once they are back in the susceptible state, they can become infected again: thus in the limit, any particular individual will perpetually move between the two states as in $S \rightarrow I \rightarrow S \rightarrow I \rightarrow S \rightarrow I \rightarrow S$....

- *Set of rates of the SIS model.* Two rates of transition are needed to describe the model. The first one (termed the *birth* rate, and symbolised by the letter β) controls transitions from the S to the I state, and the second one (termed the *cure* or *death* rate, and symbolised by the letter δ) regulates transitions from the I to the S state. The rates can be conceived of as probabilities: susceptible individuals become infected by the virus with probability β and infected individuals recover with probability δ. Thus, if β is large relative to δ (say, $\beta = 0.9$ and $\delta = 0.1$), most individuals will tend to be in the infected state.

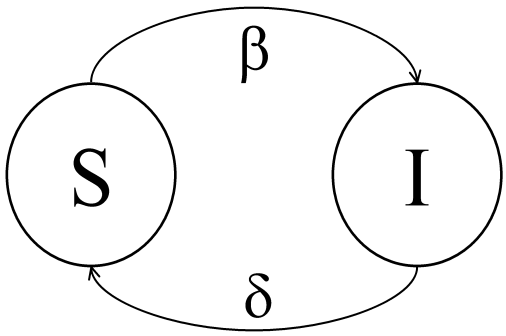

Fig. 2.1: **The SIS model.** Each individual oscillates between the *Susceptible* (S) state and the *Infectious* (I) state. The susceptible individual becomes infectious at a rate β (birth rate) if it is connected to one or more neighbours. The infectious individual becomes susceptible at a rate δ, independent of its neighbours.

Other more complex models include the *Susceptible-Infected-Removed* (SIR) [34] model and the *Susceptible-Exposed-Infected-Removed* (SEIR) model[14]. The former states that individuals cannot go through a perpetual loop as in the SIS model ($S \to I \to S \to I \to S \to I \to S...$). Instead, susceptible individuals that have been infected once and then recover from the virus are considered immune (or simply dead) to further infections. Once the individual is immune, it can no longer transmit the disease to other neighbours. The chain of events is thus of the type $S \to I \to R$. Note that, as in the SIS model, the same fixed rates (*birth* rate β, and *cure*(*death*) rate δ) are assigned to the transitions between states.

The SEIR model is very similar to the SIR model, but it accounts for the fact that some viruses go through a latent period before the host becomes infectious. Typically, a virus will infect a susceptible host (S) before going in the latent period. During the latent period, the host is *infected* but is not *infectious*, a state called *exposed* (E). After some time, the same host becomes infectious (I), and later becomes immune or dead (R). The SEIR model requires the definition of an additional transition rate, meant to regulate transitions between the exposed and the infected states.

2.2 Network topologies

As mentioned above, susceptible individuals become infected at a certain rate β if they are in contact with an infectious individual[44]. This implies that the patterns of contacts between individuals are known.

Patterns of contact are represented with graph models. Graphs are composed

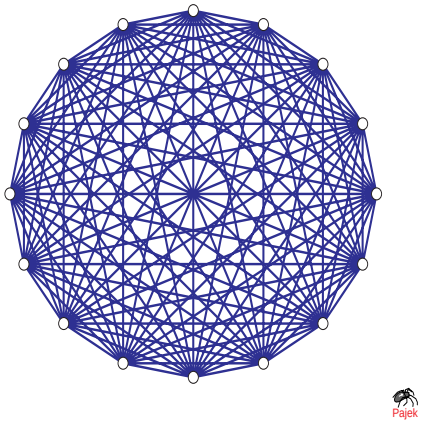

Fig. 2.2: **Fully connected graph with 16 vertices.** Each individual (vertex) can infect each of the other 15 individuals through a direct physical contact (edge). All graph drawings in this thesis were performed using Pajek [5].

of a set of nodes (≡ vertices) connected by a set of edges. Vertices connected to a given node (each through a different edge) are called the *neighbours* of that node. The number of neighbours of a given node is called its *degree*[19]. Two individuals (each represented by a different node), are said to have some sort of direct contact if an edge exists that links the two nodes.

2.2.1 Homogeneous graphs

The simplest possible graph is the *fully-connected* graph: each node is connected to every other node (see Figure 2.2). Fully connected graphs are also called *homogeneous* graphs (HM)[26].

In this dissertation, HM graphs are considered to be good models of some technological network topologies. It has been argued that fully-connected graphs do not offer a realistic account of computer networks [26]. Users tend to communicate with a subset of users, not with everyone in the network. Therefore, the pattern of connections is not really fully-connected. Other graphs have been proposed to model technological networks. The tree network, for example, models technological nets as being an ensemble of communities with many connections intra-community but few connections inter-community. Lattice graphs and random graphs have also been proposed as alternatives [25]. It was true that network topology diverged from the HM graph in the case of ancient viruses, since they could only spread via exchanges between users. However, in recent large corporate networks, where massive mailing lists are stored in each employee's address book, the topology of the network truly resembles a fully-connected graph. For example, many email worms send a copy of themselves

to every email address that they find on an infected host. If one of those email address is in fact a mailing list (as in corporate networks), the worm sends itself to everyone in the mailing list. IP networks are also thought to be homogeneous, since it is possible to reach all IP adresses from any particular IP address [48]. Thus, some technological networks display a homogeneous topology.

2.2.2 Scale free graphs

Recent studies have shown that the topology of some technological networks is represented by a class of graphs called *scale-free networks* (SF). In SFs, most machines have very few connections to other neighbours. However, a few nodes have a very large number of connections [3]. A typical SF is shown in Figure 2.3. For example, the graph of links between web pages in the World Wide Web [4] and the Internet router map are both SF networks [1]. A SF graph can be constructed by starting with a fixed set of nodes, and then adding new nodes one at a time for an arbitrary number of timesteps, a principle called *incremental growth*. Each new node is connected to the existing network by randomly selecting one or more neighbours according to their degree, a principle called *preferential connectivity*. In real networks, these phenomena of *incremental growth* and *preferential connectivity* are also present, which probably explains their scale free structure [3].

SF networks are part of the superclass of *small world* networks [2]. In general, small world networks have a small diameter, which means that very few hops are necessary to travel between any two nodes [39]. The map of human contacts (both social and sexual contacts) is a small world network [52, 1]. That is, some people have many contacts with other people, but some have few contacts. Moreover, it is the case that any two people are related by a relatively small number of intermediate acquaintances[52] (small diameter). Similarly, in email networks (not corporate email networks, but email networks in general), some people have a lot of email addresses in their address book, while some have just a few. Unsurprisingly, the email network has been found to be a SF network[18]. Mobile phone networks may also be SF networks [50]. Therefore, in this dissertation, the scale free topology is taken to be an adequate model of some technological networks topologies. The scale-free structure appears to be pervasive in a wide range of technological (and biological) phenomena [16], probably because most realistic networks turn out to be built according to the principles of incremental growth and preferential connectivity. A method that uses those principles to create scale-free graphs is presented in Appendix B.

2.3 Epidemics and network topologies

Various network topologies exist and each has different properties. It has been shown that the particular topology of a network (i.e. whether it is a SF or a HM network) influences the propagation of a virus. For example in a HM network, using the SIS model of virus propagation, it is possible to show the existence of an epidemic threshold $\lambda = \beta/\delta$ [25]. That is, if the birth rate (β) of a virus

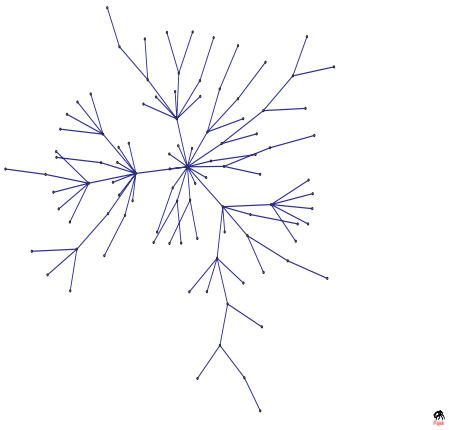

Fig. 2.3: **Scale free graph with 100 vertices.** Graphs such as this are thought to represent many technological networks including the Internet, the WWW and email networks.

is high enough compared to the cure rate (δ), the virus will infect a substantial fraction of nodes in the network. However, if the cure rate is higher than the birth rate, the virus will die out until no one is infected. Formally, if $\beta > \delta$, an outbreak occurs; if $\beta < \delta$, no major outbreak will happen.

In other graphs like the *random network* (ER), the *tree*, or even the *lattice* model, the threshold behaves differently [25]. Recent studies show that SFs don't have an epidemic threshold [44]. Thus, when modelling virus propagation, it is important to consider not only the epidemic model but also the network topology.

2.4 Summary

Epidemiological models are described as a set of states with transitions rates between states. Different models have different predictions about the evolution and outcome of an outbreak.

The topology of a network also plays a role in determining the outcome of an outbreak. Technological networks appear to be best approximated using scale free graphs or homogeneous graphs in some cases.

In the next chapter, a new model is proposed to account for technological epidemics. As for any epidemiological model, it is described as a set of states with transition rates. In the following chapters, analytical predictions and numerical simulations will be conducted on both homogeneous and scale free graphs since they span the whole range of technological networks. This means that the results of the new model will be extensible to a broad variety of technological networks.

3. THE PROGRESSIVE SIDR MODEL

In this chapter, based on ideas gained from previous models, a new epidemiological model is presented that models real processes going on in computer epidemics. The chapter starts by looking at characteristics of a typical outbreak. In the second section, aspects of real outbreaks are included in the PSIDR (*Progressive Susceptible-Infected-Detected-Removed*) model, first in an informal way and then analytically. The definition of the PSIDR model is also accompanied with a discussion about relevant details of its various parameters. A third section mentions aspects not included in the model. The chapter ends with a brief summary.

3.1 Time course of a technological outbreak

Let's imagine the sequence of events that happen when a worm tries to infect a technological network. For simplicity, the network considered here is the email network of a large corporation. One assumption is that all computers in the network have some sort of antivirus software. This software can be updated at a regular rate, say once a day, to make sure that the latest virus signatures are included in the antivirus (AV) software[1].

The first event to happen is the primary infection. For example, an employee opens a file (executable) attached to an email sent from someone from outside the company. This program, once executed, sends itself to some of the employee's contacts (say, the first ten addresses in the address book). Unfortunately, if some of these contacts are in fact mailing lists (lists of contacts), then the worm has the potential of infecting all the contacts listed on it. Once the other users receive the sent emails, some of them may or may not open it immediately, depending on various factors (such as personal habits, etc.).

Before the worm can be cleaned from computers, it has to be detected first. Detection will be particularly difficult when the worm doesn't inflict any direct payload to the machines. Moreover, AV software will only detect worms for which it has the signatures. Therefore, once a few instances of a virus have been noticed, antivirus companies will strive to extract the virus signature and make it available so that all computers can update their AV software. The first

[1] A virus signature is a pattern of instructions, specific to the virus, that is always found in infected files. To know if a file is infected by a virus, the AV scans the file to try to find patterns that correspond to the virus' signature. If the file turns out to be infected, the AV software indicates to the user that there might be an infected file on his computer. Then, depending on the virus, appropriate action is taken by the user.

instances of the worm will be noticed for various reasons. In some cases, the performance of the network is hampered because the various copies of the worm take too much bandwith. In some other cases, the payload inflicted clearly indicates that there is an infection. Note that there are many other ways one could detect an outbreak, such as having "vigilent staff" or even monitoring from AV vendors. At this crucial instant, the users are aware of the actual threat, and start to elaborate a scheme to stop it. Once the signature is available, any user that logs into his computer can automatically update his AV software with the last signature.

As more user integrate the new AV, uninfected computers will then be immunized against the worm, and infected computers will gradually be detected. In a large corporate network, the typical reaction when a machine is found to be infected is to call the technical support and ask for a complete clean-up. Usually, the first thing to do at this time is to isolate the host so that it cannot infect other hosts: the user pulls the cables and shuts the computer down to prevent further effects of the payload. That is, the infected computer goes from an *infectious* state to an *infected but not infectious* state (here called *detected* state).

The duration of this state depends now on how quick the technical expert is at cleaning-up the computer. It can take a few minutes to up to a few hours (or even days). Once the computer is cleaned, it is put back into the network and is already immunized to further infections because it has the new virus signature included in its AV definitions.

The worm is eradicated (or extremely infrequent) when all computers are immunized. In practice, there always are a few infections due to incomplete immunization or to some users being unaware of the threat. If the prevalence of the virus is plotted over time, it would look something like what is shown in Figure 3.1.

It is the sequence of events just described that forms the basis of the PSIDR model.

3.2 The PSIDR model

In this section, it is shown how aspects mentioned above are integrated in the model. According to the Progressive Susceptible-Infected-Detected-Removed model, epidemic events in computer networks can be divided into two chronological periods (see Figure 3.2):

1. *The Pre-response period.* First, an initial worm infects one machine in the network. For the next few days (or hours), the worm propagates freely in the network without being noticed by most users. In PSIDR terms, this is modelled as a positive birth rate β and no cure. Susceptible nodes therefore become infectious with probability β if they are in contact with an infected node.

2. *The Response period.* After some time, the worm is detected on some

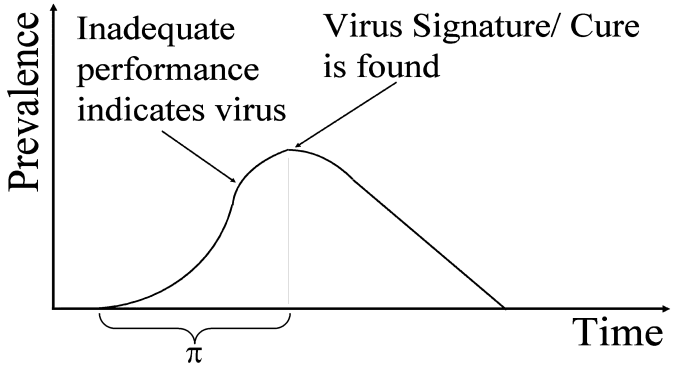

Fig. 3.1: **Time course of an epidemic outbreak.** In the initial phase, the worm spreads unoticed at a fast rate. At some time, users realise that there is an outbreak, and take appropriate action. A virus signature is isolated and made available to computers in the network (at time $t = pi$). One the one hand, uninfected computers are directly immunized. Secondly, infected computers are gradually cleaned-up and immunized so that the prevalence decreases smoothly.

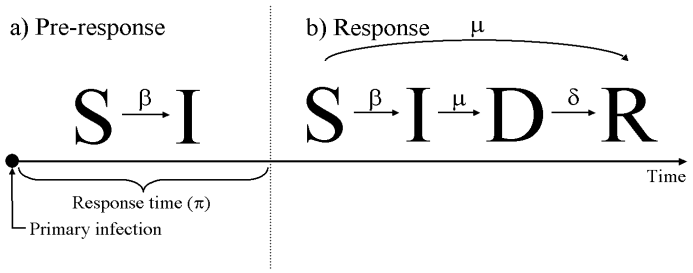

Fig. 3.2: **The PSIDR model for technological networks.** In the *pre-response* period, the virus spreads at a rate β without being cured from infected hosts. At time=π (start of the *response period*), new infections are still made at a rate β, susceptible hosts are immunized at a rate μ, and infected hosts are detected at a rate μ and then cured at a rate δ.

machines and immediate action is taken to prevent further spread and to cure infected computers. A worm signature is extracted and included at a certain rate in the antivirus (AV) software of most machines in the network. Machines that were not infected then become automatically immune to the worm, and previously infected machines are being detected at a certain rate (depending on how often the AV update is made). These machines are then isolated, cured and immunized against further infection. Again, in the PSIDR model this period is modelled with the same birth rate as before, but this time susceptible nodes are immunized at a rate μ, and infectious nodes are detected at a rate μ and then cured at a rate δ. The rate μ represents the speed of the distribution of the AV signature.

The only detail left is the time when the system goes from the *Pre-response* period to the *Response* period. In the PSIDR model, this time is represented by a parameter π, which can take an arbitrary value. This parameter represents the time it takes to have an AV signature since the first worm infection occured.

3.2.1 Specific contributions of the PSIDR model

As for the SIS, SIR and SEIR models, the PSIDR model is best described as a sequence of states with rates of transitions between the states. The description above highlights several factors that should be taken into account when modelling virus propagation in computer networks. These are the main contributions of the PSIDR model to epidemiological models in general.

- *Variability of the cure rate.* Originally, no infected computers are cured. It is only after a certain period of time that instances of the worm start to be identified and cleared from infected hosts. In the PSIDR model, the epidemic event is thus divided into two chronological periods respectively

termed *pre-response* and *response* periods. In the first period, viruses spread at a rate β and are not cleared (the detection (μ) and cure (δ) rates have zero value). Then, at some time determined by a parameter π, the system jumps to the second period where infected hosts can now be cured (the detection and cure rates respectively take fixed nonzero values). Previous epidemic models did not account for this kind of variability of the cure rate.

- *Straight transitions from S to R.* From the time the virus signature is available, susceptible computers can become immune without going through the infected state if the AV software on susceptible hosts are updated before the virus could infect them. In the PSIDR model, this is represented by possible straight transitions from S to R during the *response* period. Specifically, in the *response* period, a susceptible host becomes removed at a rate μ. Direct transitions like this were not included in older models.

- *Detection state.* In the *response* period, an infected (but still functional) computer is identified only when the AV software is updated with the new signature. Once it is detected, the user (or technician) isolates it from the network and spends some time curing it. In the PSIDR model, this is modelled by inserting a new state (called "D" for *detected*) between the I and R states. In the *response* period, infected computers become detected at a rate μ (since it depends on AV update), and then removed at a rate δ. The D state stands for the period when the infected computer is in the process of being cured by a technical expert (or by other means). The inclusion of this state is a proper characteristic of the PSIDR model, never mentioned in other models.

Note that the traditional SIS, SIR and SEIR models do not take these three aspects into account [2]. In the PSIDR model, the epidemic event is thus modelled as a $S \rightarrow I$ system that becomes, after time $t = \pi$, a $S \rightarrow I \rightarrow D \rightarrow R$ system with possible transitions of the type $S \rightarrow R$. The reason why the model is called *Progressive* is now clear: it is because of the *Progression* (or change) in the system's dynamics. The model is formally presented in the next section.

3.2.2 Formal definition of the PSIDR model

In this formal model, it is assumed that the number of computers in the network (N) is constant.

The Pre-response period

For $t < \pi$, the following constraint must be satisfied:

[2] Although it could be said that the SEIR model includes a state similar to the D state (where individuals are infected but no infectious) major differences between both models reside in the order of the states and the straight S→R transitions.

$$S(t) + I(t) = N \tag{3.1}$$

and the differential equations that govern the system are given by:

$$\frac{dS}{dt} = -\beta SI \tag{3.2}$$
$$\frac{dI}{dt} = \beta SI \tag{3.3}$$

In fact it is possible to deduce the second equation from the first one and vice-versa.

The Response period

At time $t \geq \pi$ the following constraint now holds:

$$S(t) + I(t) + D(t) + R(t) = N \tag{3.4}$$

Since there are more than two states, we can represent the evolution of the network by a system of coupled differential equations:

$$\frac{dS}{dt} = -\beta SI - \mu S \tag{3.5}$$
$$\frac{dI}{dt} = \beta SI - \mu I \tag{3.6}$$
$$\frac{dD}{dt} = \mu I - \delta D \tag{3.7}$$
$$\frac{dR}{dt} = \delta D + \mu S \tag{3.8}$$

We can verify that $\frac{dS}{dt} + \frac{dI}{dt} + \frac{dD}{dt} + \frac{dR}{dt} = 0$ which implies that the system satisfies Eq 3.4.
Finally, the starting conditions for the system are: $S(0) > 0, I(0) > 0, D(0) = 0$, and $R(0) = 0$.

3.2.3 Estimation of costs

One advantage of the current model is that it suggests a natural and efficient way of calculating various costs related to the epidemic event.

1. *Fixing cost.* The cost related to fixing the computers is related to how long it takes to cure computers and to how many computers are infected (i.e. were infected and now are in the D state). Therefore, this cost is

measured as the sum of the number of *detected* computers over each time step (the area under the curve calculated as a Riemann integral):

$$\textit{Fixing cost} \quad = \quad \int_{\pi}^{T} D(t)dt \approx \sum_{\pi}^{T} D(t) \tag{3.9}$$

2. *Disruption cost.* The cost of disruption is given by the area under the curve of the number of infected nodes at each time step. It represents how much of the network was affected throughout the outbreak. It is a compound measure of *how many computers are infected* and of *how long* they are infected. It thus captures a lot of information about the costs of the outbreak. As for the fixing cost, the disruption is given by:

$$\textit{Disruption cost} \quad = \quad \int_{t_0}^{T} I(t)dt \approx \sum_{t_0}^{T} I(t) \tag{3.10}$$

3. *Maximum number of infected nodes.* This is also an interesting variable since it gives an idea about the worst state of the system. Indeed, the disruption can yield similar values for very different epidemic events, where the maximum number of infected nodes can differentiate more between types of events.

$$\textit{Maximum number of infected nodes} \quad = \quad \max(I(t))|_{t=t_0}^{t=T} \tag{3.11}$$

4. *Time to immunization.* Real networks are seldom completely immunized (indeed, in large networks, it is not trivial to ensure that *all* machines have been immunized), but they can become mostly completely immune to a worm. Thus, the time it takes to immunize 95% of the network's computers is calculated instead: note that this level of 95% is chosen somewhat arbitrarily; levels of 90% or 99% could also have been chosen. It can be advantageous to immunize the network as quickly as possible to prevent any large outbreak. The time taken to mostly complete immunization is thus measured as a function of the parameter configurations.

In addition to measuring traditional quantities, such as the number of susceptible and/or infected individuals at each time step, these four costs can be measured and used to suggest the best response strategies. Note that models such as SIS, SIR or SEIR *cannot* provide any indication regarding the fixing cost.

3.2.4 Model details

In this section, various aspects of the model are examined in more details to show how it relates to real epidemic events.

The rate β is assumed to be constant and depends on how fast the virus can propagate itself to new hosts. For example, the Code Red Virus (CRv2) could probe hundreds of IP addresses per second [33]. In contrast, email worms are thought to be much slower.

The parameter π represents the time taken to find a signature. Obviously, in the current context, this parameter depends on how fast we are at finding a good countermeasure to the viral attack. However, this is a parameter whose value would likely be reduced by the use of automated systems for computer security. It makes sense therefore to simulate the outbreak for different values of π in order to estimate the relative merits of autonomous security systems.

If $t < \pi$, the rate $\mu = 0$, when $t \geq \pi$, μ takes a specific positive value where $\mu << \beta$. This is because often the anti-virus update is made only once or twice a day, while the worm spreads a lot faster (hundreds of addresses per second for example). The detection rate will also be influenced by the fact that not all computers are switched on every day. In the present context, we could evaluate the effect of a proactive policy where anti-virus update would be more frequent, or of a relaxed one where update would be performed once a week for example.

As of now, the cure rate depends mainly on the number of technical staff available to deal with the epidemic, the time it takes to cure a computer, and the amount of time each staff member can spend on the problem. Also, since cures are not always effective, some computers may not be cured the first time. In today's networking reality, most cures are performed manually, which means that the cure rate will be much lower than the birth rate. Here again, the effect of autonomous security systems can be evaluated, where the cure rate δ would likely be increased.

3.3 Limits of the PSIDR model

The PSIDR model extends previous models to offer a better account of technological epidemics. However, here are some aspects that it does not capture.

- *Variabilty in cure rate δ* Indeed, the more infected machines there are, the more people are assigned to fighting it. That is, $\delta \propto I$ which is likely to influence the time needed to get rid of the worm. The exact relation between I and δ may be linear or non-linear. More data is needed in order to settle this issue. The PSIDR model is easily extendable to a variable cure rate.

- *Variability in birth rate β* In the case of self-launching worms, the spreading rate is partly determined by how fast the worm will probe new IP adresses. For example, in the case of CodeRed, the worm was programmed to stop probing new hosts at midnight on the 20th of July. Other worms

also had this stopping feature, causing the birth rate to be variable. Apparently, the outcome of the epidemic outbreak was largely determined by this feature [33]. Examples of models with periodicity include [6] for a modelling of computer viruses, and [14] for measles epidemics.

- *Other periodical parameters* Some worms inflict damage periodically. For example, the *Klez.e* worm only damages infected machines on the 6^{th} of every odd number month (January, March, May, etc.)[20]. It is not clear how this aspect should be incorporated in the model, but it is likely to play on the probability of detecting instances of the worm.

3.4 Summary

The PSIDR model is an alternative to traditonal models. Unlike previous models, it encompasses part of the variability in the cure rate δ, direct transitions from S to R, as well as the isolation period (the D state) between the infectious and removed states. In order to assess the relative benefits of the new model, simulations are conducted with various parameter configurations. In addition, the PSIDR model is explicitely compared to the SIR model in order to show the influence of straight S→R transitions.

However, before this is done, simulations of simpler models are performed in order to provide a good understanding of simpler dynamics, and to validate the subsequent results. In return, this may facilitate the understanding of the more complex PSIDR model.

4. SIMPLE EPIDEMICS IN HOMOGENEOUS AND SCALE-FREE NETWORKS

In this chapter, four series of experiments are reported that serve both as a reality check for further simulations, and as a way to get a good grasp of the dynamics of simple models.

The first set of experiments shows that very few repetitions are necessary to yield a reliable estimate of the average behaviour of homogeneous (HM) networks. The SI (Susceptible-Infected) model is used for this purpose. The SI model can be considered as SIS model with a cure rate of zero, or as the PSIDR model in the *Pre-response* period. The fact that few repetitions are necessary implies that simulation time can be reduced for HM networks.

Moving on to a more complex model, the second section reports simulations on the SIS model in HM networks. It is possible to derive a prediction relating to the prevalence (fraction of infected nodes) at steady-state (when the prevalence does not change over time). The effect of *timeslicing* (dividing each timestep in smaller periods) is also illustrated in this section.

The third set of simulations explores more complex issues related to the SIS model in scale-free (SF) networks. In SF networks, the situation is not as trivial as it is in HM networks. Work by Pastor-Satorras [44] offers some predictions about the SIS model in SF networks of different sizes, for various birth rates (β), and for various network densities (as determined by m). Some of these predictions are validated by current results.

Finally, more complete simulations of the SI model are conducted in HM and SF networks. The purpose of this is twofold: because it allows for a reality check of simple dynamics, and because the SI model corresponds in fact to the PSIDR model in its first phase (The *Pre-response* phase). The results reported in this final section pave the way for the second phase of the PSIDR model in the next chapter.

The method for creating SF networks is reported in Appendix B.

4.1 Variability in HM networks

A few experiments are conducted on the SI model to illustrate the fact that the *true* behaviour of HM networks can be approximated with a relatively small number of trials.

It is expected that the number of repetitions will not influence the average prevalence recorded over time since the HM network is by definition not af-

fected by any *heterogeneities*. For example, let's consider the $Y = \binom{N}{x}$ ways of infecting x nodes at random in a network of N nodes. In a HM network, these Y configurations are in fact the *same* configuration. In a SF network however, due to heterogeneities in the network, these Y configurations are likely to be different. Heterogeneities in the SF graph are present in two forms: large variations in the number and in the identity of neighbouring nodes. Therefore, in a SF graph, it is important to average over a large number of trials in order to smooth out the effects of heterogeneities. While there are *a priori* reasons to run many repetitions on the SF network, it may not be necessary to do as many repetitions for HM networks.

4.1.1 Method

One HM network of 6250 nodes is simulated for 100 timesteps. At each timestep, each node is infected at a rate β if it is connected to at least one infected neighbour (i.e. if one of its edges leads to an infected node). There is no cure in the SI model, thus all nodes become infected given enough time. The number of infected individuals is recorded at each timestep. Each timestep is divided in ten small timeslices for continuous approximation (see next section). Update is performed in parallel (the state of all nodes is updated at each timestep).

The simulations are run for 10, 50, 100, 200 and 1000 repetitions. Thus, it is possible to see how well the behaviour of the system is approximated as a function of the number of trials.

4.1.2 Results and discussion

The number of infected individuals is plotted over time in Figure 4.1. The number of repetitions does not influence the average prevalence. Therefore, when simulating the PSIDR model, the simulation time for HM networks is reduced by setting the number of repetitions to around 100 (a conservative number) for most simulations of homogeneous networks.

4.2 SIS model in HM networks

This section and the next cover the SIS model in HM and SF networks respectively. A large number of recent studies [34, 44, 42, 43, 46, 7, 45] assume that the number of infected neighbours does not significantly affect the probability of infection. For simplicity, the same assumption is used here and in all subsequent simulations. In HM networks, this has the important consequence of changing the differential equation governing the spread of the worm from

$$\frac{d\rho}{dt} = \beta\rho(1-\rho) - \delta\rho \tag{4.1}$$

to the somewhat simpler

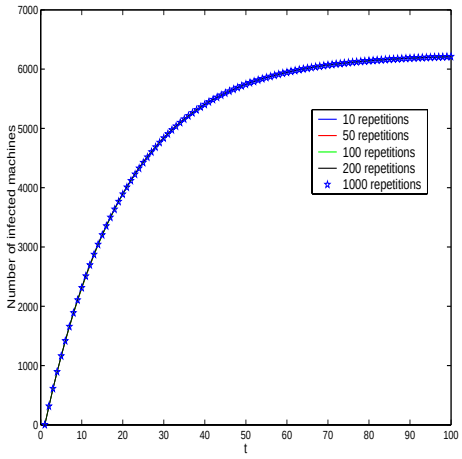

Fig. 4.1: **Average number of infected machines as a function of the number of repetitions.** All lines completely overlap each other and are not easily distinguished. Since there is no noticeable difference between the averages obtained with few or many trial, it is possible to cut simulation time by performing only a few repetitions (10 to 50 repetitions).

$$\frac{d\rho}{dt} = \beta(1-\rho) - \delta\rho \tag{4.2}$$

This model is valid as long as there is at least one infected node in the system. The prevalence at equilibrium (ρ_{eq}) can be determined by setting $\frac{d\rho}{dt} = 0$ and solving for ρ:

$$\rho_{eq} = \frac{\lambda}{1+\lambda}$$

where $\lambda = \frac{\beta}{\delta}$. The parameters β and δ are the *birth* and *cure* rates respectively. As the birth rate is increased comparatively to the cure rate, it is observed that $\lim_{\lambda\to\infty} = 1$. Conversely, if the cure rate is increased and the birth rate lowered, $\lim_{\lambda\to 0} = 0$. Thus, in this recent version of the SIS model (on HM networks only), there does not seem to be any epidemic threshold (except for the critical point at $\lambda = 0$).

4.2.1 Method

A methodology similar to the one employed in [44] is used here. A single node is infected at the beginning of each simulation run, and the propagation is made according to the SIS model. The state of all nodes is updated at each iteration (in parallel) for a total of 100 iterations. If a node has at least one infected

neighbour, it becomes infected at a rate β. At the same time, infected nodes are cured at a rate δ. Each simulation run is repeated 10 times. The birth rate is varied ($\frac{1}{10} \leq \beta \leq \frac{6}{10}$) across simulations. In all simulations, the cure rate $\delta = 0.4$. The system reaches a stable state in the very first trials, and the prevalence at equilibrium is recorded and plotted in Figure 4.2 as a function of $1/\lambda$ ($\lambda = \beta/\delta$) and network size. Networks used are composed of $N = 6.25 \times 10^3$, $N = 1.25 \times 10^4$ and $N = 2.5 \times 10^4$ nodes.

Equation 4.2 is a continuous model of the system. However, numerical simulations are based on discrete timesteps, which is generally considered an inexact way to simulate real processes: discretization can induce error in the approximation of the continuous case. To clean this artifact, timesteps are divided in small timeslices, and the values of the various transition probabilities (i.e. β, δ, etc.) are divided by the number of slices. For example, if timesteps are divided in n slices, then, over one full timestep, the birth rate equals $n \times \frac{\beta}{n} = \beta$. The effect of timeslicing on the accuracy of numerical results is also studied by running simulations with different number of timeslices (1, 10 or 100 slices). The differences between the theoretical (continuous) prediction and the numerical (discrete) results are reported in Figure 4.3.

4.2.2 Results and discussion

Numerical results confirm the theoretical predictions (see Figure 4.2). This implies that the SIS model, as it was recently proposed, is accurately instantiated in the current simulations. Note in Figure 4.3 how substantial improvements in accuracy are achieved by using even a small number of timeslices (10 slices). Moreover, using 100 fine slices is too computationally expensive for large-scale simulations. Therefore, in subsequent simulations of the PSIDR model, simulations will be run using 10 timeslices.

4.3 SIS model in scale-free networks

In this section, the SIS model is studied in SF networks. It has been shown elsewhere [44] that SF networks updated according to the SIS model quickly reach a steady state. At steady-state, the virus prevalence does not change over time. If the simulated worm spreads quickly enough, the system reaches a nonzero steady-state. However, if the worm spreads slowly, the prevalence tends to zero (another steady-state called *absorbing state*)[1]. In this section, the first experiments concern the effects of network size and spreading rate on the prevalence at steady-state in SF networks.

In the SF networks used here (Barabási-Albert model), nodes have an average degree of $2m$, where m is a free parameter (its value is chosen arbitrarily when creating the network). This parameter can be thought of as the network density. For a low m, most nodes will have very few connections (low density

[1] This is only true for finite networks. Theoretical results indicate that, in infinite SF networks, a nonzero state is always reached, independently of the spreading rate [44].

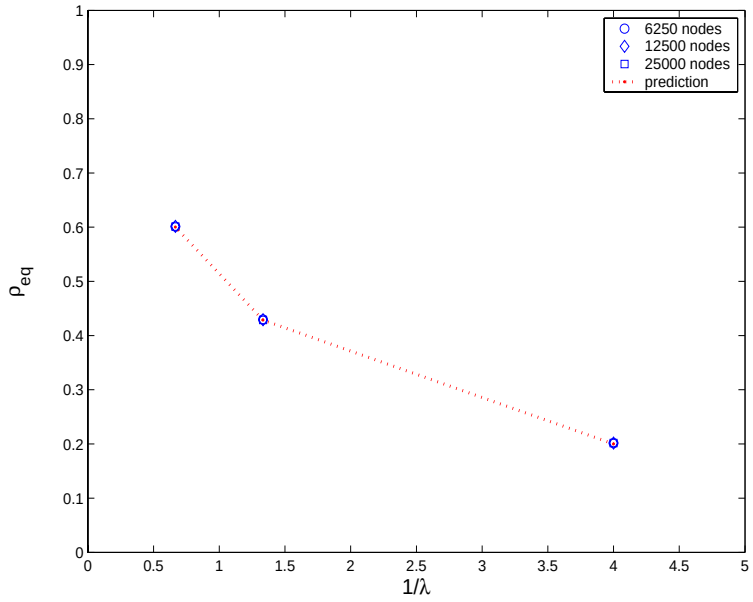

Fig. 4.2: **SIS model in HM networks.** Prevalence at steady-state in HM networks as a function of spreading rate and network size (100 timeslices). The prediction (dotted line) is well approximated by numerical simulations. Markers for networks of different sizes completely overlapp each other.

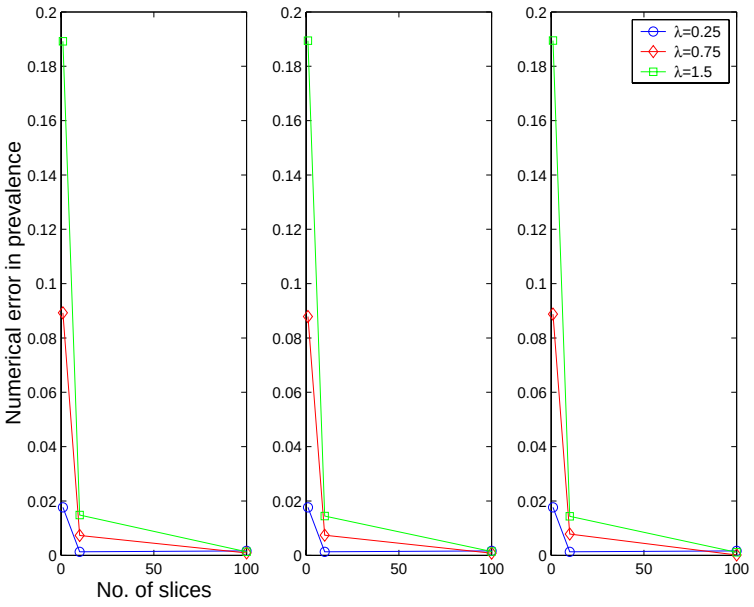

Fig. 4.3: **Error in approximation of prevalence due to discretization.** The error is computed as the difference between theoretical predictions and numerical results and is displayed as a function of network size, inverse spreading rate and number of slices. Increasing the number of slices decreases the error. Major improvements are gained by going from 1 to 10 slices.

of connections), but for a high m, most nodes will have many neighbours (high density). The second set of experiments explores the effects of network density and network size in SF (BA) networks.

4.3.1 Prevalence at stable state

Method

For simplicity, the same methodology used in [44] applies here. At the beginning of each simulation run, half of the nodes are infected, and the propagation made according to the SIS model. The state of all nodes is updated at each iteration (in parallel) for a total of 100 iterations. If a node has at least one infected neighbour, it becomes infected at a rate β. At the same time, infected nodes are cured at a rate δ. In all simulations, $\delta = 1$. Each simulation run is repeated at least 1000 times (to allow for at least 1000 different starting configurations). The birth rate is varied ($\frac{1}{20} \leq \beta \leq \frac{1}{8}$) across simulations. The system reaches a stable state in the very first trials, the prevalence at equilibrium is recorded and plotted as a function of $1/\lambda$ and network size. The major difference between the current experiments and the ones reported in [44] resides in the network sizes used: here the networks will be of $N = 6.25 \times 10^3$, $N = 1.25 \times 10^4$, $N = 2.5 \times 10^4$ and $N = 1 \times 10^5$ nodes. Larger networks were used in the original report, and as such they bring a different analysis because they are susceptible to finite-size effects to a lesser extent. Also, since timeslicing was not mentioned in the original report, timesteps are not divided in fine slices.

Results and Discussion

Figure 4.4 shows the data currently obtained, and the data obtained by Pastor-Satorras and Vespignani [44]. Since no numerical value is available, their data was retrieved by taking two points from their figure, and fitting a curve of the form $y = Ce^{-x}$ between the points.

The first result of interest is the relation between the final prevalence and the spreading rate. Prevalence decreases as a function of $1/\lambda$, which is indicated by theoretical results $\rho \approx 2e^{\frac{-1}{m\lambda}}$ reported by Pastor-Satorras [44].

The second important result is the independence of the final prevalence from the network size. Although it is true for high birth rates, it is obviously false for smaller λ (smaller β). This difference in network size was not observed in Pastor-Satorras's paper. Indeed, in his report, data for all network sizes lie on the dotted line. It is useful to note that he used larger networks to plot his data. The discrepancy between the current results and their results might thus be related to finite-size effects.

This brings up the third point: the critical threshold where the outbreak dies out. The striking fact about scale free networks is the absence of an epidemic threshold as demonstrated by $\rho \approx 2e^{\frac{-1}{m\lambda}}$. However, this relation holds only if the the size of the network is assumed to be infinite. Indeed, as it was demonstrated later by Pastor-Satorras [43], the critical threshold in BA networks is given by

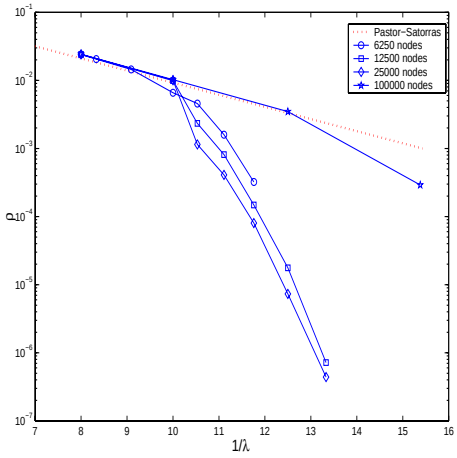

Fig. 4.4: **SIS model in SF networks.** Final prevalence as a function of network size and $1/\lambda$. The original data of Pastor-Satorras is plotted as a dotted line. For low $1/\lambda$, prevalence is independent of network size. However, for high $1/\lambda$, larger networks display a higher prevalence than smaller networks. In general, prevalence decreases with the inverse spreading rate ($1/\lambda$). Discrepancies between current data and previous results are explained by finite-size effects (see Figure 4.5)

$\lambda_c = \frac{<s>}{<s^2>}$, where $\lambda = \frac{\beta}{\delta}$, s is the average degree, and s^2, the degree variance. In an infinite network, $s^2 \to \infty$, hence $\lambda_c \to 0$. However, in a finite network the variance is finite and the epidemic threshold is introduced again. The average epidemic threshold was calculated for each network size and plotted against the same results in Figure 4.5. As predicted, the epidemic threshold tends to zero as network size increases. Also, the prevalence reaches the absorbing state ($\rho = 0$) in the neighbourhood of the epidemic threshold (especially for 6250 and 25000 nodes). One exception seems to be the network with 12500 nodes, where there still is a nonzero (albeit very low) prevalence after the epidemic threshold. The threshold for 100000 nodes is even lower and is not plotted on this graph.

Inconsistencies might be attributed to noise in the region of low spreading rate (instability of the prevalence). It could also be due to a too small number of repetitions (1000 for each network): since there are high heterogeneities in the network's connectivity, sampling just a few possible starting points can affect the average. Finally, finite networks differ from infinite networks in two ways:

1. The finiteness of the variance
2. Departures from the theoretical distribution which strictly holds only for infinite networks.

Pastor-Satorras' analysis of finite-size effects only takes into account the first type of effect [43]. Considering all these factors, it is not surprising that the

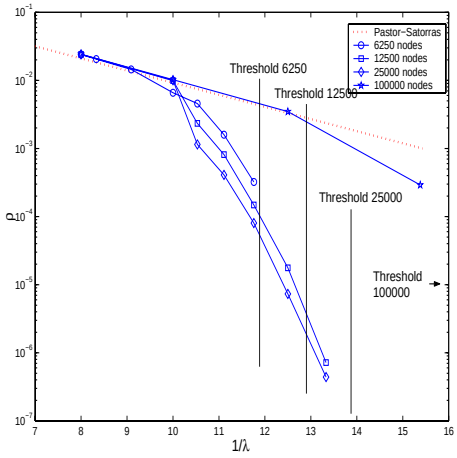

Fig. 4.5: **SIS model in SF networks with threshold.** Final prevalence as a function of network size and $1/\lambda$ with thresholds. The epidemic thresholds for each network size are plotted as straight lines. Departures from previous data (dotted line) are explained by finite size effects. As the spreading rate approches the epidemic threshold, prevalence tends to zero.

observed results for finite nets only partially agree with the finite threshold prediction.

In general, the results for final prevalence can be explained by the existing theoretical framework. This implies that the outcome of an outbreak will likely be worse in large networks, than in small networks. This is obviously bad news for large networks like the internet.

4.3.2 Evolution of prevalence and network density

Another prediction from the relation $\rho \approx 2e^{\frac{-1}{m\lambda}}$ is that the prevalence should be higher in dense networks (density is determined by the parameter m). One restriction imposed by the BA algorithm is that $m \leq m_0$. This means that m is always finite and hence the prevalence can never reach 1 (unless the spreading rate is itself infinite).

Method

Simulations of the SIS model were performed as before on networks where m=3, 4, 5 for a higher spreading rate (necessary to attain an endemic state in small networks) to verify that the obtained prevalence would follow the prediction (simulations were run on 10 SF networks, for 100 iterations and 1000 repetitions).

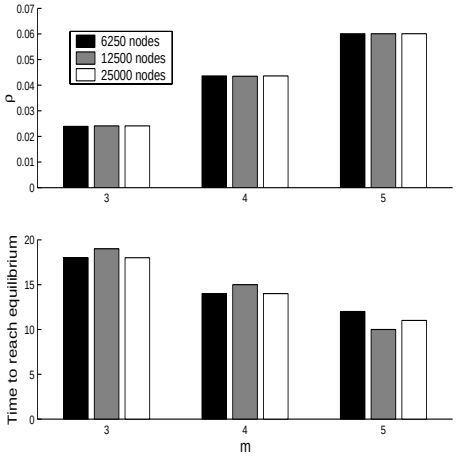

Fig. 4.6: **Density and network size in the SIS model on SF networks** for densities of m=3, 4 and 5 and network sizes of N=6250, 12500 and 25000 ($\beta = 0.125$, $\delta = 1$). *Upper graph*) The final prevalence is independent of network size but scales with density. *Lower graph*) The time to reach the steady-state is independent of network size but is inversely related to density.

Results and Discussion

The plot in Figure 4.6 (top) generally confirms a higher prevalence for larger m. Moreover, the final prevalence is independent of network size.

Previous numerical and theoretical work offered no clues about the progression of prevalence over time. Current data suggests that the time taken to reach the steady-state is independent of network size but is inversely related to density (see Figure 4.6 (bottom)).

4.3.3 Discussion

The data obtained here confirms earlier results and predictions[2]. Finite-size effects appear to be less prominent for high spreading rates ($\lambda = \frac{\beta}{\delta} \approx 0.125$). In the PSIDR model, there is no straightforward definition of the spreading rate (since there is also a μ parameter to consider). Nonetheless, the ratio of the birth rate β to the cure rate δ should remain at a high value in order to minimize artifacts: the following simulations of the PSIDR model are made accordingly.

4.4 SI model in HM and SF networks

The SIS and PSIDR models cannot be analysed in the same way since they do not contain the same parameters, and the dynamics are inherently different

[2] Although there were some problems concerning the replication of other aspects of the data reported in [44]. The details are reported in Appendix C.

(the SIS can reach a nonzero steady-state, while the PSIDR will always tend to zero). The *Pre-response* period of the PSIDR model (formally equivalent to a SI model) is covered in this section. The results obtained serve both as a reality check and as a way of predicting virus prevalence when the response starts. Homogeneous and scale-free networks are used since the PSIDR model targets both kinds of technological networks.

4.4.1 Method

The SI model is simulated on HM and SF networks of 6250 and 25000 nodes. The state of all nodes is updated at each timestep (in parallel) for a total of 150 iterations. If a node has at least one infected neighbour, it becomes infected at a rate β. Timesteps are divided in 10 small timeslices to minimize effects of time discretization. The number of infected individuals is recorded at each timestep. Simulations are repeated 100 times in the case of HM networks, and 1000 times for SF networks. Results for both kinds of networks are treated in separate sections.

4.4.2 Results

Fully connected (HM) graph

In a homogeneously mixed finite network, since $\delta = 0$, the equation governing the spread corresponds to the logistic growth:

$$\frac{\partial \rho}{\partial t} = \beta \rho (1 - \rho) \tag{4.3}$$

Thus, for any spreading rate $\beta > 0$, given enough time, the virus will spread to every computer in the network if there is no response. The solution to this equation is given by:

$$\rho(t) = 1 - (1 - \rho_0) e^{-\beta t} \tag{4.4}$$

In the simulations, $\rho_0 = \frac{1}{N}$, where N corresponds to the network size. Figure 4.7 displays the prevalence (fraction of infected nodes) in homogeneous networks of different sizes and with different values for β. The time to 95% prevalence is independent of network size, which is explained by the fact that the virus spreads faster in larger networks. The predictions from the differential equation are plotted in full lines. The predictions agree well with the data. In general, this plot shows that the growth model is simulated adequately. Note that this also constitutes a sanity check for the SIS model where $\delta = 0$.

Virus spreading in homogeneous networks is rather quick due to the fact that a virus can reach all the computers from a single node. This is alarming for corporate email networks (some of which have a fully-connected topology), and in some popular peer-to-peer networks, where each individual can potentially connect to every other individual [12].

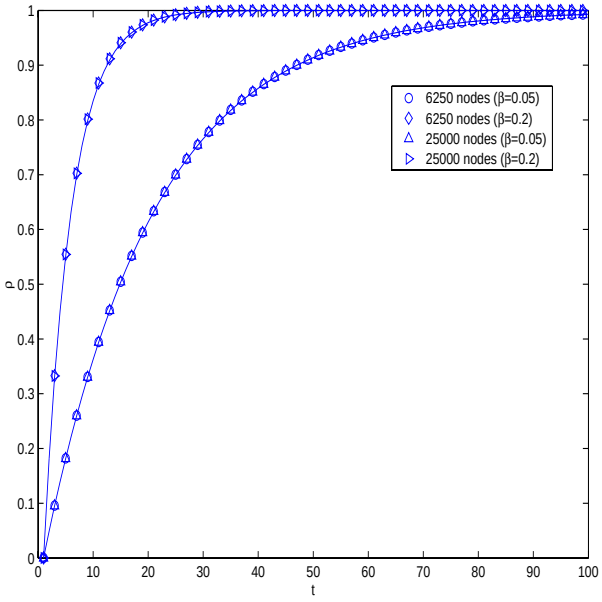

Fig. 4.7: **Pre-response phase in HM networks.** Virus prevalence over time in HM networks as a function of network size and birth rate (predictions plotted as the full lines). The predictions are confirmed by the data. Given enough time, all nodes become infected by the virus, independent of network size.

The fact that analytical predictions and numerical results agree well with each other suggests that simulations are not corrupted by implementation details. Moreover, the solution to the differential equation allows us to predict how many machines are infected when the antiviral response comes into play, hence we do not need to simulate the period before $t = \pi$ in the case of homogeneous networks.

Barabási-Albert (SF) model

In the case of scale free networks, the work of Pastor-Satorras gives indications about the final state of the system, but it does not provide clues about the progression of prevalence in time. Moreover, he assumes a cure rate of 1, while the cure rate here is zero.

The number of infected individuals, as shown in Figure 4.8, follows a logistic growth ($Verhülst$ growth [35]). This classical shape is given by the solution to the differential equation $\frac{d\rho}{dt} = \beta\rho(1-\rho)$. However this last equation assumes a homogeneous mix hypothesis, where nodes have approximately the same number of neighbours. What the figure shows is that the $Verhülst$ growth still holds in cases with high heterogeneity and with various birth rates. Although it may not be possible to prove it analytically, it is easy to see how the growth in a heterogeneous network relates to that in a homogeneous network.

The equation for the relative prevalence is:

$$\frac{\partial \rho_k}{\partial t} = \beta k(1-\rho_k)\theta(\beta) \tag{4.5}$$

In this version of Pastor-Satorras' original formulation, the cure rate has been set to zero. The $\theta(\beta)$ term represent the probability of a node to become infected. It is analogous to the ρ term in the classical logistic growth function. Therefore, since all ρ_k progress similarly to the logistic growth, the total prevalence $\rho = \sum_k \rho_k$ follows the same course.

Contrary to the homogeneous case, there seems to be a slight difference in the evolution of prevalence due to different network sizes. In the HM net, the average number of neighbours increases directly with network size, while the diameter remains the same (diameter=1). The speed of infection is therefore the same whatever the network size. However, in the SF net, the average degree remains the same for all network sizes, but the diameter increases logarithmically with the size. In SF networks, the speed of infection is slower at the beginning for larger networks, but gets faster than smaller networks after some point in time (around t=33 for $\beta = 0.05$). The outbreak's speed may become faster because highly connected nodes in large networks have a higher degree than their counterpart in small network: therefore, once these nodes are infected, they can reach a larger number of nodes in a short time. But at the beginning it is slower because a few more hops are needed to first reach them. Note that the difference between small and large nets is increased by lowering the birth rate.

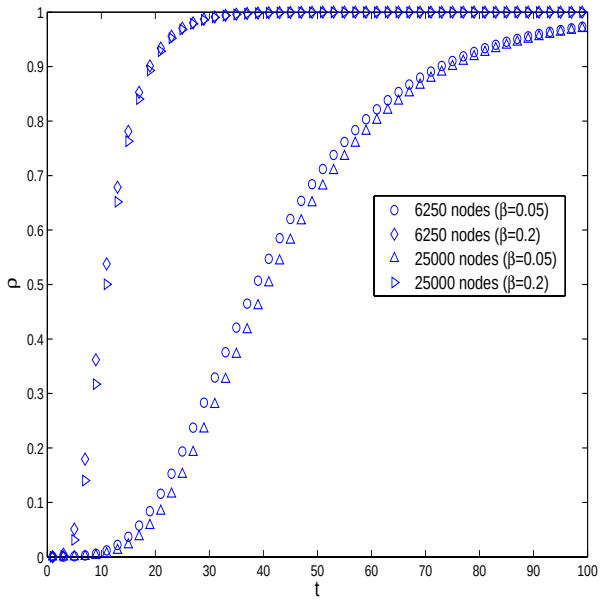

Fig. 4.8: **Pre-response phase in SF networks.** Virus prevalence in SF networks over time as a function of network size and birth rate. Slight differences in network size may be attributed to the topology.

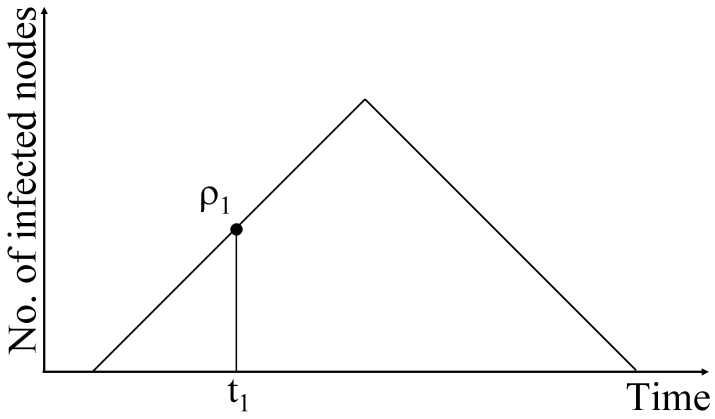

Fig. 4.9: **The Pre-response phase and the initial prevalence.** Different starting conditions correspond to different timesteps. In HM networks, starting a simulation at time $t = 0$ with one infected node and running the updating until time step t_1 is equivalent to starting at time t_1 and infecting ρ_1 nodes at random.

4.4.3 Discussion

To summarize the results, the growth of the virus population is faster in homogeneous networks than in scale free networks, and follows a different growth curve. It is possible to predict the prevalence in HM networks using the solution given for the differential equation. The simulation time is thus reduced for the full PSIDR model. Indeed, the initial period (before $t = \pi$) can be skipped by setting the initial number of infected nodes to a predefined value. For example in PSIDR, at time t = 1, perhaps only one computer will be infected, and at t=10, 100 computers may be infected. By setting the initial number of infected nodes at 100 (randomly selected nodes) we could start simulating the epidemic as if it had started 10 time steps before (the situation si illustrated in 4.9).

In practice, this equivalence relation allows us to cut the simulation time by π steps for homogeneous networks, as long as we acknowledge that different initial conditions correspond to a different time in the epidemic event for the introduction of the virus signature. This is an important results because simulations of the PSIDR are surprinsingly computationally intensive and any reduction in the simulation time deserves consideration.

The same cannot be said of heterogeneous networks. Pastor-Satorras has indeed demonstrated that the probability of a node to be infected is proportional to its degree [42] in the SIS model at stable state. Even if the assumption of

stable state is removed, and in the PSIDR model, it is still true that a node with many neighbours is more likely to become infected than a node with few neighbours: since infection can only occur via transmission of the virus by a neighbour. That is, the probability of a node with k links to be infected is proportional to k. However, in a simulation started by infecting N nodes at random, the probability of a node with k links to be infected is proportional to $P(k)$ (the fraction of nodes with k links). In a BA network, these two quantities are inversely related as shown by $P(k) = \frac{2m^2}{k^3}$ [3]. The distribution of the fraction of infected nodes of degree k will not be the same if the simulation is started by infecting 1 node and iterating the model π steps, than if the simulation is started at time $t = \pi$ with N infected nodes.

Moreover, if we want to test the effect of a virus slowing policy[3], we may need to simulate the period before $t = \pi$.

4.5 Summary

This chapter sets the foundations for further simulations on the PSIDR model. It was shown that numerical accuracy could be greatly improved by dividing timesteps in ten fine slices: this general principle is applied to all subsequent simulations. It was demonstrated that very few repetitions are necessary to obtain reliable estimations of the properties of HM networks. The simple dynamics of the SI model also imply that the *Pre-response* period does not have to be simulated in those networks. This can result in a significant reduction in simulation time. In SF networks, however, the *Pre-response* period must be included in the simulations. Finally, results on the SIS model confirm various predictions and indicate that a high birth rate β should be used in subsequent simulations to minimize the magnitude of finite-size effects. Indeed, simulations on the PSIDR model are more computationally intensive than other simulations (due to the complexity of the model), and will therefore be run only on small networks. Subsequent work could address the issue of network size on the behaviour of the model.

[3] Such mechanisms are active at any time, like in the virus throttling concept [54].

5. SIMULATIONS OF THE PSIDR MODEL

This chapter reports various simulation experiments made with the PSIDR model. Let's recapitulate the chain of events included in the PSIDR model, shown again in Figure 5.1:

1. *The Pre-response period ($S \rightarrow I$).* First, an initial worm infects one machine in the network. For the next few days (or hours), the worm propagates freely in the network without being noticed by most users.

2. *The Response period ($S \rightarrow I \rightarrow D \rightarrow R, S \rightarrow R$).* After some time, the worm is detected on some machines and immediate action is taken to prevent further spread and to cure infected computers. A worm signature (see Chapter 3) is extracted and included at a certain rate in the antivirus (AV) software of most machines in the network. Machines that were not infected then become automatically immune to the worm, and previously infected machines are being detected at a certain rate (depending on how often the AV update is made). These machines are then isolated, cured and immunized against further infection.

The last section of the previous chapter concerned the *Pre-response* period. It was shown (both numerically and analytically) that, given enough time, the worm will infect all computers. Faster worms require less time to infect all machines. In this chapter, properties of the second period are investigated numerically. Since the PSIDR model contains many free parameters, many different configurations can be tried in order to gain an adequate understanding of the model. Here only a subset of values are explored to show basic dynamics of the model.

The first set of experiments is meant to give a general overview of the model. The time to initial detection (π) is set to different values to illustrate the main effect of this factor.

In the second part, values for different parameters - time to initial detection (π), detection and immunization rate (μ), and cure rate (δ) - are varied across simulations. The emphasis is put on the interactions involving the π and μ, π and δ, and μ and δ parameters. Current strategies to cope with computer worms are modelled with these parameters, and one of the main goals of this paper is to evaluate the efficiency of these methods.

A new way to cope with epidemics is to slow down the spread of the worm [54]. In the present context, this strategy can be tested by simulating slower birth rates (β). A third set of experiments investigates this issue. A proper

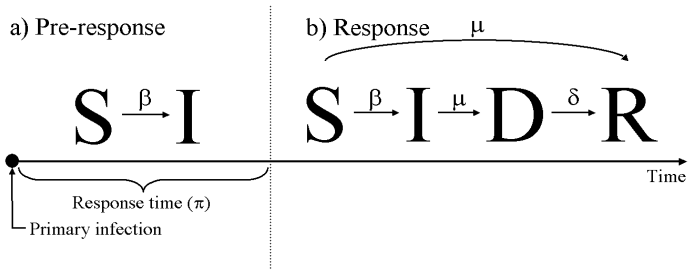

Fig. 5.1: **The PSIDR model for technological networks.** In the *pre-response* period, the virus spreads at a rate β without being cured from infected hosts. At time=π (start of the *response period*), new infections are still made at a rate β, susceptible hosts are immunized at a rate μ, and infected hosts are detected at a rate μ and then cured at a rate δ.

study of this effect should involve simulations of the interactions between β and the parameters π, δ, and μ. Only basic dynamics are shown here due to the computational demands of these simulations.

Finally, the SIR model is compared to the PSIDR model when $\pi = 0$ in order to show the influence of the direct transitions from S to R (one of the main features of the PSIDR model).

Unlike for the SIS model, the focus is not explicitly put on the existence of a possible epidemiological threshold. Indeed, it is not clear how the four different parameters should be related to each other in order to represent an epidemic threshold.

5.1 Method

Ten different scale-free (SF) networks of 6250 nodes and one homogeneous (HM) network are used in the simulations. The state of all nodes is updated at every timestep (parallel update) for at least 150 iterations. Timesteps are divided in 10 small timeslices in order to approximate continuity and asynchrony in the system. As in [44], simulations are repeated at least 100 times (to up to 1000 times) in the case of SF networks. Due to their robustness to noise (see Chapter 4), only 50 repetitions are made for HM networks.

At each timestep, the numbers of *infected*, *detected* and *removed* machines are respectively counted to provide the raw data. The four different costs mentioned in Chapter 3 (fixing cost, disruption, maximum number of infected nodes, time to immunization) are calculated from this data.

5.1.1 Update rule

The update is performed according to the PSIDR model:

1. In the *Pre-response* period ($t < \pi$), a node is infected at a rate β if it is connected to at least one infected node.

2. In the *Response* period ($t \geq \pi$), susceptible nodes become either *infected* at a rate β or *removed* at a rate μ. At the same time, already infected nodes become *detected* at a rate μ and detected nodes are *removed* at a rate δ.

5.1.2 Estimation of parameters

Instead of calculating specific values for each of the parameters β, δ, μ and π, their values are approximated in the following manner.

- *Spreading rate* Since worms spread considerably faster than they are detected or removed, the value of β should be higher than the detection (μ) and cure (δ) rates.

- *Response time* The number of timesteps before an initial detection (π), is not constrained by any of the other parameters. Thus, values in the interval $0 \leq \pi \leq 20$ and also $\pi = 40$ are used to provide a general estimate of the effect of this parameter.

- *Detection rate* The value of the detection rate is in between that of the birth and cure rates due to the fact that it is partially automated (the AV update is made automatically at least once daily by the software).

- *Cure rate* Since curing requires manual labour, it is rather slow: curing a few dozens of computers can take days. Therefore, the cure rate is set to a low value. In the following simulations, various values for δ are simulated between $\delta = 0.03$ and $\delta = 0.10$. The actual cure rate (i.e. in real networks) is probably somewhere near the lowest boundary $\delta = 0.03$.

5.2 Results

5.2.1 General overview of the model

Figure 5.2 shows the behaviour of the PSIDR model over time.

The number of infected individuals increases steadily, peaks near $t = \pi$, and then slowly decreases to zero. In contrast, the number of detected individuals remains zero until the *Response* period, then increases until it reaches a certain peak, following what it slowly goes down to zero. Finally, the number of immunized/cured individuals increases from the time $t = \pi$ until it saturates over all the network.

The number of infected machines in the PSIDR model follows a similar course than in the SIR model (see [34]). That is, there is a sharp increase followed by a slow decrease, ending ultimately to zero. The exponential increase in prevalence has been observed in the case of the Code Red (Crv2) worm [10] outbreak of July 2001.

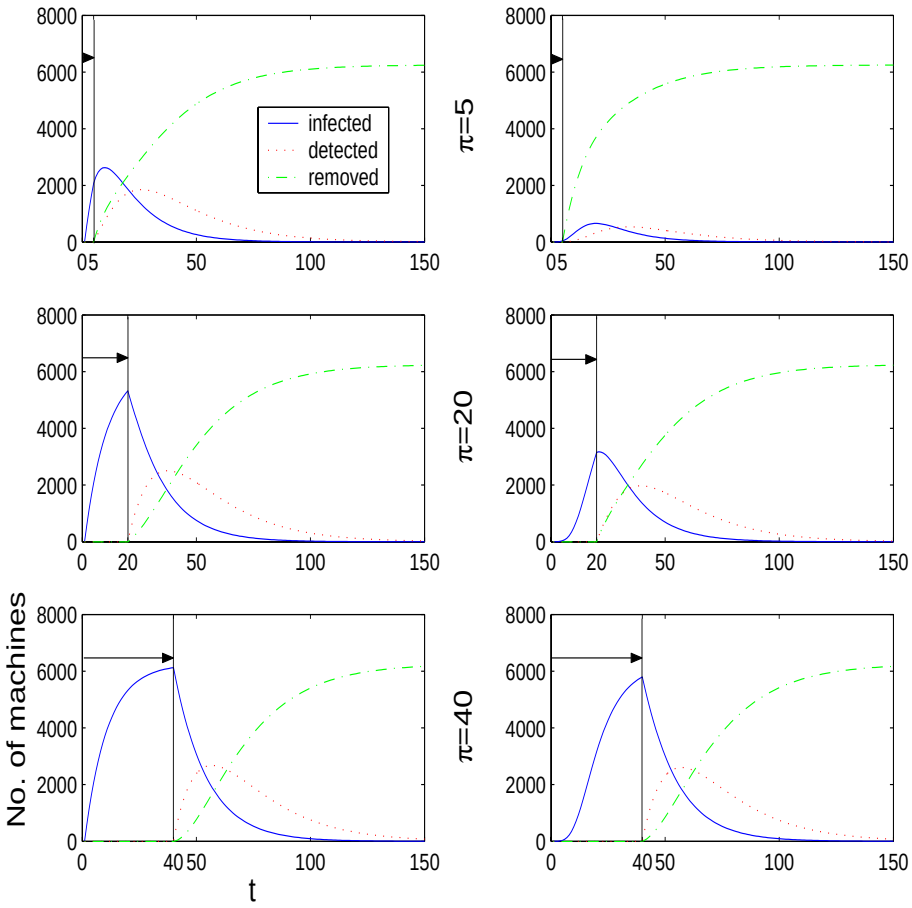

Fig. 5.2: **Overview of the PSIDR model.** HM nets are displayed on the left and SF nets are on the right. From top to bottom, $\pi = 5, 20, 40$ ($\beta = 0.1, \delta = 0.05, \mu = 0.07$). Initially, one node is infected, and the virus propagates freely in the network. At $t = \pi$, the numbers of *detected* and *removed* machines start to increase as the antiviral response comes in action. At the same time, the number of infected machines decreases. Overall, SF networks are less affected by the virus than HM networks for small π.

Another interesting feature is the sharpness of the peak at the maximum number of infected individuals. Indeed, the peak appears smoother when few machines are infected, and sharper when almost all machines are infected. The presence of this peak is important since in real computer epidemics, such a strong peak is not observed [9]. Therefore, it could be an artifact created by the model, which would suggest that the PSIDR model does not appropriately capture real outbreaks. This peak may be due to two different factors:

1. The number of newly infected machines is small when most machines are already infected. Thus, in practice, the birth rate is really small in these cases. Also, if the birth rate itself is small compared to the other rates, this kind of peak may occur even when the number of infected machines is low.

2. The number of machines that go to the curing stage (transit from I to D) follows a binomial law with probability μ. Therefore the number of machines cured in a single timestep is expected to be $D(t+1)-D(t) = I\mu$. As the number of infected machines approaches the network size, I reaches its maximum value, hence the number of newly detected computers in a single time step is the greatest at this time.

In the simulations, the combination of these two factors probably generates this peak. This effect is expected in any finite networks, but should not be present in an infinite network. In a real finite network (like the Internet), it is often observed that worms have a low prevalence [44]. Moreover, typical worms spread a lot faster than they are cured. Therefore, the absence of the sharp peak may be explained by the first factor mentioned above (small number of infected machines and higher birth rate than detection/cure rates).

Finally, for a relatively low π, the number of infected computers is smaller in SF networks than in HM networks. Indeed, it takes longer for the worm to infest the SF network because, in this network, the worm cannot reach all neighbouring nodes from any infected node. However, when the time to initial detection is longer, the worm has enough time to span over all the network, and then the decay in prevalence follows a similar course for HM and SF networks (see two bottom plots). This is due to the fact that the probabilities of detection and of cure are unrelated to the topology.

In general, the PSIDR model seems to conform to the intuistic picture of how real outbreaks occur in technological networks. In addition, it is compatible with existing data about worm prevalence.

5.2.2 Effects of control parameters (π, μ, δ) on costs

Interactions between π and μ

This section concerns the interactions between the response time (π) and detection rate (μ). Figures 5.3 and 5.4 report the values for HM and SF networks respectively.

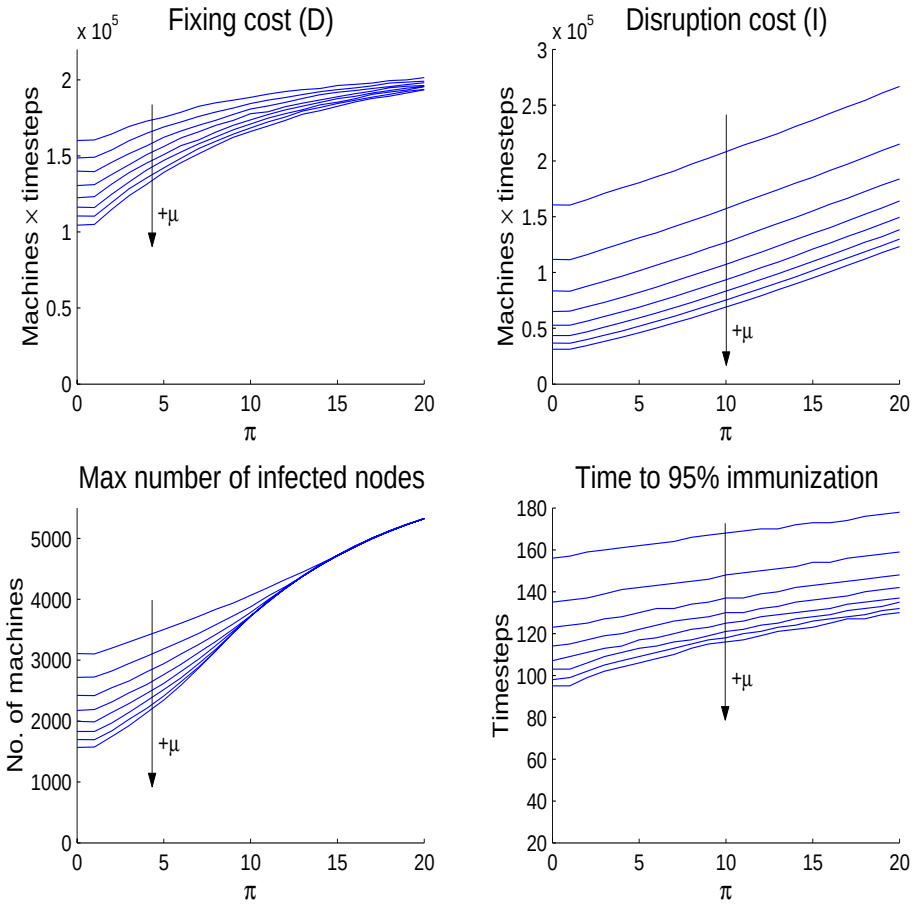

Fig. 5.3: **Costs as a function of π and μ in 6250 nodes HM nets** ($\beta = 0.1$, $\delta = 0.03$) in the intervals $0 \geq \pi \geq 20$ and $0.03 \geq \mu \geq 0.10$ (from top to bottom). Interactions are clearly present. In order to benefit from the effect of increasing μ, the response time π has to be kept as low as possible.

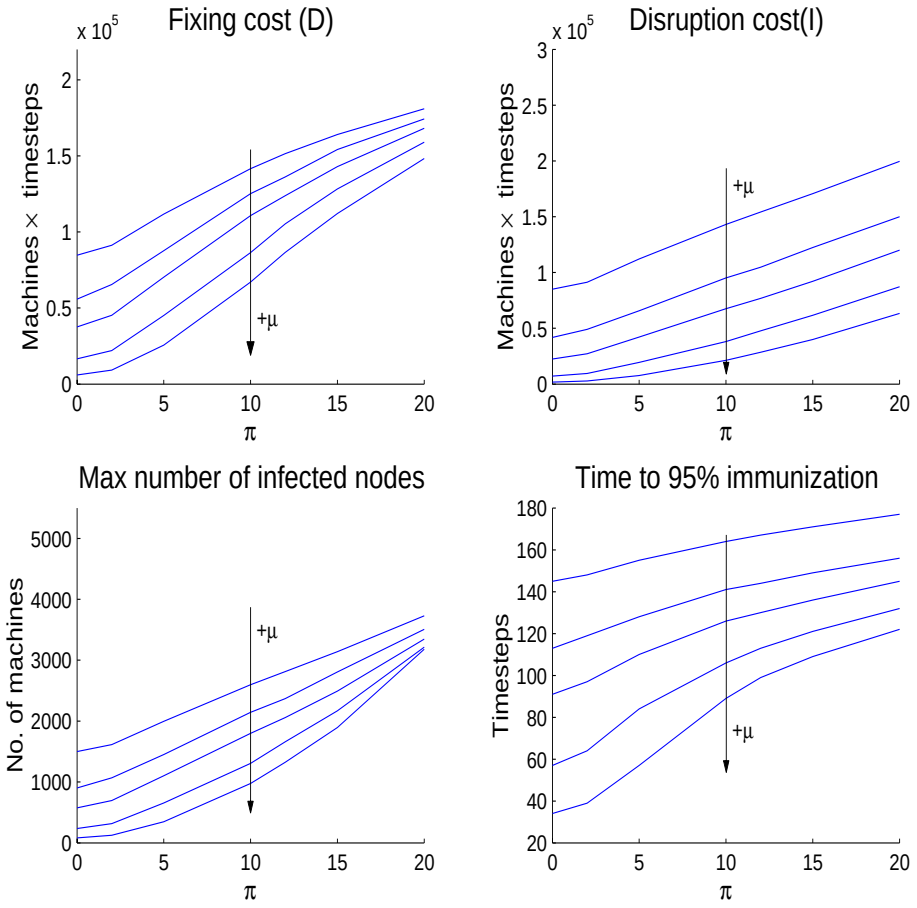

Fig. 5.4: **Costs as a function of π and μ in 6250 nodes SF networks** ($\beta = 0.1$, $\delta = 0.03$) for $\pi = 0$, 2, 5, 10, 12, 15, and, 20 and $\mu = 0.03$, 0.04, 0.05, 0.07 and 0.10 (from top to bottom). A similar situation than in HM networks is observed, although costs tend to be smaller.

The main effect on costs is due to the response time (π). That is, decreasing response time decreases all costs.

Increasing the detection rate will also decrease all costs. A slight increase in the detection rate μ (say from $\mu = 0.03$ to $\mu = 0.04$) greatly improves the disruption cost and immunization time. The effect of μ is present on the fixing costs and maximum prevalence only for low values of π in HM networks. The same interaction between μ and π is observed for SF networks, although to a lesser extent. This is because HM networks cannot benefit from immunization when most nodes are already infected.

Applying these results to the real world, it is important to make the antivirus signature available as soon as possible after the first few worm infections. Also, costs will be improved by distributing faster the antivirus signature.

Interactions between π and δ

This section concerns the interactions between the response time (π) and cure rate (δ). Figures 5.5 and 5.6 report the values for HM and SF networks respectively.

In general, decreasing the response time (π) decreases all the costs. The cure rate (δ) does not influence either the disruption or the maximum prevalence. However, an increase in cure rate reduces the fixing cost and time to immunization. This is intrinsically coded in the PSIDR model: the cure rate is not meant to control transitions *to* or *from* the I state: it only transits units from the D state to the R state. The two factors interact nonlinearly for the fixing cost. That is, decreasing the response time has a greater effect on the fixing cost for low values of δ. Again, keeping the response time as low as possible is a priority, but a large increase in cure rate (say from $\delta = 0.03$ to $\delta = 0.10$) will greatly improve some of the costs.

The maximum prevalence is higher in HM than in SF networks. The opposite is also true for disruption. It indicates that the infection is more scattered in time for SF networks but is never as acute as it is in HM networks. A similar phenomenon has previously been observed in the case of the SIS model [42]. It reflects the fact that infection is inherently slow in SF networks. This phenomenon would probably be attenuated by using a higher detection rate, because susceptible machines would then transit to the removed state instead of waiting to be infected by the worm. In return, this explanation also accounts for the longer immunization time observed in SF networks than in HM networks (see bottom right plot in Figures 5.5 and 5.6).

Applying these results to the real world, it is again important to make the worm signature available as soon as possible. If the fixing cost and immunization time are the main concerns, curing infected computers faster will drastically improve those costs.

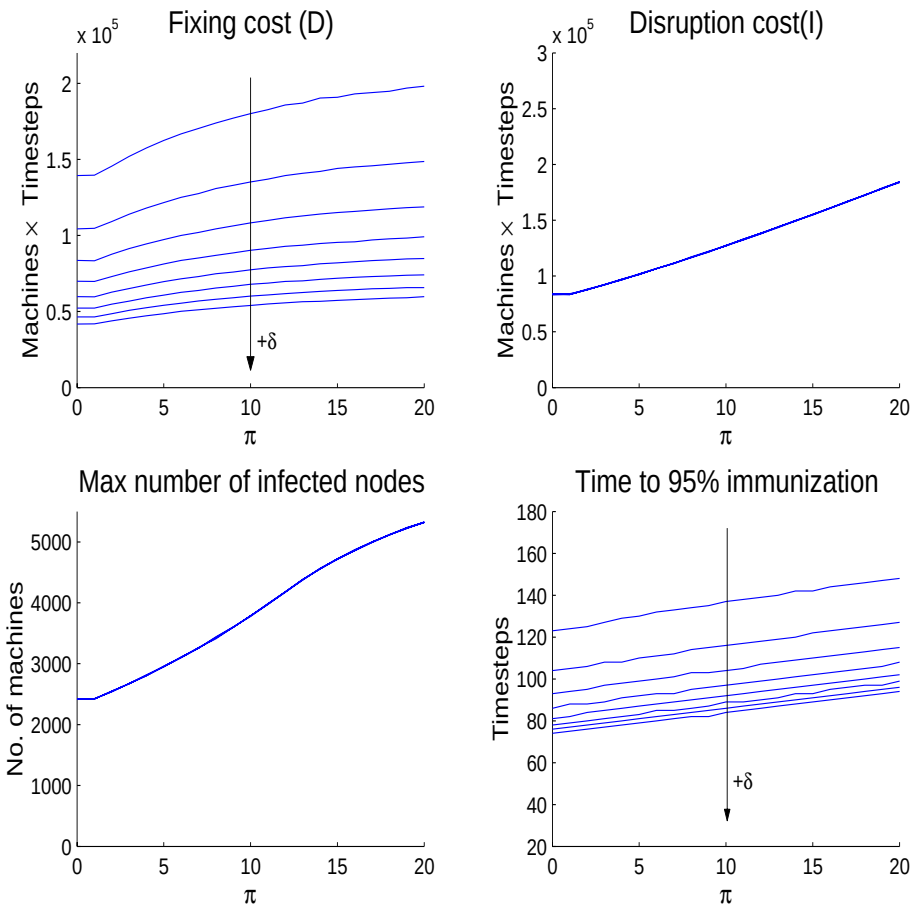

Fig. 5.5: **Costs as a function of π and δ in 6250 nodes HM networks** ($\beta = 0.1$, $\mu = 0.05$) for the intervals $0 \geq \pi \geq 20$ and $0.03 \geq \delta \geq 0.10$ (from top to bottom).The cure rate does not influence the disruption and maximum prevalence as expected by the model. Response time and cure rate interact on the other costs.

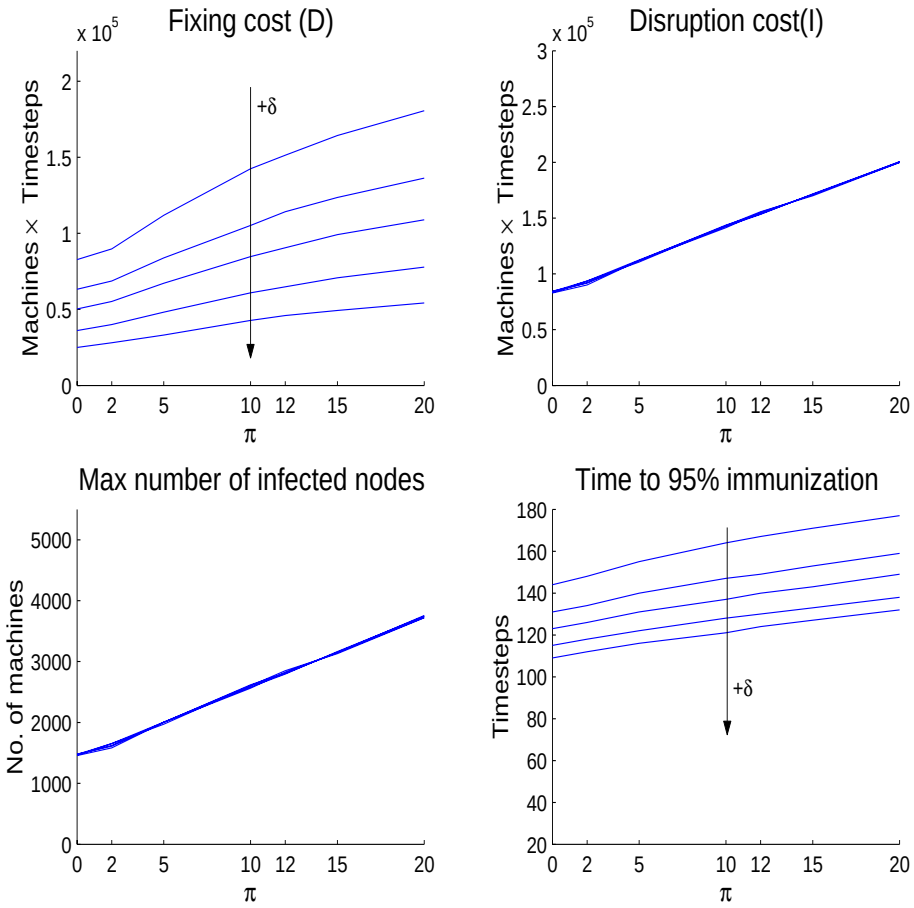

Fig. 5.6: **Costs as a function of response time π and cure rate δ in 6250 nodes SF networks** ($\beta = 0.1$, $\mu = 0.03$) for $\pi = 0$, 2, 5, 10, 12, 15, and 20 and $\mu = 0.03$, 0.04, 0.05, 0.07 and, 0.10. The cure rate does not influence the disruption and maximum prevalence as expected by the model. Except for the time to immunization and disruption, the costs are lower in SF than in HM networks. This is due to the fact that, in SF networks, the outbreak is more scattered in time but never as acute as in HM networks.

Interactions between μ and δ

This section concerns the interactions between the detection (μ) and cure (δ) rates. Figures 5.7 and 5.8 report the values for HM and SF networks respectively.

In general, increasing the cure rate δ decreases the fixing cost, and increasing the detection rate μ reduces the disruption cost. Both factors seem to have a lowering effect on the time to immunization. This suggests that efficient improvement in security strategies cannot be delimited to only one mechanism, but have to be made in both aspects of the response (detection and cure), in order to reduce most costs. Different topologies seem to respond in different ways to the various parameters. In HM networks, the detection rate has no effect at all on the fixing cost, but this is due to the response time $\pi = 20$ used in the simulation, which cancels the effect of μ in HM networks (see Figure 5.3). Since SF networks take more time to become infected, they can still benefit from an increase in the detection rate (most nodes are available for direct immunization). This explanation also accounts for the observed maximum number of infected nodes, which does not change as a function of μ in HM networks, but does in SF networks. Finally, the disruption may be accounted for by the same explanation. However, it does not explain the really big improvement observed when increasing μ from 0.03 to 0.05 in SF networks for the disruption cost. This considerable improvement might be attributed to the increased probability of immunizing highly connected nodes: it has been shown elsewhere that, once those nodes are immunized, virus prevalence drops really quickly [15].

Applying these results to the real world, it is important to act on both the antivirus distribution and the cure to have an effect on all costs. While a faster cure will reduce the fixing cost, a faster antivirus distribution will reduce the disruption. By increasing the speed of either factor, the network will be immunized more quickly. In respect to network topology, it is important to immunize highly connected nodes.

5.2.3 Spreading rate and virus throttling

Virus throttling refers to a strategy whereby monitoring network connections allows for a reduction in speed of propagation of the infectious worm. Here, for simplicity, the main effect of throttling is taken to be a reduction in the birth rate to a low value ($\beta = 0.05$). The benefits of virus throttling (slowing) are likely to be greater for fast worms. This hypothesis is tested by simulating various birth rates (in the range $0.05 < \beta < 0.14$) on HM and SF networks. The course of the outbreak over time is shown in Figure 5.9.

SF networks appear to be more affected than HM networks by throttling. Figure 5.10 clearly illustrates the effects on costs.

Slowing the propagation has the general effect of reducing the different costs. The figure shows that throttling is most effective for faster worms, and in particular for SF networks. The time to immunization and the fixing costs are the most improved by the slowing strategy. This is probably due to a single phenomenon, that is, more computers are available for immunization when the

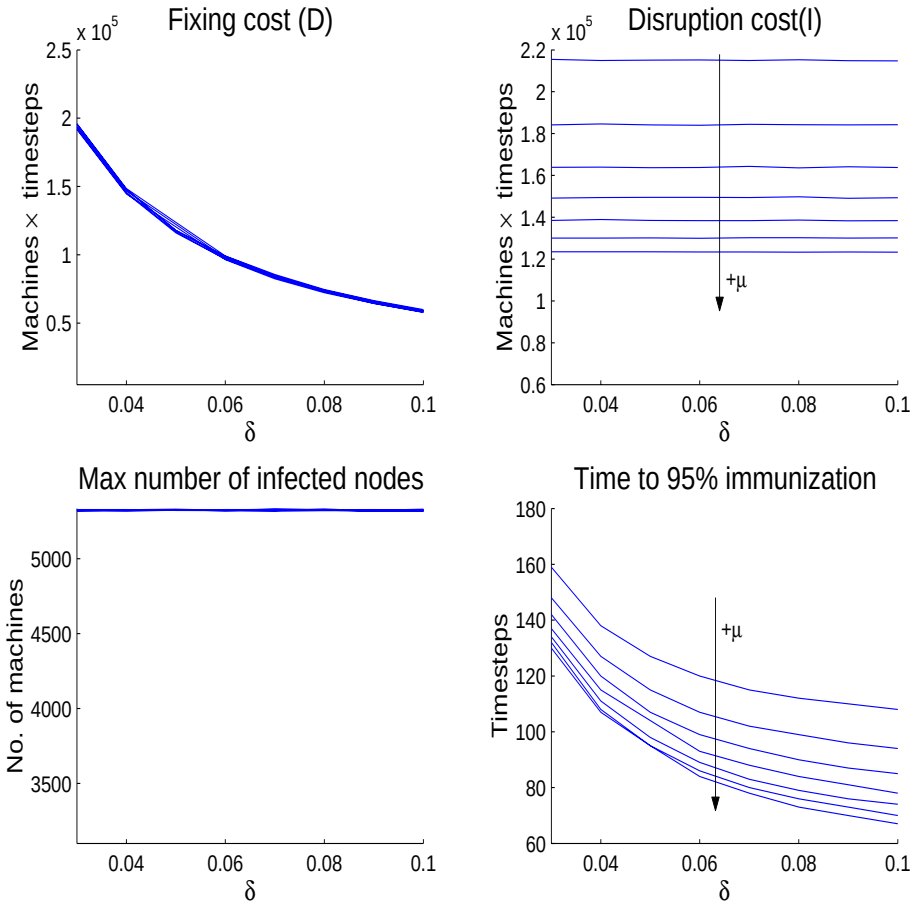

Fig. 5.7: **Costs as a function of the μ and δ in 6250 nodes HM networks** ($\beta = 0.1$, $\pi = 20$) for the interval $0.03 \geq \delta \geq 0.10$ and $0.03 \geq \mu \geq 0.10$ (from top to bottom). Different parameters act similarly on different costs, but interact on the time to immunization. Both factors do not influence the maximum number of infected machines.

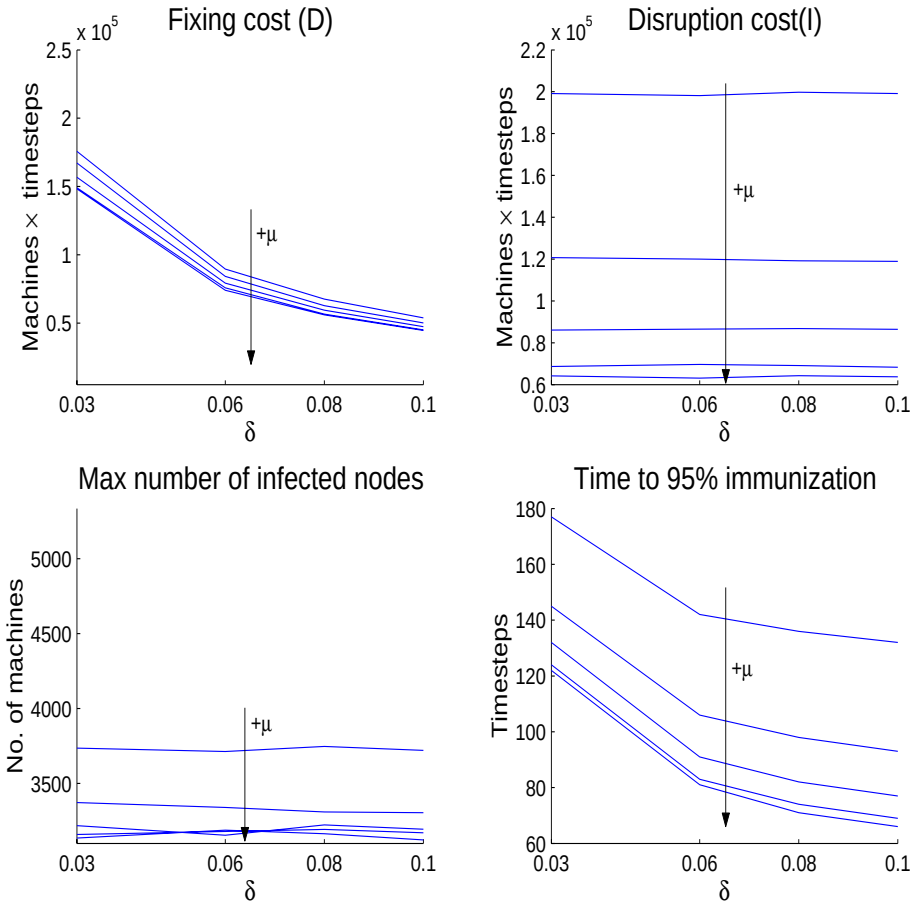

Fig. 5.8: **Costs as a function of the μ and δ in 6250 nodes SF networks** ($\beta = 0.1$, $\pi = 20$) for $\delta = 0.03, 0.06, 0.08$ *and* 0.10 and $\mu = 0.03, 0.05, 0.07, 0.09$ *and* 0.10 (top to bottom). Contrary to what is observed in HM networks, the detection rate μ influences the maximum number of infected machines.

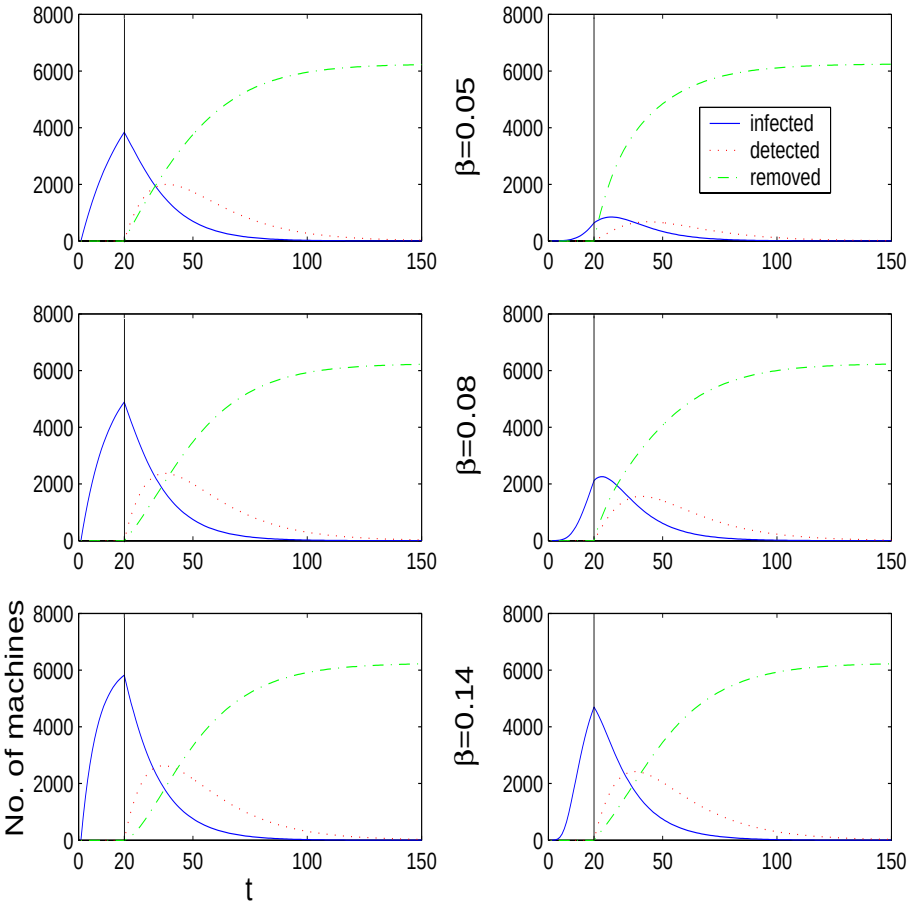

Fig. 5.9: **The PSIDR model as a function of spreading rate** in HM (left) and SF (right) networks ($\pi = 20$,$\mu = 0.07$, $\delta = 0.05$). Reducing the spreading rate (β) affects the evolution of the outbreak. The effect of reducing the worm's speed is clearer in SF networks.

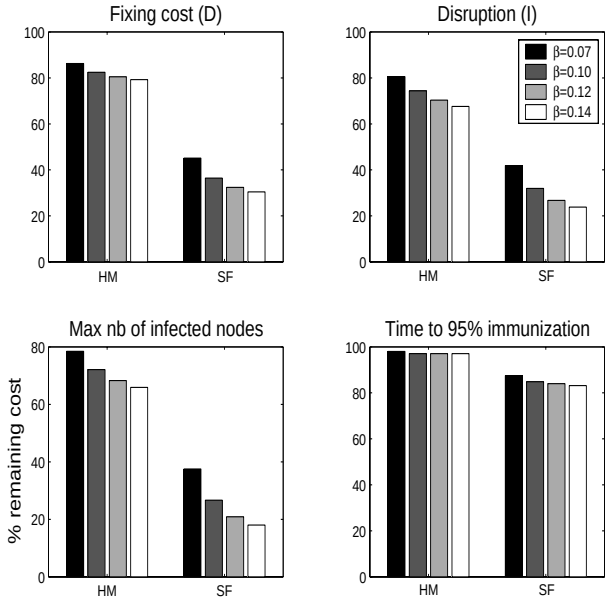

Fig. 5.10: **Effect of virus throttling on costs** in HM and SF networks of 6250 nodes ($\delta = 0.05$, $\mu = 0.07$, $\pi = 20$). Bars represent the cost after throttling as a percentage of the original cost (without throttling). In general, virus throttling is most effective for SF networks and for higher β.

spreading rate is slow (they are immunized before they are infected). Although the benefits appear limited in HM networks, it must not be forgotten that the value chosen for the birth rate after throttling ($\beta = 0.05$) is rather conservative (high). The actual rate is likely to be smaller than the one assumed here, and improvements in cost are likely to be more considerable.

5.2.4 Comparison with the SIR model

The PSIDR model is perhaps the most complex epidemiological model that has attempted to capture technological epidemics. This beneficial increase in complexity makes it more appropriate to describe the behaviour of real outbreaks. However, it also complicates the analysis of the model due to interactions between different parameters. In this section, the effect of straight transitions $S \rightarrow R$ is studied separately by comparing the PSIDR model to the SIR model. It is shown that the PSIDR model can account for the general low prevalence of worms over the Internet, which makes it a more powerful model than the SIR model. One way to compare both models is to set $\pi = 0$ in the PSIDR model. If the number of infected individuals is the only quantity of interest, the D state can be assimilated to the R state. If the cure rate in the SIR model is set to equal the μ parameter (detection and cure) in the PSIDR, the difference observed between the SIR and PSIDR models should therefore reflect the process of transiting some nodes directly from S to R. Immunization should have the effect of reducing the total number of infected individuals by limiting the number of susceptibles. There doesn't seem to be any example of such kind of immunization in the relevant literature: some work [45] used *static* immunization (performed before the simulation) while here *dynamic* immunization is the one at test.

The shape of the SIR model appears similar to what was observed in [34]. The SIR model appears to overestimate the number of infected machines, even when detection in the PSIDR model is low. The difference is particularly large in the case of scale free networks. The general low prevalence of computer viruses over the Internet was attributed to its scale-free nature in [44]. However, this explanation was flawed since the scale free structure of the Internet is valid only for the router maps. Indeed, worms spread not on the router topology, but really from one address to another, as in IP and email networks. Figure 5.11 shows that the low prevalence could be attributed to the existence of direct transitions from S to R.

5.3 Summary of the results

5.3.1 Model validity and improvements over previous models

The general behaviour of the mode conforms to the intuitive picture of worm outbreaks and to existing data on worm prevalence. Moreover, the low prevalence of worms on the Internet can be easily explained by the PSIDR model (as

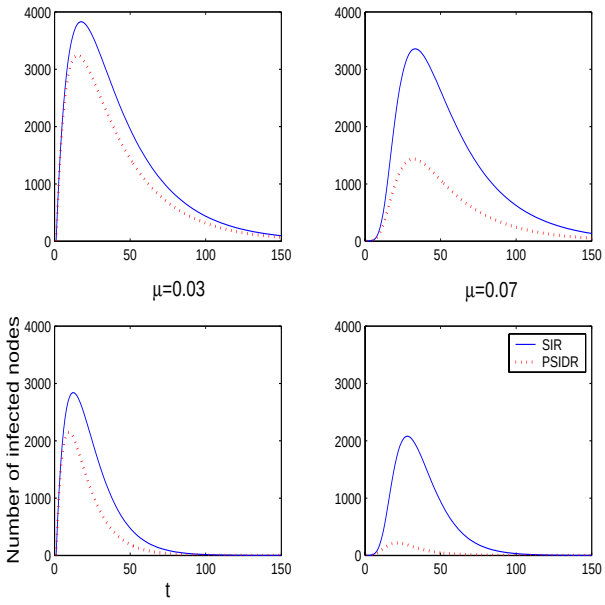

Fig. 5.11: **The SIR model compared to the PSIDR model** in HM (left) and SF (right) networks of 6250 nodes ($\beta = 0.1$). Direct transitions from S to R result in a lower prevalence, especially in SF networks.

opposed to the SIR model). Costs defined by the PSIDR model, especially the fixing cost, allow us to define the best strategies to fight worms.

5.3.2 Best control strategies

In general, it will be preferable to keep the response time (π) as low as possible. This implies that the first instances of a worm must be detected really quickly, and the worm signature made available in a very short time. Automated security systems are advisable since their speed of action in this phase is a lot greater than manual labour.

If the fixing cost is the main concern, augmenting the cure rate (δ) will reduce it considerably. Again, as of now, curing is mostly manual. An automated curing process would be very helpful to reduce this cost.

If the disruption cost is more important, then the detection rate (μ) will have to be augmented. That is, the antivirus will have to be distributed more quickly. Trying to immunize highly connected computers may also help in bringing down prevalence.

The time to immunization is influenced by all control factors (π, μ and δ) since it captures the evolution of the quantity of machines in the removed (immune) state. Therefore, it is affected by what happens in all previous states (Susceptible, Infected and Detected). Any improvement in control strategies will thus have some effect on the time to immunization.

In addition to automated control systems, the effect of virus throttling is a promising avenue for computer security. The combined effects of throttling with various control strategies deserves consideration.

6. GENERAL DISCUSSION

Recent worms pose a major threat to computer networks, and as such they have to be understood in order to reduce the costs associated with them. In this dissertation, an epidemiological model (the PSIDR model) was proposed to account for the spread of computer worms in technological networks.

The PSIDR model includes new features that have a significant impact on the outcome of worm outbreaks. First, the existence of direct immunization (some machines are immunized before they are infected) generally has the effect of lowering worm prevalence. Second, the length of the *Pre-response* phase (an initial phase when the antivirus is not available) is crucial to the various costs and to the effect of various other control parameters. Finally, the existence of a *detected* state (when the machine is in the process of being cured), in addition to influencing the system's dynamics, allows us to measure the cost related to the cure of infected computers.

Most technological networks display either a homogeneous topology or a scale free topology. Therefore, evaluation of the new model was based on those topologies. Results can thus be extended to most technological networks. The properties of the model were evaluated mostly numerically.

The PSIDR model underwent an extensive series of validation tests. The tests revealed that the basic dynamics of the model were not corrupted by simulation artifacts.

Three types of questions were asked about the model:

First, does the model offer a realistic account of real outbreaks? To answer this question, empirical data about computer prevalence was compared against the global behaviour of the model. It was found that the exponential increase as well as the general evolution of prevalence was compatible with the model.

Second, is the increase of complexity in the model beneficial, or are other simpler models already sufficient? To answer this, the PSIDR model was compared to the SIR model. It was shown that the PSIDR model could account for the general low prevalence of computer worms over the Internet, a fact not accounted for by earlier models.

Moreover, the different costs defined on the model allow us to determine the best antiviral strategy. It was shown that the best practice is to make the antiviral signature available as soon as possible. Distributing the signature and curing computers more quickly also decreases costs although each strategy does so in a different way. The use of automatic antiviral systems was also advocated as a general mean to reduce the reponse time and increase the distribution and cure rates. Taking into account the topology of the network may also help in

making the antiviral signature distribution effective.

Finally, is virus throttling an efficient strategy to reduce costs? It was shown that slowing worms could significantly reduce costs especially in SF networks. Note that simulations reported here are conservative (slowing was simulated as having a limited effect) and may not illustrate the full potential of this strategy.

6.1 Future Directions

Here is a list of interesting directions that deserve consideration:

- *Different network sizes* Comparisons with larger networks are helpful to disantangle finite-size effects and properties of the different topologies.
- *Antivirus spreading* One idea advanced by Kephart is to propagate the virus signature in the same way that the virus propagates in the network [24]. The signature would be injected to the computer that started the infection, and then would be left by itself to propagate to neighbouring computers. This strategy would have the advantage of attacking the outbreak at its core.
- *Analytical predictions* Properties of the network in the *Response* phase could also be studied analytically, especially the possible existence of an epidemic threshold as a function of the various parameters.

6.2 Conclusion

As noted in [6], models can be either complex or simple. Complex models have the advantage of offering realistic test cases, and to allow for more accurate predictions. However, simpler models may lead to important insights difficult to get with a complex model.

In the case of computer epidemics, it is possible to build experiments on real worms (eg. [49]), although it can be difficult to understand the system's dynamics. On the other hand, there is a growing body of literature focusing on models such as the SIS, SIR and SEIR models. The PSIDR model shows that more complex models can easily be built and analyzed in details to confer a better characterization of real epidemics. In return, results on the PSIDR model give a better understanding of the mechanisms that will lead to an efficient control of worm attacks.

APPENDIX

A. RELATED WORK

This appendix starts with a review of epidemiological models. Results about epidemic models in simple networks are discussed in the second section. The third section is concerned with the same models but in *scale-free* (SF) networks. The appendix concludes with a brief summary.

A.1 Epidemiological models

Epidemiological models are useful in that they capture essential properties of the spread of diseases in a simplifed way. From such models it is possible to derive important information such as, for example, the maximum prevalence (fraction of infected individuals) of an infection. One popular assumption underlying most epidemiological models is the so-called *homogeneous mixing* hypothesis [26]. That is, every individual has a non-zero probability of directly transmitting the disease to every other individual. In the language of graph theory, this is exemplified by the class of fully connected graphs [25] (see Figure A.1). In a homogeneous graph, an individual is represented by a vertex, and a physical contact between two individuals is displayed as an edge. Each vertex is connected to all other vertices.

The simplest epidemiological model is the so-called SIS model, where S stands for *Susceptible* and I for *Infected*. An individual goes from the S state to the I state at a rate β if at least one of its neighbours is infected (that is, individuals in the I state can infect their neighbours, they are *infectious*). The individual then goes back to the S state at a rate δ, which corresponds to the individual being cured but not immune to a further infection. The parameters β and δ are referred to as the *birth rate* and the *death rate* (or *cure rate*) respectively. In the world of computers the SIS model states that users that recently restored their infected machine are not more alert to computer viruses than before the attack. The equation governing the spread of a virus for the SIS model in a homogeneous network is given by:

$$\frac{d\rho}{dt} = \beta\rho(1-\rho) - \delta\rho$$

where $\rho = \frac{I}{S+I}$ represents the fraction of infected individuals (prevalence). The first term expresses the increase in the number of infected individuals, while the second term stands for its decrease. The solution of this differential equation yields the prevalence at time t:

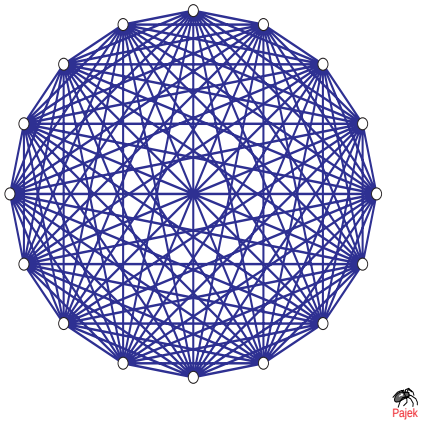

Fig. A.1: **Fully connected graph with 16 vertices.** Each individual (vertex) can infect each of the other 15 individuals through a direct physical contact (edge). All drawings in this thesis were performed using Pajek [5].

$$\rho(t) = \frac{\rho_0(1-\lambda)}{\rho_0 + (1-\lambda-\rho_0)\exp\left(-(\beta-\delta)t\right)}$$

where $\lambda = \frac{\beta}{\delta}$.

As $t \to \infty$, the value $\rho(t)$ converges to $(1-\lambda)$ iff $\beta > \delta$. However, if $\beta < \delta$, the fraction of infected individuals converges to zero. This is one of the fundamental results of classical epidemiology, namely, the existence of an epidemic threshold $\lambda = \lambda_c$ for the occurence of an endemic state[1]. Interestingly, the same value that determines the occurence of an outbreak defines the final prevalence of the disease.

One problem with the SIS model is the assumption that individuals go back from the I state to the S state. A model that attempts to capture the immunization of individuals after infection is the SIR model, where R stands for *Removed*. In this model, individuals leave the state I at a fixed rate as is the case for the SIS model. However, instead of going back to the I state, individuals go to the R state, where they are considered immune to a future infection or simply dead, which is a more common scenario in biological epidemics [29]. In the world of computers, the R conditions corresponds roughly to the immunization of computers by the introduction of antivirus software. The system of differential equations representing the SIR model is shown below, where S and R represent the fraction of *susceptible* and *removed* individuals respectively.

[1] An endemic state is defined as an equilibrium state where prevalence is greater than 0.

$$\begin{aligned} \frac{dS}{dt} &= -\beta IS \\ \frac{dI}{dt} &= \beta IS - \delta I \\ \frac{dR}{dt} &= \delta I \end{aligned}$$

As for the SIS model, an epidemic threshold $\lambda = \frac{\beta}{\delta} = \lambda_c$ does exist and determines the presence of an endemic state. The SIR model has been applied to analyze real outbreaks and has shown good agreement with existing data [35]. One natural consequence of the SIR model is that the virus prevalence decreases due to the gradual reduction in the number of susceptible units throughout the epidemic [29].

A third epidemiological model is the SEIR model, which takes into account the fact that many viruses have an incubation period. During the incubation period, the disease does not inflict any visible damages to the host, but the latter is nonetheless contagious. In this model, E stands for *Exposed* and represents the incubation period. The SEIR model has been successfully applied to the analysis of foot-and-mouth disease outbreaks that happened recently in the UK and Taiwan [11][2].

It is possible to translate models such as the SIS, SIR or SEIR into the language of percolation theory. Newman and coworkers [40, 37, 38, 32] have used percolation models to study epidemics on different kinds of complex networks. Their work has consistently pointed to the existence of a *percolation threshold*, similar to the epidemic threshold, that depends on network configuration.

Finally, epidemics have been analyzed using a *discrete Markov model*, and shown again to have an epidemiological threshold [30]. Although this approach seems to be very recent (May 2002) and more work needs to be done on complex networks, it may prove to be useful by making analytical derivations simpler than in other models (*mean field* or percolation).

A.2 Network topologies and epidemiology

A.2.1 Simple networks

It was shown empirically that the topology of many computer networks largely diverges from a homogeneous graph (see [1] for a review). The first serious attempt to understand the interaction between topology and computer epidemics in a complex graph model is due to Jeffrey Kephart [26]. In this original work, it was demonstrated both analytically and empirically that different networks show different patterns of epidemics.

[2] However, the acronym SLIR was used instead of SEIR, where L stands for *latency*. Appart from this literary distinction, the model remains the same.

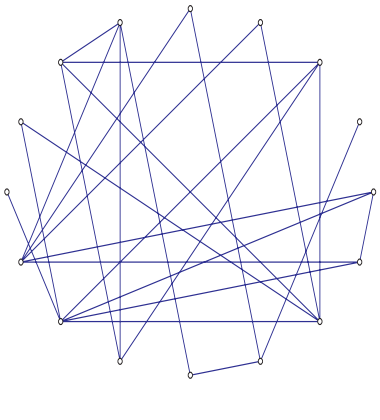

Fig. A.2: **ER random graph with average degree 3.**

Kephart investigated the properties of the Erdös-Rényi (ER) graph (Figure A.2) whose degree distribution follows a Poisson distribution. If a specific ER graph is denoted by G, and its set of vertices and edges by V and E respectively, then $G = (V, E)$ has a maximum of $\frac{|V|(|V|-1)}{2}$ possible undirected edges. Vertices are connected together according to a probability p. Thus, the probability that vertex i has k neighbours is given by the binomial $P(k_i = k) = \binom{|V|-1}{k} p^k (1-p)^{|V|-1-k}$. Therefore, on average there should be $\lambda_k =\mid V \mid P(k)$ vertices with k neighbours. When $\mid V \mid \rightarrow \infty$ the degree distribution is well approximated by the Poisson distribution $P(k) = \frac{e^{-\lambda}\lambda}{k!}$ where λ is given by $\mid V \mid P(k)$ [1].

In a random graph, the probability of having a vertex connected to another *spatially contiguous* node is the same as that of having the vertex connected to a far node: there is no local neighbourhood in a ER model.

If a number of nodes are initially infected and the propagation simulated according to the SIS model, the outcome (ie. whether there will be an endemic state or not) can be accurately predicted by the solution to the usual differential equations of the SIS model. This is because there are no irregularities in the network, which makes it suitable to a similar treatment to that for a fully connected graph. It is critical to note that the epidemic threshold in a ER graph tends to increase as the average degree of the vertices is lowered. From a percolation point of view, the probability of connection of any two vertices p is strongly related to the critical threshold p_c.

Kephart also investigated properties of the hierarchical model [26] (see Figure A.3), which is a loop-free and cycle-free graph [19]. The advantage of the tree model over previous ones is that it intrinsically encapsulates the tendency of users — represented by nodes — to share programs with only a limited num-

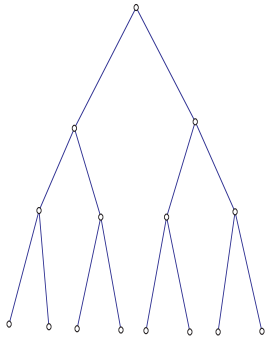

Fig. A.3: **Tree graph with $l = 3$ levels.** A communitity is composed of the nodes of a same subtree.

ber of users. That is, a user will often share programs with users in the same subtree, creating a small community. This ultimately results in the probability of infectious spreading among users in a single community to be higher than the probability of spreading between users pertaining to different communities. The epidemic threshold is also higher in the case of the tree model compared to both the ER and the homogenous graphs [26].

Finally, Kephart investigated the properties of a regular lattice (Figure A.4 and Figure A.5), or *spatial model* since nodes can be thought of as being distributed spatially. Nodes in a lattice are only connected in local neighbourhoods. The spread of a disease in a spatial model turns up to be quadratic [26].

These examples clearly demonstrate that a network's topology influences the spread of virus and argue for the study of their interaction. Although tree models may capture part of the structure of real networks, *small world* models are considered an even better approximation.

A.2.2 Small worlds *and* scale-free *networks*

It has been noticed some time ago that any pair of persons can be connected through a rather small number of intermediate acquaintances [39]. In terms of graph theory this phenomenon reflects the small value of the *average path length* of many social nets, comparatively to the number of individuals. Networks that display this property have been called *small worlds* [39]. In some sense, since the average path length of the ER graph increases proportionally to the logarithm of its size, it is also a small world [1]. However, the ER graph is not a small world strictly speaking since another defining characteristic of small worlds is their *clustering coefficient* which is of lesser value than in lattice models, but

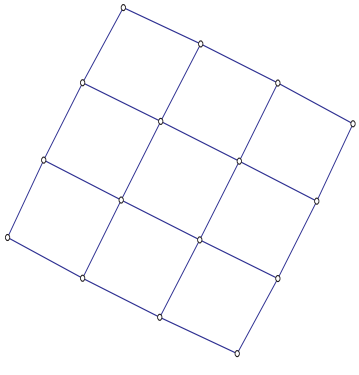

Fig. A.4: **Two dimensional regular lattice.** Virus propagation in the lattice model is quadratic due to the fact that nodes can only infect their direct neighbour.

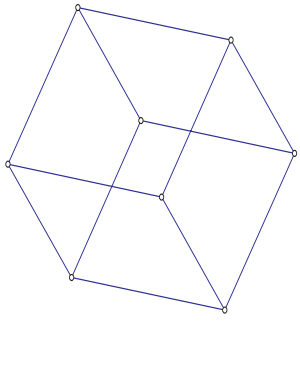

Fig. A.5: **Three dimensional lattice.** In general, it is possible to construct a d-dimensional lattice.

greater than in an ER graph. The definition of the clustering coefficient is given in the next paragraph.

If k_i is the number of neighbours of vertex i, then there are $\frac{k_i(k_i-1)}{2}$ possible undirected connections between them. Let E_i be the actual number of such connections, then the clustering coefficient can be formally defined as the average over all i of $C_i = \frac{2E_i}{k_i(k_i-1)}$. This coefficient takes its smallest value in ER models, and its largest value in lattice models. Small worlds are networks with a short average path length and a clustering coefficient between that of regular lattices (maximum clustering) and random graphs (minimum clustering). The clustering coefficient indicates if neighbouring nodes tend to connect to the same neighbours.

According to one classification, there are three classes of small world networks: *broad scale* networks, *single scale* networks, and *scale free* networks [2]. The first type is characterised by a degree distribution (the distribution of the number of neighbours) that is a mixture of a power law and an exponential distributions. An example of such graph can be found in [37]. Graphs of the second type follow a sharp distribution, which means that nodes with a very high connectivity are not present. One such example is the Watz and Strogatz (WS) model [52]. Finally, scale free networks (SFNs) strictly follow a power law distribution, implying that nodes with a large connectivity have a statistically significant probability of being present. The most popular example of a SFN is the Barabási and Albert (BA) model [3]. The second and third types of graphs have been most extensively studied in the past few years.

Small world networks, especially of the third type, appear to be pervasive in a wide range of phenomena, including the graph of actor/scientist collaborations [2], sexual contacts [28], metabolic networks [21], the WWW [4], the internet router map [1] and the email network [18]. Although it is not certain if mobile phone networks[3] are scale-free, one paper shows that the topology of phone calls in one day follows a power-law (often taken as indication of a SF graph) [50]. More research on phone networks could be done to determine the exact topology of the graph.

The prototype SFN is represented by the Barabasi-Albert (BA) model [3]. The BA algorithm for the creation of a SFN incorporates the assumptions of *incremental growth* and *preferential connectivity* [17]. Incremental growth signifies that a network increases in size by the natural inclusion of additional nodes, which is obvious in cases like the Internet or the WWW. Preferential connectivity implies that newly added nodes tend to get connected with existing nodes that already have a high number of connections, which is also true in the Internet architecture. The BA model is thus considered a reasonably good model of the Internet router and WWW networks [4] as well as of many other real networks [1]. The exact algorithm to create BA networks is reported in Appendix B.

[3] Phone networks are maps of *who has the phone number of who*. Individuals are nodes, and edges link nodes when two individuals exchange phone numbers.

Epidemics in small world networks

A number of studies have investigated the properties of the WS network in simulated viral epidemics [11, 32, 38, 34, 42, 27, 47]. These studies have generally shown that there exists an epidemic threshold. In fact, the behaviour of the WS model can be approximated using equations for the homogeneous model due to the small fluctuations in the connectivity distribution [34, 42].

Most epidemiological studies on the BA network have been conducted by Pastor-Satorras and co-workers [44, 41, 42, 43]. Results obtained for the BA network with the SIS model indicate the surprising absence of an epidemic threshold [44] in both infinite size [29] and finite size networks [43]. Analytically, the threshold λ_c for the occurence of an outbreak in a BA network is found to follow $\frac{1}{logN}$, indicating that as N grows infinite, the threshold tends to zero. Also, the survivability in time of an epidemic is largely augmented as N grows large. Pastor-Satorras and al. [42] attribute this to the decrease in probability that all nodes be cured at the same time.

In practical terms, it means that an outbreak will occur, whatever the value of λ the spreading rate λ. A second result however, states that the final prevalence of the outbreak will be lower than what would be observed in a homogeneous network. In fact, the final prevalence in the BA network is given by

$$\rho \simeq 2 \exp\left(\frac{-1}{m\lambda}\right)$$

This equation shows that, whenever λ is greater than 0, a fraction of the susceptible individuals will be infected, showing the absence of the epidemic threshold. For small values of λ, the prevalence will be small [44].

Lloyd and May [29] explain this result using the *basic reproductive number* $R_0 = \kappa_0(1 + (CV)^2)$ familiar to epidemiologists. In this equation, CV represents roughly the variation in degree across vertices. κ_0 stands for β $*$ *duration of infection* $*$ *average connectivity*. In a homogeneous net, $CV \equiv 0$, and $R_0 \equiv \kappa_0$. However, in a SFN, CV is potentially infinite, which means that the basic reproductive number is very large, ultimately lowering the epidemic threshold [29]. Also, the same authors argue that the smooth increase in prevalence for small values of λ are due to the fast saturation of infected individuals in highly affected subgraphs of the net. Further increases in prevalence are due to infection over less active subregions of the network [29].

Results for the BA networks were extended to the GSFN model (*Generalised Scale-Free Network*)[42]. In short, the GSFN model is a BA model but with an arbitrary power (determined by a parameter γ) in the distribution. When $0 < \gamma < 1$, as in the BA network, the epidemic threshold is absent. However, for increasing γ, the threshold reappears suddenly, reflecting the fact that the connectivity distribution becomes bounded as in WS, ER and other models [42]. Importantly, similar analytical results were reported for the SIR model [34]. In epidemiological terms, the high value of γ annihilates the CV term. Although not studied here, the GSFN is important because it represents the entire class of SF networks, not just a particular instantiation such as the BA network.

The relation between the epidemic threshold and the bound on the connectivity distribution has also been studied by Newman [37] on a *broad scale* network having the degree distibution $P(k) \sim Ck^{-\alpha}exp\frac{-k}{\kappa}$ using percolation theory and generating functions. The purpose of the constant C is to ensure that the distribution is normalised to one. This particular distribution can approximate a SFN when $\kappa \to \infty$, otherwise the distribution is bounded and an epidemic threshold appears again. Moreover, as shown in [37], when the value of α is up to a certain level, the connectivity distribution becomes bounded and the epidemic threshold reappears. However, when $\alpha < 3$ the epidemic threshold disappears.

In brief, virus propagation in complex networks seems to be highly reliant on the existence of highly connected nodes.

A.3 The control of outbreaks

Surprisingly, very little attention has been paid to the control of epidemic outbreaks in epidemiological models of computer viruses. Control can be exerciced in two ways, prevention and cure. Prevention deals specifically with the adequate immunization of computers inside a network. One paper shows that the immunization of nodes with high connectivity is likely to reduce the chances of an outbreak [46]. Conversely, it has been demonstrated in [15] that a similar strategy applied during the curing phase could dramatically reduce the pervasiveness of an outbreak. It is worth noting that, in this study, the individual curing of computers was modelled as having a certain probability of success, which makes the simulation more realistic since cures are not always successful.

One possible avenue for future research is to look at dynamical properties of the immunization process. That is, the effect of immunization might be different whether it is made *statically* (before the epidemic event) or *dynamically* (during the epidemic event). This question is addressed in chapter 5.

A.4 Other related work

A recent topic of interest is the effect of *correlation* in complex networks. A correlated complex network is one where the probably for a node to be connected to a neighbour of degree k' depends on its own degree k. Results of epidemic spreading on correlated complex networks suggest that they have different characteristics [7].

Recent work [6] suggests that chaos can be induced by simply adding noise to various parameters. In the same spirit, it is shown in [51] that adding noise in the transmissibility of each edge leads to an increased transition region in the neighbourhood of the epidemic threshold: the probability of reaching an endemic state does not grow extremely quick near the epidemic threshold but rather follows a smooth curve, as in real phenomenons. This highlights to the importance of noise in models.

A.5 Summary

So far, useful biological models have been used to understand computer epidemics. Any study about virus propagation needs to mention what kind of network (ex: ER, WS, SF, etc.) it uses, and which epidemiological model (SIS, SIR, SEIR, etc.).

Indeed, network topology influences the outcome of an outbreak. Scale-free graphs are thought to reflect some real networks like the email network or the World Wide Web. Kephart argued that homogeneous graphs could not model patterns of communications between the users: indeed each user tends to exchange emails (or programs, etc.) frequently only with a subgroup of people, not with the entire network. Therefore, a random graph is a better model. However, with the advent of new worms that can propagate using corporate mass-mailing lists (reaching potentially all the network), this argument is no longer valid. Homogeneous networks can still be taken as models of some real phenomena. Thus, in this dissertation, simulations will be conducted on both homogeneous and scale-free networks, which means that the results will have the potential to be applied to a broad range of technological networks.

Different models like the SIS, SIR and SEIR could be used to model different aspects of virus propagation. However, they necessarily leave out some details.

B. SCALE-FREE NETWORKS

This section contains a description of the algorithm used to create BA networks, it is described in more details in [3].

The algorithm devised by Barabási and Alberts creates networks with a distribution following $P(k) = k^{-2.9\pm0.1}$. It works in the following way:

1. Start with m_0 number of nodes, not linked by any edge.
2. For t iterations, add one node, and link it with m previously existing nodes. The neighbours are chosen according to their connection probability: the more connections a node already has, the more likely it is to be selected as neighbour for the newly added node.

What this algorithm does not mention is how to connect the $m_0 + 1^{th}$ node: the m_0 first vertices have no neighbours assigned at this time, thus their probability of being selected is zero. The algorithm here works by assigning the neighbours of the $m_0 + 1^{th}$ node at random among the m_0 first nodes. Note that this would not solve the case where $m_0 > m$: here $m_0 \equiv m$ in all networks.

In fact, the distribution of the BA network can be extended to the *Generalised Scale Free Network* $P(k) = (1+\gamma)m^{1+\gamma}k^{-2-\gamma}$ where $\gamma > 0$. The purpose of using the GSFN instead of the BA is that it allows for the study of SFN with different values for γ. GSFNs are not investiguated here.

It is noteworthy that a similar algorithm has been proposed long ago by Simon, although in this earlier model the newly added edges were not necessarily connected to the newly added node [8, 17]. Both models can generate a similar degree distribution [8].

At the end of the creation process, the networks contains $t + m_0$ nodes; therefore, the size of the network can be controlled by modifying the t parameter. The average degree in a network created with this procedure is equal to $2m$, and the minimum degree in large networks is m.

Using this recipe, scale free networks of 6250, 12500, 25000, 100000, 500000 and 1000000 nodes were created. The slope of the degree distribution, the average and minimum degrees were checked against the expected values to ensure that the networks are adequate. Although not all sizes were used in the virus simulations, the large sizes help to make sure that the algorithm is performed correctly since large networks contain less noise due to lesser finite size effects.

The distribution in Figure B.1 clearly shows the existence of a few highly connected nodes. The slope of the distribution is approximately $\lambda = 3$.

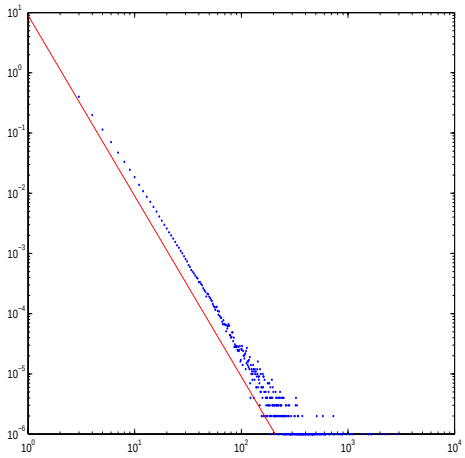

Fig. B.1: **Degree distribution of a BA network with 1 000 000 vertices.** The X-axis stands for the degree of the node, the Y-axis, for the frequency. The full line shows the theoretical slope of 3.

A sample network is displayed in Figure B.2, where *small world* characteristics can easily be observed:

1. The existence of a small number of highly connected nodes.
2. A relatively small average distance from any two nodes. This feature can be appreciated by selecting any two nodes and noting that few hops are necessary to link the two nodes. Exact calculations can be conducted to show that the diameter in BA networks is smaller than in lattice models with the same number of nodes.

If m is set to 3, in the limit of large networks, all vertices will have at least 3 neighbours. In small networks however, it is possible that the m_0 first vertices be connected with less than m neighbours. At the extreme, a vertex could be completely isolated from the rest of the nodes. This situation never occured in the present networks. A sample network where $m = 3$ is displayed in Figure B.3. The structure displayed does not really look like a real network because real networks are unlikely to have a minimum degree of 3 (and a mode of 3). This argument holds for any arbitrary value of m (except perhaps $m = 1$). It thus comes as a surprise that most simulations of epidemics on scale free networks were conducted on the BA networks with $m = 3$.

In all simulations involving the BA models, the network is represented using its adjacency matrix encoded as a sparse matrix in Matlab. Since the network is undirected, the matrix is symmetrical. Moreover, since no reccurent connections are allowed, the diagonal is composed uniquely of zeros.

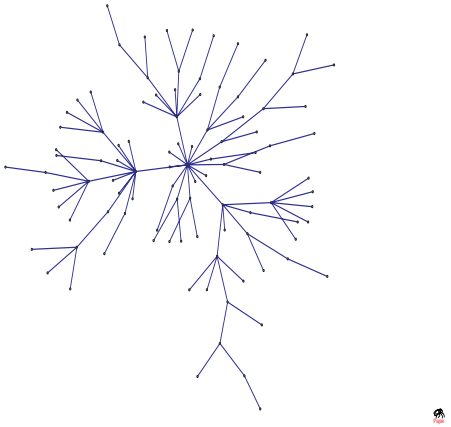

Fig. B.2: **Barabasi-Albert model with $m = m_0 = 1$.** The graph is plotted using the algorithm defined in [22].

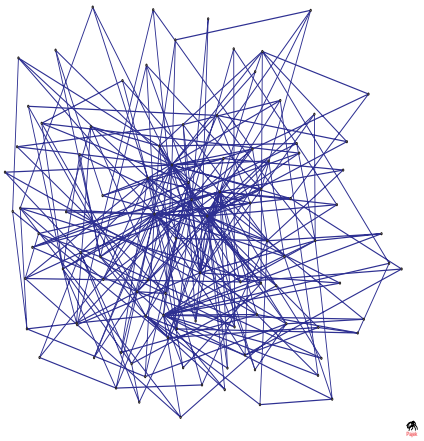

Fig. B.3: **Barabasi-Albert model with $m = m_0 = 3$.**

C. SURVIVAL PROBABILITY IN SIS MODEL AND BA NETWORKS

The data concerning the survival probability reported in [44] could not be replicated well enough to be considered as an exact replication. Although it is of little interest to get acquainted with the SIS model or for the PSIDR model, the discrepant results are reported here.

C.1 Method

BA Networks of $N = 6.25 \times 10^3$, $N = 1.25 \times 10^4$, $N = 2.5 \times 10^4$ and $N = 5 \times 10^5$ nodes are used in the original paper. At the beginning of each simulation run, 1 node is infected at random and the propagation performed as in the SIS model. Again, simulations are done for 100 iterations and repeated 100 times from different starting configurations. In these simulations, $\beta = 0.065$ and $\delta = 1$. The survival probability is calculated at each iteration: it is given by the ratio of the number of trials where there still is at least one infected node, over the total number of trials. The survival probability is plotted against time and as a function of network size.

The same parameters are used here, except for the network sizes, where networks of $N = 1 \times 10^5$ replace the larger $N = 5 \times 10^5$ network.

C.2 Results

The problem is that the survival probability drops too quickly to zero in the current results (as shown in Figure C.1).

One possible source of error could lie in the discretization of time, different time steps could result in different results. The simulations were ran using various timeslices (1, 0.1 and 0.01) on 10 networks of 6250 nodes, for 100 iterations and 1000 repetitions. There is no significant improvement in survival probability experiments whatever the timeslice (see Figure C.2).

In fact it looks like the prevalence and survival probability are even *lower* than expected when the system is simulated with a small timeslice. One possibility could be that the nodes were not updated all at once in [44]: only one node is updated at each time step. This would result in a slower decrease, but the same end state. However it is mentioned that *parallel* updating of nodes is used [44], which strongly suggests that all nodes were updated at each time step.

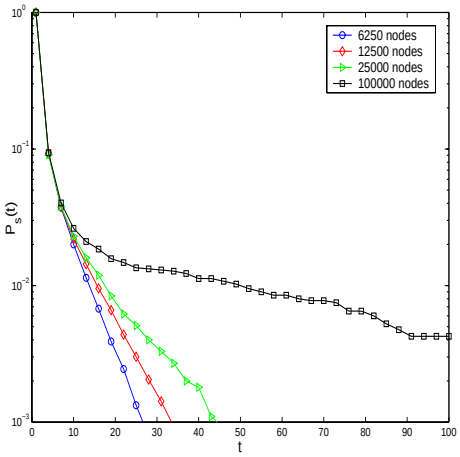

Fig. C.1: **Survival Probability in SF nets as a function of network size ($\beta = 0.065$).** Only 400 repetitions were made for the 100000 nodes network. This figure is meant to replicate fig.3a in [44] except for the data on 100000 and 500000 nodes networks.

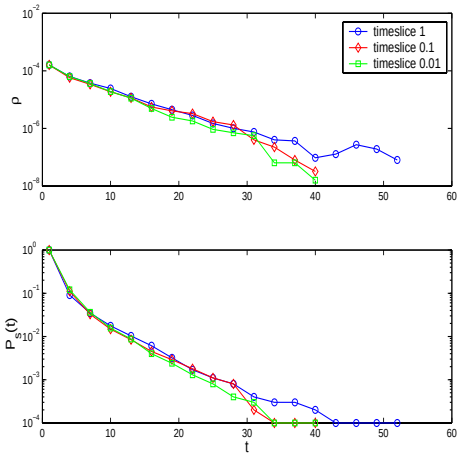

Fig. C.2: **Effect of timeslice on prevalence** (ρ) and survival probability ($P_s(t)$) in SF networks of 6250 nodes. Timeslices do not make current data closer to the original data.

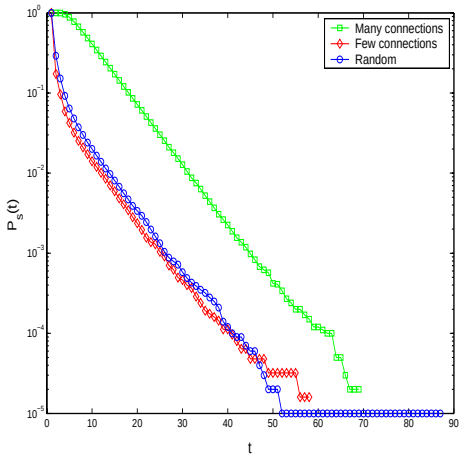

Fig. C.3: **Survival probability as a function of the degree of the originally infected node.** The survival probability drops too quickly even for highly connected starting nodes.

One reason why the survival probability drops so quickly might as well be that the starting nodes were not infected independently of their degree. For exemple, starting the epidemic with the most highly connected node might allow the outbreak to survive long enough to support Pastor-Satorras' data. This idea was tested in 10 networks of 6250 nodes, each simulated for 100 iterations and 10000 trials.

The degree of the initially infected node seems to influence the initial spread. In the first few iterations, epidemics started on a highly connected node (actually the maximum degree) all survive which makes the survival probability of 1. The survival probability then decreases at a rate similar independently of the degree of the initially infected node, ultimately reaching the absorbing state of 0 prevalence. In the random starting condition, the 682^{th} trial led to a nonzero survival probability until iteration 87. It could be that the outbreak is worst when the starting node is not necessarily the most connected, but the one in the best location (has many highly connected neighbours). But it could also be due to a random effect: the outbreak was just particularly strong in one repetition. To clarify the issue a subsequent simulation was run on the same networks with 100000 repetitions. This long-lasting effect was not observed, which suggests that it was just a random effect. In brief, the starting node can affect the survival probability in the short term, but the system always tends to the same state, whatever the starting condition, which is a good indication that the system simulated is not chaotic[1]. In Pastor-Satorras' data, the survival probability is even higher than what is observed here with the starting condition being the mostly connected node.

[1] This is true in the sense that one hallmark of chaos is sensitivity to initial conditions [23].

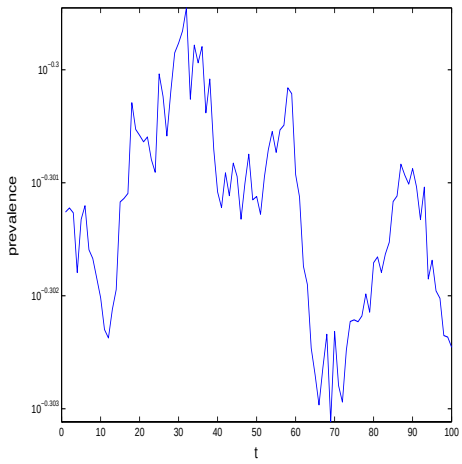

Fig. C.4: **Prevalence in a SF network of 6250 nodes.** The infection probability is proportional to the number of infected neighbours. Noisy oscillations can be observed in the data.

In [44] it is understood that a node becomes infected with rate β *if it is connected to one or more infected nodes.* This suggests, counter-intuitively, that the probability of infection is not proportional to the number of infected neighbours. All simulations performed here are based on this assumption. If the infection probability followed the number of infected neighbours, the outbreak could potentially be more persistent. To discard the possibility that Pastor-Satorras used an infection rate proportional to the number of infected neighbours, simulations using proportional infection were performed on 10 networks of 6250 nodes, for 100 iterations and 100 repetitions.

The survival probability is *one* at each time step, so the prevalence is reported to give more details. The system behaves in a totally different way than what is observed in Pastor-Satorras' work. This strongly suggests that his data was not obtained with the infection probability proportional to the number of infected neighbours. Interestingly, the prevalence data shows some kind of noisy periodicity. Periodicity was also observed in [27], where infection was proportional to the number of infected neighbours. The data in the latter paper is less noisy, which could be due to many differences in the parameters that were used (even the topology was different). Nonetheless, the periodicity may be an important characteristics of these systems, somewhat independent of topology.

Finally, there is a possibility that the survival probability is not calculated properly, which would explain the fact that only the prevalence data is compatible.

In conclusion, the results concerning the survival probability do not agree enough with the original data to say that they replicate them. Different aspects were investiguated and seemed to corroborate the existing theoretical and ex-

perimental (in the case of periodicity) literature. An exact replication of the data in [44] would be greatly facilitated by comparison of the source codes.

D. SOURCE CODE

The choice of Matlab as the programming language for the various simulations rather shortened the source code. Morevover, the source codes that implement the various epidemiological models varies very little between the models. This is because most epidemiological models have a similar structure. Instead of writing a single function that would encompass all the models, separate functions were written to optimize performance (by reducing the number of *if* statements etc.).

D.1 SF networks

The function *netsp2* is used to create SF networks of *NbIterations* vertices. The variables m, m0 can determine the density of the network. The output variables *pK* and *vout* are the degree distribution and the adjacency matrix respectively. The function *bin_sear* performs binary search to accelerate the algorithm.

D.1.1 Netsp2

```
function [pK,vout] = netsp2( NbIterations )
%Creates networks with sparse adjacency matrices

%Initialization: create the first few vertices
m = 3;
m0 = 3;

myOnes = ones(m,1);

i = [1:m0];
j = [(m0+1) * ones(1,m0)];

mk = NbIterations;
nk = NbIterations;

s = ones(m0,1);

%The total nb of connections is 2 * m * NbIterations
Net = sparse(i, j , s , mk , nk , NbIterations * m);

clear i j mk nk s;


pK = zeros(NbIterations,1);      %Vector contains the degree of nodes

%Connects the m0+1'th vertex
pK(1:m0) = myOnes;
pK(m0+1) = m;

%buffers
ni = zeros(1,m);
```

```
tni = zeros(1,m);

$Incremental growth of the network
for i = 2: NbIterations-m0

    %Selection of the m neighbours for the new node, using roulette wheel style method

    ProbC = pK / (2 * m * (i-1));

    tmpID = rand(m,1);

    Go = 0;
    tmpSum = cumsum(ProbC);

    ni(1) = bin_sear(tmpSum,tmpID(1));
    ni(2) = bin_sear(tmpSum,tmpID(2));
    ni(3) = bin_sear(tmpSum,tmpID(3));
    %ni(4) = bin_sear(tmpSum,tmpID(4));
    %ni(5) = bin_sear(tmpSum,tmpID(5));

    %Ensure that a node has not been selected twice
    while ~Go

        dNi = diff(sort(ni),1);
        cNi = find(dNi == 0); %Finds the zero elements

        if(cNi)

            lcNi = length(cNi);
            tni = zeros(lcNi,1);

            tmpID = rand(lcNi,1);

            for j=1:lcNi
                tni(j) = bin_sear(tmpSum,tmpID(j));
            end

            ni([cNi]) = [tni];

        else
            Go=1;
        end

    end

    %Adds the ones to the right places in the adjacency and degree matrix
    Net([ni],i+m0) = myOnes;
    pK([ni]) = pK([ni]) + myOnes;
    pK(i+m0) = pK(i+m0) + m;

end

vout = Net; %Outputs the net
```

D.1.2 Bin_sear

Performs binary search and return the index of the element just below the threshold. This function is a modification of the original function written by Eran O. Ofek (1994).

```
function i=bin_sear(x,v);
% This version is a modification of the original source code written by
% Eran O. Ofek(September 1994)
%-------------------------------------------------------------------------
N = length(x);

if N~=0,
   i1 = 1;
   i2 = N;
```

```
    while x(i1)~=v,
        if i1==i2,
            break;
        elseif (i2 - i1)==1,
            if (x(i2) + x(i1))<2.*v,
                i1 = i2;
            end
            break;
        end
        mdi = round((i1 + i2)./2);

        mdv = x(mdi);

        if v>mdv,



            i1 = mdi;
        elseif v<mdv,
            i2 = mdi;

            if x(mdi-1) < v
                i = mdi;
                return
            end
        else
            i1 = mdi;

            break;
        end
    end
    i = i1;
else
    i=0;
end
```

D.2 SIS model

D.2.1 Updatehm

The function *updatehm* performs the update of the SIS model on HM networks.

```
function varargout = updatehm(varargin)
%varargout = UPDATEHM(NBVERTICES, TIMESLICE, ITERATIONS, CURERATE, BIRTH, INIT)

%setup
nbVertices = varargin{1};
timeslice = varargin{2};
Iterations = varargin{3};
curerate = varargin{4};
birth = varargin{5};
init = varargin{6}; %'half' or 'unique'

state = sparse(nbVertices,Iterations);
period = ceil(1/timeslice);
tmpstate = sparse(nbVertices,period+1);

delta = 1-curerate*timeslice;
beta = birth*timeslice; %Prob of infection

%Initialisation
if strcmp('half',init)
    tmpstate(:,1) = rand(nbVertices,1)<0.5;  %Initially, half the nodes are infected for graph2
else
    tmpstate(ceil(rand(1)*nbVertices),1) = 1;   %Initially, infect only one node
end
```

```
%Process
state(:,1) = tmpstate(:,1); %Useful if no timeslice

for i=2:Iterations
   for qq = 1 : period
      tmp = tmpstate(:,qq) & rand(nbVertices,1)<delta;
      tmpstate(:,qq+1) = tmp | (rand(nbVertices,1)<beta);
   end

   state(:,i) = tmpstate(:,end);
   tmpstate(:,1) = tmpstate(:,end);

end

varargout{1} = sum(state)'; %Nb of infected nodes
varargout{2} = any(state)'; %Survival probability
```

D.2.2 Batchupdatehm

This script is executed to perform large-scale simulations with many repetitions and on many HM networks.

```
function time = batchupdatehm(NETS,NETSIZE,REPS,ITERS,TIMESLICE,BETA,DELTA,START,FILENAME)
%BATCHUPDATE3
%   TIME = batchupdate3(NETS,NETSIZE,REPS,ITERS,TIMESLICE,BETA,DELTA,START,FILENAME);
%   The parameters are :
%   NETS: number of networks
%   NETSIZE: Size of each network
%   REPS: the number of repetitions
%   ITERS: the number of timesteps
%   TIMESLICE: the timeslice between each timestep (ex: 0.1, 0.001)
%   BETA, DELTA: the birth and death rates
%   START: the starting config (half infected or unique infected)
%   FILENAME: file to store the results

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%   Setup

nets = NETS;        %The number of networks simulated
netsize = NETSIZE;
reps = REPS;      %The number of repetitions of the simulation
iters = ITERS;    %The number of iterations
netype = ['net',num2str(netsize)]; %The size of network simulated
timeslice = TIMESLICE;%change in update2 also
topology = 'homogeneous';
init = START;
beta = BETA;
delta = DELTA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totrelt = sparse(iters,1);
totprev = sparse(iters,1);

t = sparse(iters,1);
d = sparse(iters,1);
denom = ones(iters,1) * (reps * nets);

start = cputime;

net_load = ['Net',num2str(netsize)];

for j=1: nets

    disp('Simulating Network...');disp(j);

    for i =1:reps
        eval(['[d,t] = updatehm(' num2str(netsize) ',' num2str(timeslice) ',\
 num2str(iters) ',' num2str(delta) ',' num2str(beta) ',init);']);
```

```
        totrelt = totrelt + t;
        totprev = totprev + d;
    end

    clear net pK; %Prepare for next network

end

totrelt = totrelt ./ denom;
totprev = totprev ./ denom;

totprev = totprev ./ (netsize * ones(iters,1));

parameters = struct('beta',beta,'delta',delta,'nbnetworks',nets,'netsize',netsize,\
'nbiterations',iters,'nbreplications',reps,'topology',topology,'timeslice',\
timeslice,'startconfig',init);

save(FILENAME,'totprev','totrelt','parameters');

time = cputime-start;
```

D.2.3 Update3

The function *update3* is used to perform the update of the SIS model. It is basically a slightly more complex version of the function *updatehm*, except that it is meant to perform the update on SF networks. There are two types of updates included in it: one is for updates with no timeslices, the other is for updates with timeslices. The second can mimick exactly the first type as long as the parameter timeslice is set to 0. However, the first type of update is faster to compute. Presently, the second type is being used (the % sign disables the first type). The output argument is a variable-length list, so more quantities can be output depending on the interest.

The code for simulations where the number of infected neighbours influences the probability of infection is not included here: it involves only minor changes.

```
function varargout = update3(varargin)
%varargout = UPDATE3(MAT,TIMESLICE,ITERATIONS,CURERATE,BIRTH,INIT)

%setup
mat = varargin{1};
nbVertices = length(mat(1,:));
timeslice = varargin{2};
Iterations = varargin{3};
curerate = varargin{4};
birth = varargin{5};
init = varargin{6}; %'half' or 'unique'

state = sparse(nbVertices,Iterations);
period = ceil(1/timeslice);
tmpstate = sparse(nbVertices,period+1);

delta = 1-curerate*timeslice;
beta = birth*timeslice; %Prob of infection

%Initialisation
if strcmp('half',init)
    tmpstate(:,1) = rand(nbVertices,1)<0.5;  %Initially, infect only one node
else
    tmpstate(ceil(rand(1)*nbVertices),1) = 1;   %Initially, infect only one node
end

%Process
state(:,1) = tmpstate(:,1); %Useful if no timeslice

for i=2:(Iterations)

   %backwards update
   % state(:,i+1) = (mat*state(:,i)) & (rand(nbVertices,1)<beta) ; %without timeslice
   % Backward update (with timeslice)
```

```
    for qq = 1 : period
        tmp = tmpstate(:,qq) & rand(nbVertices,1)<delta;
        tmpstate(:,qq+1) = tmp | ((mat*tmpstate(:,qq)) & rand(nbVertices,1)<beta);
    end

    state(:,i) = tmpstate(:,end);
    tmpstate(:,1) = tmpstate(:,end);

end

varargout{1} = sum(state)'; %Nb of infected nodes
varargout{2} = any(state)'; %Survival probability
```

D.2.4 Batchupdate3

This script is executed to perform large-scale simulations with many repetitions and on many networks. It loads the mat-files containing the networks and calls the *update3* function. It saves the output as a mat-file and includes a structure where the values of all parameters are stored for usability. Other scripts named *Batchupdate4* and *Batchupdate5* are used to simulate SF networks with m=4 and m=5 respectively.

```
function time = batchupdate3(NETS,NETSIZE,REPS,ITERS,TIMESLICE,BETA,DELTA,START,FILENAME)
%BATCHUPDATE3
%    TIME = batchupdate3(NETS,NETSIZE,REPS,ITERS,TIMESLICE,BETA,DELTA,START,FILENAME);
%    The parameters are :
%    NETS: number of networks
%    NETSIZE: Size of each network
%    REPS: the number of repetitions
%    ITERS: the number of timesteps
%    TIMESLICE: the timeslice between each timestep (ex: 0.1, 0.001)
%    BETA, DELTA: the birth and death rates
%    START: the starting config (half infected or unique infected)
%    FILENAME: file to store the results

%The suffixes of the names of each network (Stored as a Mat-File)
networks = [
     'A3'
     'B3'
     'C3'
     'D3'
     'E3'
     'F3'
     'G3'
     'H3'
     'I3'
     'J3'
     'K3'
     'L3'
     'M3'
     'N3'
     'O3'
     'P3'
     'Q3'
     'R3'
     'S3'
     'T3'
     'U3'
     'V3'
     'W3'
     'X3'
     'Y3'
];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%    Setup

nets = NETS;        %The number of networks simulated
netsize = NETSIZE;
reps = REPS;      %The number of repetitions of the simulation
iters = ITERS;    %The number of iterations
netype = ['net',num2str(netsize)]; %The size of network simulated
timeslice = TIMESLICE;%change in update2 also

init = START;
beta = BETA;
delta = DELTA;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

totrelt = sparse(iters,1);
totprev = sparse(iters,1);

t = sparse(iters,1);
d = sparse(iters,1);
denom = ones(iters,1) * (reps * nets);
```

```
start = cputime;          %Takes the starting time

net_load = ['Net',num2str(netsize)];

for j=1:nets

    disp('Loading Network...');disp(networks(j));

    eval(['cd networks\m=3\' net_load]);

    eval(['load ' net_load networks(j,:) ';']);

    cd ..\..\.. %Changes directory to use the fct update3

    disp('Parallel Updating');

    net = net|net';

    for i =1:reps
        eval(['[d,t] = update3(net,' num2str(timeslice) ',' num2str(iters) ',' \
          num2str(delta) ',' num2str(beta) ',init);']); % If we are not interested in S
        totrelt = totrelt + t;
        totprev = totprev + d;
    end

    clear net pK; %Prepare for next network

end

totrelt = totrelt ./ denom;
totprev = totprev ./ denom;

totprev = totprev ./ (netsize * ones(iters,1));

%The parameters structure contains the values of simulation parameters
parameters = struct('beta',beta,'delta',delta,'nbnetworks',nets,'netsize',netsize,\
                    'nbiterations',iters,'nbreplications',reps,'topology',topology,\
                    'timeslice',timeslice,'startconfig',init);

cd simulations\m=3       %Changes directory to store the results

save(FILENAME,'totprev','totrelt','parameters');          %saves the results as mat-file

cd ..\..

time = cputime-start;
```

D.3 PSIDR simulations

The function *update2* serves to update networks according to the PSIDR model. It can perform updating for both HM and SF networks, depending on the option specified in the argument list. The output is also a variable argument list.

The function *Batchupdate2* executes the repetitions over all networks, where m=3, and calls *update2* to perform the updating. It stores the results and a structure with the parameters used in a mat-file.

D.3.1 Update2

```
function varargout = update(varargin)
%UPDATE2 performs parallel update
%    [Infected,Detected,Removed] = UPDATE2(NET) performs the update on NET for 100 iterations
%    It returns the number of infected, detected and removed individuals at each timestep in
%    a variable length argument list.
%    UPDATE2(NET,IT) performs the update for IT iterations.
%    UPDATE2(NET,IT,ST) performs the update for IT iterations starting with initial state ST.
%    UPDATE2(NET,IT,PI,BETA,MU,DELTA,TIMESLICE,TOPOLOGY)  includes the initial PI period and uses
%    defined parameters. If PI == IT, we only simulate the period preceding the antivirus. TOPOLOGY
%    can take either the value 'homogeneous' or 'scalefree'.

%Setup
mat = varargin{1};
mat = mat | mat';                    %Transposes the matrix since the network is undirected


nbVertices = length(mat(1,:));

timeslice = 1; %default value

%default parameters
beta = 0.2*timeslice;
mu = 0.1*timeslice;
delta = 0.1*timeslice;%1-curerate*timeslice;
pi=0;

initstate = sparse(nbVertices,1);

switch length(varargin)
case 1
```

```
    maxIteration = 100;
    initstate(ceil(rand(1) * nbVertices),1) = 1;
case 2
    maxIteration = varargin{2};
    initstate(ceil(rand(1) * nbVertices),1) = 1;
case 3
    maxIteration = varargin{2};
    initstate(:,1) = varargin{3};
case 8
    maxIteration = varargin{2};
    initstate(ceil(rand(1) * nbVertices),1) = 1;
    timeslice = varargin{7};
    pi = varargin{3};
    beta = varargin{4} * timeslice;
    mu = varargin{5} * timeslice;
    delta = varargin{6} * timeslice;
    topology = varargin{8};
otherwise
    error('Incorrect number of parameters');
end

if maxIteration<pi
    error('The number of iterations cannot be smaller than pi');
end

period = 1/timeslice;

state = sparse(nbVertices,maxIteration);          %Stores the state (S, I, etc.) of all nodes at each timestep
tmpstate = sparse(nbVertices,period+1); %this stores the data between timesteps (when slice<1)
tmpstate(:,1) = initstate;

state(:,1) = tmpstate(:,1);

pI(1) = length(find(state(:,1)==1));

buffstate = sparse(nbVertices,1);

%Initial infected node must be transfered in buffstate
startindex = find(tmpstate(:,1));
buffstate(startindex) = 1;

%If a HM network is simulated
if strcmp(topology,'homogeneous')

    %Initial period without DETECTION (S->I)
    for i=1:pi-1

        for qq=1 : period

            %S->I
            idS = find(tmpstate(:,qq)==0);
            if idS ~= []
                buffstate(idS) = (rand(length(idS),1)<beta);    %Use for HM network
            end
            tmpstate(:,qq+1) = buffstate;
        end

        tmpstate(:,1) = buffstate;
        state(:,i+1) = buffstate;

        pI(i+1) = length(find(buffstate==1));

        pD(i+1) = 0;
        pR(i+1) = 0;

    end

    %Period with DETECTION and CURE
    for i=pi:maxIteration-1
        for qq=1 : period

            %S->I
            idS = find(tmpstate(:,qq)==0);
            if idS ~= []
                buffstate(idS) = (rand(length(idS),1)<beta);    %Use for HM network
            end

            %S->R
            idS2 = find( ~(tmpstate(:,qq) | buffstate) );
            if idS2 ~= []
                buffstate(idS2) = 3*(rand(length(idS2),1)<mu);
            end

            %I->D
            idI = find(tmpstate(:,qq)==1);
            if idI ~= []
                buffstate(idI) = buffstate(idI) + (rand(length(idI),1)<mu);
            end

            %D->R
            idD = find(tmpstate(:,qq)==2);
```

```
            if idD ~= []
                buffstate(idD) = buffstate(idD) + (rand(length(idD),1)<delta);
            end

            tmpstate(:,qq+1) = buffstate;

        end

        tmpstate(:,1) = buffstate;
        state(:,i+1) = buffstate;

        pI(i+1) = length(find(buffstate==1));
        pD(i+1) = length(find(buffstate==2));
        pR(i+1) = length(find(buffstate==3));

    end

else     %Simulates a heterogeneous network

    %Initial period without DETECTION (S->I)
    for i=1:pi-1

        for qq=1 : period

            %S->I
            idS = find(tmpstate(:,qq)==0);
            if idS ~= []
                st = (tmpstate(:,qq)==1);   %Use for SF network
                buffstate(idS) = (mat([idS],:)*st) & (rand(length(idS),1)<beta); %for SF network
            end
            tmpstate(:,qq+1) = buffstate;
        end

        tmpstate(:,1) = buffstate;
        state(:,i+1) = buffstate;

        pI(i+1) = length(find(buffstate==1));

        pD(i+1) = 0;
        pR(i+1) = 0;

    end


    %Period with DETECTION and CURE
    for i=pi:maxIteration-1
        for qq=1 : period

            %S->I
            idS = find(tmpstate(:,qq)==0);
            if idS ~= []
                st = (tmpstate(:,qq)==1);   %Use for SF network
                buffstate(idS) = (mat([idS],:)*st) & (rand(length(idS),1)<beta); %for SF network
            end

            %S->R
            idS2 = find( ~(tmpstate(:,qq) | buffstate) );
            if idS2 ~= []
                buffstate(idS2) = 3*(rand(length(idS2),1)<mu);
            end

            %I->D
            idI = find(tmpstate(:,qq)==1);
            if idI ~= []
                buffstate(idI) = buffstate(idI) + (rand(length(idI),1)<mu);
            end

            %D->R
            idD = find(tmpstate(:,qq)==2);
            if idD ~= []
                buffstate(idD) = buffstate(idD) + (rand(length(idD),1)<delta);
            end

            tmpstate(:,qq+1) = buffstate;

        end

        tmpstate(:,1) = buffstate;
        state(:,i+1) = buffstate;

        pI(i+1) = length(find(buffstate==1));
        pD(i+1) = length(find(buffstate==2));
        pR(i+1) = length(find(buffstate==3));

    end

end

%Check the relative prevalences (can be modified)

varargout{1} = pI';      %Nb of infected individuals
```

```
varargout{2} = pD';      %Nb of detected individuals
varargout{3} = pR';      %Nb of removed individuals
```

D.3.2 Batchupdate2

```
function varargout = batchupdate2(NETS,NETSIZE,REPS,ITERS,TOPOLOGY,TIMESLICE,BETA,DELTA,MU,PI,FILENAME);
%BATCHUPDATE2
%    TIME = batchupdate2(NETS,NETSIZE,REPS,ITERS,TOPOLOGY,TIMESLICE,BETA,DELTA,MU,PI,FILENAME);
%    The parameters are :
%    NETS: number of networks
%    NETSIZE: Size of each network
%    REPS: the number of repetitions
%    ITERS: the number of timesteps
%    TOPOLOGY: can take either 'homogeneous' or anything else for a heterogeneous network
%    TIMESLICE: the timeslice between each timestep (ex: 0.1, 0.001)
%    BETA, DELTA, MU: the birth, death and detection rates
%    PI: the number of timesteps before any response is made
%    FILENAME: file to store the results

%Contains the suffix of the names of the networks
networks = [
     'A3'
     'B3'
     'C3'
     'D3'
     'E3'
     'F3'
     'G3'
     'H3'
     'I3'
     'J3'
     'K3'
     'L3'
     'M3'
     'N3'
     'O3'
     'P3'
     'Q3'
     'R3'
     'S3'
     'T3'
     'U3'
     'V3'
     'W3'
     'X3'
     'Y3'
];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%    Setup

nets = NETS;        %The number of networks simulated
netsize = NETSIZE;
reps = REPS;      %The number of repetitions of the simulation
iters = ITERS;    %The number of iterations
netype = ['net',num2str(netsize)]; %The size of network simulated
topology = TOPOLOGY; %can be scalefree or homogeneous
timeslice = TIMESLICE;%change in update2 also

beta = BETA;
delta = DELTA;
mu = MU;
pi = PI;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

start = cputime; %Takes the starting time

dI = zeros(iters,1);
meanI = zeros(iters,1);

denom1 = reps * ones(iters,1);
denom2 = nets * ones(iters,1);

if pi~=iters    %Means we want other info as well since we simulate not only the period<pi
     sumI = zeros(iters,1);
     sumD = zeros(iters,1);
     sumR = zeros(iters,1);

     dD= zeros(iters,1);
     meanD = zeros(iters,1);
     dR = zeros(iters,1);
     meanR = zeros(iters,1);
end

for i=1:nets     %For each network ...

     %Load network
     cd networks
```

```
    if ~strcmp(topology,'homogeneous')
        disp('Loading Network...');disp(networks(i));
        eval(['load ' netype networks(i,:) ';']);
        net = net|net'; %The matrix is symmetric cuz undirected
    else
        net = sparse(netsize,netsize);
        disp(networks(i))
    end

    disp('Parallel Updating...');
    cd ..

    if pi==iters     %Means we only want info relevant to before PI
        for j=1:reps
            %Checks only the virus prevalence
            eval(['[dI] = dI + update2(net,' num2str(iters) ',' num2str(pi) ','\
              num2str(beta) ',' num2str(mu) ',' num2str(delta) ',TIMESLICE,TOPOLOGY);']);
        end

        dI = dI ./ denom1;
        meanI = meanI + dI;
        dI = zeros(iters,1);

    else  %We do full PSIDR
        for j=1:reps
            %Checks all parameters
            eval(['[dI,dD,dR] = update2(net,' num2str(iters) ',' num2str(pi) ','\
              num2str(beta) ',' num2str(mu) ',' num2str(delta) ',TIMESLICE,TOPOLOGY);']);

            sumI = sumI + dI;
            sumD = sumD + dD;
            sumR = sumR + dR;
        end
    end
end

if pi~=iters
    sumI = sumI ./ denom1;
    sumD = sumD ./ denom1;
    sumR = sumR ./ denom1;
end

disp('Analysis...');

lim95  = (19*netsize)/20; %The 95% limit

if pi == iters
    meanI = meanI ./ denom2;

    %Get maximum slope of the I progression for experiments before pi
    slope = diff(meanI);
    maxslope = max(slope);

    time95 = find(meanI>=lim95);

    if time95 ~= []
        time95 = time95(1);
    else
        time95 = 0;
    end
else
    meanI = sumI ./ denom2;
    meanD = sumD ./ denom2;
    meanR = sumR ./ denom2;

    maxI = max(meanI);  %The maximum number of infected computers

    costD = sum(meanD); %The riemann integral of D for cost of fixing computers

    disr = sum(meanI);  %The riemann integral of I for total disruption

    %Here we want the time to 95% removed machines
    time95 = find(meanR>=lim95);

    if time95 ~=[]
        time95 = time95(1);
    else
        time95 = 0;
    end
end

disp('Done');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Saving

cd psidrsims

%Save parameters
parameters = struct('beta',beta,'delta',delta,'mu',mu,'pi',pi,'nbnetworks',nets,\
                        'netsize',netsize,'nbiterations',iters,'nbreplications',reps,\
```

```
                                'topology',topology,'timeslice',timeslice);

if pi==iters
    save(FILENAME,'meanI','slope','maxslope','time95','parameters');
else
    save(FILENAME,'meanI','meanD','meanR','costD','maxI','disr','time95','parameters');
end

cd ..

varargout{1} = (cputime-start); %outputs time taken to simulate
```

D.4 SIR model

As in other simulations, there is a function to perform the updating (*updatesir*) and a function to repeat the updating over all networks (*batchupdatesir*). These functions work in a similarly to previous functions.

D.4.1 Updatesir

```
function varargout = updatesir(varargin)
%UPDATESIR performs parallel update of the SIR model
%    [Infected,Detected,Removed] = UPDATESIR(NET) performs the update on NET for 100 iterations
%    It returns the number of infected, detected and removed individuals at each timestep in
%    a variable length argument list.
%    UPDATESIR(NET,IT) performs the update for IT iterations.
%    UPDATESIR(NET,IT,ST) performs the update for IT iterations starting with initial state ST.
%    UPDATESIR(NET,IT,BETA,DELTA,TIMESLICE,TOPOLOGY) uses ddefined parameters. If PI == IT, we
%    only simulate the period preceding the antivirus. TOPOLOGY can take either the value
%    'homogeneous' or 'scalefree'.

%setup
mat = varargin{1};
nbVertices = length(mat(1,:));
timeslice = 1; %default value

%default parameters
beta = 0.2*timeslice;
mu = 0.1*timeslice;
delta = 0.1*timeslice;%1-curerate*timeslice;

initstate = sparse(nbVertices,1);

switch length(varargin)
case 1
    maxIteration = 100;
    initstate(ceil(rand(1) * nbVertices),1) = 1;
case 2
    maxIteration = varargin{2};
    initstate(ceil(rand(1) * nbVertices),1) = 1;
case 3
    maxIteration = varargin{2};
    initstate(:,1) = varargin{3};
case 6
    maxIteration = varargin{2};
    initstate(ceil(rand(1) * nbVertices),1) = 1;
    timeslice = varargin{5};
    beta = varargin{3} * timeslice;
    delta = varargin{4} * timeslice;
    topology = varargin{6};
otherwise
    error('Incorrect number of parameters');
end

if maxIteration<pi
    error('The number of iterations cannot be smaller than pi');
end

period = 1/timeslice;

state = sparse(nbVertices,maxIteration);
tmpstate = sparse(nbVertices,period+1); %this stores the data between timesteps (when slice<1)
```

```
tmpstate(:,1) = initstate;

state(:,1) = tmpstate(:,1);

pI(1) = length(find(state(:,1)==1));

buffstate = sparse(nbVertices,1);

%Initial infected node must be transfered in buffstate
startindex = find(tmpstate(:,1));
buffstate(startindex) = 1;

%If a HM network is simulated
if strcmp(topology,'homogeneous')

    for i=1:maxIteration-1
        for qq=1 : period

            %S->I
            idS = find(tmpstate(:,qq)==0);
            if idS ~= []
                buffstate(idS) = (rand(length(idS),1)<beta);     %Use for HM network
            end

            %I->R
            idI = find(tmpstate(:,qq)==1);
            if idI ~= []
                buffstate(idI) = buffstate(idI) + (rand(length(idI),1)<delta);
            end

            tmpstate(:,qq+1) = buffstate;

        end

        tmpstate(:,1) = buffstate;
        state(:,i+1) = buffstate;

        pI(i+1) = length(find(buffstate==1));
        pR(i+1) = length(find(buffstate==2));

    end

else    %Simulates a heterogeneous network

    for i=1:maxIteration-1
        for qq=1 : period

            %S->I
            idS = find(tmpstate(:,qq)==0);
            if idS ~= []
                st = (tmpstate(:,qq)==1);   %Use for SF network
                buffstate(idS) = (mat([idS],:)*st) & (rand(length(idS),1)<beta); %for SF network
            end

            %I->R
            idI = find(tmpstate(:,qq)==1);
            if idI ~= []
                buffstate(idI) = buffstate(idI) + (rand(length(idI),1)<delta);
            end

            tmpstate(:,qq+1) = buffstate;

        end

        tmpstate(:,1) = buffstate;
        state(:,i+1) = buffstate;

        pI(i+1) = length(find(buffstate==1));
```

```
        pR(i+1) = length(find(buffstate==2));

    end

end

%Check the relative prevalences

varargout{1} = pI';      %Nb of infected individuals
varargout{2} = pR';      %Nb of removed individuals
```

D.4.2 Batchupdatesir

```
function varargout = batchupdatesir(NETS,NETSIZE,REPS,ITERS,TOPOLOGY,TIMESLICE,BETA,DELTA,FILENAME);
%BATCHUPDATESIR
%   TIME = batchupdatesir(NETS,NETSIZE,REPS,ITERS,TOPOLOGY,TIMESLICE,BETA,DELTA,FILENAME);
%   The parameters are :
%   NETS: number of networks
%   NETSIZE: Size of each network
%   REPS: the number of repetitions
%   ITERS: the number of timesteps
%   TOPOLOGY: can take either 'homogeneous' or anything else for a heterogeneous network
%   TIMESLICE: the timeslice between each timestep (ex: 0.1, 0.001)
%   BETA, DELTA: the birth, death rates
%   FILENAME: file to store the results

networks = [
    'A3'
    'B3'
    'C3'
    'D3'
    'E3'
    'F3'
    'G3'
    'H3'
    'I3'
    'J3'
    'K3'
    'L3'
    'M3'
    'N3'
    'O3'
    'P3'
    'Q3'
    'R3'
    'S3'
    'T3'
    'U3'
    'V3'
    'W3'
    'X3'
    'Y3'
];

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%   Setup

nets = NETS;        %The number of networks simulated
netsize = NETSIZE;
reps = REPS;      %The number of repetitions of the simulation
iters = ITERS;    %The number of iterations
netype = ['net',num2str(netsize)]; %The size of network simulated
topology = TOPOLOGY; %can be scalefree or homogeneous
timeslice = TIMESLICE;%change in update2 also

beta = BETA;
delta = DELTA;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

start = cputime;

dI = zeros(iters,1);
meanI = zeros(iters,1);

for i=1:nets

    %Load network
    cd networks

    if ~strcmp(topology,'homogeneous')
        disp('Loading Network...');disp(networks(i));
        eval(['load ' netype networks(i,:) ';']);
        net = net|net'; %The matrix is symmetric cuz undirected
    else
        net = sparse(netsize,netsize);
        disp(networks(i))
```

```
    end

    disp('Parallel Updating...');
    cd ..

    for j=1:reps
        %Checks only the virus prevalence
        eval(['dI = dI + updatesir(net,' num2str(iters) ',' num2str(beta)  ',\
        num2str(delta) ',TIMESLICE,TOPOLOGY);']);
    end

    dI = dI ./ (reps * ones(iters,1));

    meanI = meanI + dI;
    dI = zeros(iters,1);

end

disp('Analysis...');

meanI = meanI ./ (nets * ones(iters,1));

%Get maximum slope of the I progression for experiments before pi
slope = diff(meanI);
maxslope = max(slope);

lim95  = (19*netsize)/20; %The 95% limit
time95 = find(meanI>=lim95);

if time95 ~= []
    time95 = time95(1);
else
    time95 = 0;
end

disp('Done');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%% Saving

cd psidrsims

%Save parameters
parameters = struct('beta',beta,'delta',delta,'nbnetworks',nets,'netsize',\
                    netsize,'nbiterations',iters,'nbreplications',reps,'topology'\
                    ,topology,'timeslice',timeslice);

save(FILENAME,'meanI','slope','maxslope','time95','parameters');

cd ..

varargout{1} = (cputime-start);
```

BIBLIOGRAPHY

[1] Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[2] L.A.N. Amaral, A. Scala, M. Barthélémy, and H.E. Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences of the United States*, 97(21):11149–11152, 2000.

[3] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[4] Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale free characteristics of random networks: the topology of the world-wide web. *Physica A*, 281:69–77, 2000.

[5] Vladimir Batagelj and Andrej Mrvar. Pajek – program for large network analysis. *Connections*, 21:47–57, 1998.

[6] Lora Billings and I. B. Schwartz. Exciting chaos with noise: unexpected dynamics in epidemic outbreaks. *Journal of Mathematical Biology*, 44:31–48, 2002.

[7] Marián Boguná and Romualdo Pastor-Satorras. Epidemic spreading in correlated complex networks, Checked on the 30/08/2002. In cond-mat/0205621 (www.arxiv.org).

[8] Stefan Bornholdt and Holger Ebel. World wide web scaling exponent from simon's 1955 model. *Physical Review E*, 64:035104R(R), 2001.

[9] caida.org. Dynamical graphs of computer prevalence, 2002. Checked on the 30/08/2002 at http://www.caida.org/dynamic/analysis/security/code-red/.

[10] caida.org. The spread of the code-red worm (crv2), 2002. Checked on the 30/08/2002 at http://www.caida.org/analysis/security/ code-red/coderedv2_analysis.xml.

[11] Shenghong Chen. Foot-and-mouth disease spread in this small world. Master's thesis, School of Cognitive and Computing Sciences, University of Sussex, 2001.

[12] Eric Chien. Malicious threats of peer-to-peer networking. Technical report, Symantec Security Response, 2001. Available at http://securityresponse.symantec.com/avcenter/reference/p2pnetworking.pdf.

[13] Frederick B. Cohen. *A Short Course on Computer Viruses, 2nd edition*. John Wiley & Sons, New York, 1994.

[14] Benjamin M. Bolker David J. D. Earn, Pejman Rohani and Bryan T. Grenfell. A simple model for complex dynamical transitions in epidemics. *Science*, 287:667–670, 2000.

[15] Zoltan Deszö and Albert-László Barabási. Halting viruses in scale free networks, Checked on the 30/08/2002. In cond-mat/0107420 (www.arxiv.org).

[16] Sergei N. Dorogovtsev and Jose F.F. Mendes. Evolution of networks. *Advances in Physics*, 51:1079–1187, 2002.

[17] Sergei N. Dorogovtsev, Jose F.F. Mendes, and A.N. Samukhin. WWW and internet models from 1955 till our days and the "popularity is attractive" principle, Checked on the 30/08/2002. In cond-mat/0009090 (www.arxiv.org).

[18] Holger Ebel, Lutz-Ingo Mielsch, and Stefan Bornholdt. Scale free topology of email networks, Cheked on the 30/08/2002. In cond-mat/0201476 (www.arxiv.org).

[19] Ralph P. Grimaldi. *Discrete and Combinatorial Mathematics, an applied introduction, 4th edition*. Addison-Wesley, New York, 1998.

[20] Atli Gudmundsson and Eric Chien (Symantec Security Response), Checked on the 30/08/2002. At http: //securityresponse.symantec.com/ avcenter/venc/data/ w32.klez.e@mm.html.

[21] H. Jeong, B. Tombor, R. Albert, Z.N. Oltvai, and A.-L. Barabási. The large-scale organization of metabolic networks. *Nature*, 407:651–654, 2000.

[22] Tomihisa Kamada and Satoru Kawai. An algorithm for drawing general undirected graphs. *Information Processing Letters*, 31:7–15, 1989.

[23] Tim D. Sauer Kathleen T. Alligood and James A. Yorke. *Chaos, and introduction to dynamical systems*. Springer-Verlag, New York, 2000.

[24] Jeffrey O. Kephart. A biologically inspired immune system for computers. In Rodney A. Brooks and Pattie Maes, editors, *Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems*, pages 130–139, 1994.

[25] Jeffrey O. Kephart. How topology affects population dynamics. In Chris Langton, editor, *Artificial Life III*. Addison Wesley Longman, 1994.

[26] Jeffrey O. Kephart and Steve White. Directed graph epidemiological models of computer viruses. In *Proceedings IEEE Symposium on Security and Privacy*, 1991.

[27] Marcelo Kuperman and Guillermo Abramson. Small world effect in an epidemiological model. *Physical Review Letters*, 86(13):2909–2912, 2001.

[28] Fredrik Liljeros, Christopher R. Edling, Luís A. Nunes Amaral, H.Eugene Stanley, and Yvonne Åberg. The web of human sexual contacts. *Nature*, 411:907–908, 2001.

[29] Alun L. Lloyd and Robert M. May. How viruses spread among computers and people. *Science*, 292:1316–1317, 2001.

[30] William M. Spears Lora Billings and Ira B. Schwartz. A unified prediction of computer virus spread in connected networks. *Physics Letters A*, 297:261–266, 2002.

[31] mcaffeasap.com. Network associates offers nimda solutions for every level of the enterprise. Checked on the 30/08/2002 at http://www.mcafeeasap.com/content/about/ presscenter/articles/09192001_nimda.asp.

[32] Christopher Moore and Mark E. J. Newman. Epidemics and percolation in small-world networks. *Physical Review E*, 61:5678–5682, 2000.

[33] David Moore. The spread of the code-red worm (crv2), Checked on the 30/08/2002. At http://www.caida.org/analysis/ security/code-red/coderedv2-analysis.xml.

[34] Y. Moreno, Romualdo Pastor-Satorras, and Alessandro Vespignani. Epidemic outbreaks in complex heterogeneous networks. *The European Physical Journal B*, 26:521–529, 2002.

[35] James D. Murray. *Mathematical Biology, (2nd, corrected edition)*. Springer Verlag, New York, 1993.

[36] Carey Nachenberg. Computer parasitology. In *Proceedings of the Virus Bulletin International Conference*, pages 1–26, 1999.

[37] Mark E. J. Newman. The spread of epidemic disease on networks. *Physical Review E*, page 016128, 2002.

[38] Mark E. J. Newman and D.J. Watts. Scaling and percolation in the small-world network model. *Physical Review E*, 60:7332–7342, 1999.

[39] Mark J. Newman. Models of the small world. *Journal of Statistical Physics*, 101:819–841, 2000.

[40] Mark J. Newman. Random graphs as models of networks, Checked on the 30/08/2002. In cond-mat/0202208 (www.arxiv.org).

[41] Romualdo Pastor-Satorras, Alexei Vázquez, and Alessandro Vespignani. Dynamical and correlation properties of the internet. *Physical Review Letter*, 87(25), 2001.

[42] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic dynamics and endemic states in complex networks. *Physical Review E*, 63:066117 1 – 066117 8, 2001.

[43] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in finite scale-free networks. *Physical Review E*, 65:035108 1 – 035108 4, 2001.

[44] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86:3200–3203, 2001.

[45] Romualdo Pastor-Satorras and Alessandro Vespignani. *Epidemics and Immunization in Scale-Free Networks*. Wiley-VCH, 2002. to be published in S. Bornholdt and H. G. Schuster (eds.) Handbook of Graphs and Networks: From the Genome to the Internet.

[46] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical Review E*, 65:036104, 2002.

[47] L. M. Sander, C.P. Warren, I.M. Sokolov, C. Simon, and J. Koopman. Percolation on disordered networks as a model for epidemics, Checked on the 30/08/2001. In cond-mat/0106450 (www.arxiv.org).

[48] Andrew S. Tanenbaum. *Computer networks, third edition*. Prentice Hall PTR, New Jersey, 1996.

[49] USENIX. *How to own the Internet in your spare time*, 2002. To appear in proceedings of the 11th USENIX Security Symposium, San Francisco (August 5-9).

[50] F. Chung W. Aiello and L. Lu. A random graph model for massive graphs. *In Proc. of the 32nd Annual Symposium on Theory of Computing*, pages 171–180, 2000.

[51] C.P. Warren, L. M. Saunder, and I. M. Sokolov. Firewalls, disorder, and percolation in epidemics, Checked on 30/08/2002. cond-mat/0106450 (www.arxiv.org).

[52] Duncan J. Watts and Steven H. Strogatz. Collective dynamics of small world networks. *Nature*, 393:440–442, 1998.

[53] Matthew Williamson. Biological approaches to computer security. Technical Report HPL-2002-131, HP Labs Bristol, Stoke Gifford, 2002. available at http://www.hpl.hp.com/techreports/.

[54] Matthew Williamson. Throttling viruses: restricting propagation to defeat malicious mobile code. In *Proceedings of Applied Computer Security Associates Conference, Las Vegas, 9-13 December*, 2002.