SELF-IMPROVING IMMUNIZATION POLICIES FOR COMPLEX NETWORKS

by

Philippe Joseph Giabbanelli

BSc., University of Nice Sophia-Antipolis, 2007

A Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

SIMON FRASER UNIVERSITY
Spring 2009

APPROVAL

Name: Philippe Joseph Giabbanelli

Degree: Master of Science

Title of Thesis: Self-improving immunization policies for complex networks

Examining Committee: Dr. Binay K. Bhattacharya, Professor
Computing Science, Simon Fraser University
Chair

Dr. Joseph G. Peters, Professor
Computing Science, Simon Fraser University
Senior Supervisor

Dr. Martin Ester, Professor
Computing Science, Simon Fraser University
Supervisor

Dr. Vahid Dabbaghian, Adjunct professor
Mathematics, Simon Fraser University
Examiner

Date Approved: *March 27, 2009*

Declaration of
Partial Copyright Licence

Abstract

To design efficient immunization strategies against viruses, we have to be aware of the properties of the graphs in which viruses spread. We review the properties found in many real-world graphs, such as small-world and scale-free, and the deterministic models that exhibit them. As a virus is an independent entity, our modeling takes into consideration parameters related to agents, such as their heuristics and their memory. We perform a 2^k factorial design to identify the contribution of the parameters of the agents and the properties of the topology. To benefit from the potential of agents to immunize dynamic networks, we specify a multi-agent system: the agents observe their environment, exchange their knowledge with minimal communication cost and fast consensus, and thus have a model of the dynamics that allows them to cope with changes. We present an algebraic framework that allows such exchange of knowledge while providing rigorous characterization.

To my father, for his dedication to work
To my mother, for her humaneness and optimism
To my stepfather, for his critical thinking.

"Le tact dans l'audace c'est de savoir jusqu'ou on peut aller trop loin"

— *Jean Cocteau,* Le coq et l'arlequin, *1979*

Acknowledgments

I would like to express my gratitude to the following people. One might not always find their direct presence in this thesis, but their impact lies between the lines. Jean-Claude Bermond introduced me to graph theory with his first year course at the University of Nice. Things would be different without his initial encouragements and teachings: firstly, this thesis would not be about graphs, and secondly there might not be a thesis. Professors at Bishop's University showed me that many, if not all topics can be thrilling depending on how they are looked at, and how they are introduced. Without them, this thesis may have been only about graphs, which would not necessary be a bad thing, but throwing in a few agents and data mining tasks surely brings some excitement.

Joe Peters supported me in numerous aspects that go beyond this thesis. Not every supervisor would try to solve problems involving eggs and flour for their students, but one cannot decently write a thesis without cakes. Many aspects stem from his guidance, not only the methodologies and the writing, but also having enough space to explore topics that appealed to me. The last chapter owes some of its existence to Oliver Schulte and Binay Bhattacharya's feedback, and to James Delgrande's insightful comments which inspired me to discuss agents. I would also like to thank Martin Ester for discussing extensively the content of this thesis and Vahid Dabbaghian for accepting to examine it.

Finally, leaving my country is an interesting experience that could have been more terrifying than exciting without the support and understanding of my family, my fellow Vancouverites (Franz-Edward Kurtzke, Chirag Vesuvala, Erkan Keremoglu, Ashgan Fararooy, Jim Parks, Javad Safaei, Muntaseer Salahuddin, ...), the frenchies (Astrid De Ligny, Philippe Semeria, Diane Berthelot) and labmates (Kianoosh Mokhtarian, Cheng-Hsin Hsu, Craig Mustard).

Contents

List of Tables

List of Figures

Chapter 1

Introduction

While health research in industrialized countries mainly focuses on cardiovascular diseases and cancer, infectious diseases remain a major cause of mortality in the world. HIV is one well-known such disease, and it affected between 30.6 and 36.1 million people as of 2007 [74]. Furthermore, global warming increases the incidence of infectious diseases: not only are insects that carry pathogens becoming more abundant, but they can also spread further north. Moreover, global warming can impact the probability of transmission for a given insect. For example, *culicoides* female flies are responsible for outbreaks of bluetongue and African horse sickness in southern Europe, and they feed on blood for the development on their eggs. As eggs develop faster with an increased temperature, the flies feed more often and thus the risk of being contaminated increases [139]. Thus, modeling infectious diseases is of crucial importance and so is the design of **immunization policies**.

The last ten years have witnessed the emergence of the field of **complex networks**, which found that many networks share common properties. Models taking into account these properties are among the more realistic models that one can use to design immunization policies. While basing our immunization policies on these properties would already confer an advantage compared to ones making weaker assumptions about the network's topology, it is far from being the end. Indeed, a fixed policy is likely to fail: most networks are dynamic, with new nodes coming and leaving, and thus a policy should be able to adapt. When little is known *a priori* about how the network evolves, a policy cannot simply be a function of time and ignore its environment.

At a lower level, a policy over a network can be seen as a decentralized spreading process, in which agents go through the network and either contaminate or immunize its members.

Fixed heuristics are not enough for a policy to evolve: it is necessary to use agents that are intelligent entities. In other words, there is a need for agents to observe their environment and deduce appropriate changes in behaviour from it. Furthermore, an agent is restricted in its observations to its immediate neighborhood, and thus only gets a partial picture of the dynamics, hence agents need to communicate and adopt cooperative behaviour. This raises further issues as not only should agents be able to cope with highly dynamics environments, similar to a data-stream setting, but they have to exchange their knowledge in a way that is efficient in terms of communication, and reach a consensus quickly. In a nutshell, the system has to be **self-improving**.

1.1 Contributions

The literature on complex networks consists of thousands of papers that define a distinct field of research. However, the field has emerged from statistical mechanics and many contributions sorely lack a computer science background, particularly from graph theory. Our emphasis is on a rigorous approach to complex networks:

- We define the main properties and highlight the pitfalls into which several recent contributions still fall.
- We show that random networks can be made small-world and/or scale-free with appropriate modifications, which are the two essential properties of complex networks. Thus, models based on random processes cannot model anything that cannot be obtained by starting from a random network, so one non-deterministic model does not present any advantages compared to another. On the other hand, purely deterministic models can be fine tuned, so instances can be normalized with respect to several properties, and deeper analytical work becomes feasible. Thus, unlike many contributions to the field, only deterministic models are used in this thesis.
- We provide a framework for immunization policies by considering four main cases, for which we highlight similarities with results from graph theory.

Having properly defined the objects of study and the approaches dealing with them, we have to identify which parameters of the problem matter the most. Thus, we formalize the behaviour of an agent and identify parameters of the agent and the topology. The

contribution of each parameter in the final outcome is evaluated by a 2^k factorial design that also allows the study of first-order interactions, whereas previous works considered simpler regressions. In order to evaluate the outcome of various situations, a simulator with a variety of models and analysis tools was implemented, as well as a user friendly interface; this simulator, as of 2009, is one of the most complete dedicated to complex networks.

The design of a cooperative agent system is a complex process and we provide background on agents and data mining for the sake of clarity. Then, we investigate the architecture of the system and its challenges. For some challenges, we point out related work in areas such as sensor networks, and we propose new approaches when little work is available to face the challenge. In particular, we developed a framework so that agents can exchange their knowledge with minimal communication cost while reaching a fast consensus. This framework also specifies an algebra which characterizes the properties of the process, and can also be used to examine the properties of other similar processes.

Overall, our work explores several components of a self-improving immunization system and ensures that each of them is sound. A significant amount of future work can already be found on the use of such components. For example, agents can understand as a group how a vertex dies by using our framework for knowledge exchange. However, taking this knowledge into account to determine if a vertex should receive a vaccine is still an open problem. Furthermore, cooperative agents may have roles and team strategies, which opens up perspectives for creative approaches. Thus, extending and using the components designed in this thesis already offers many opportunities for future research. Among these, we describe two original problems for our which initial work remains at an early stage:

- Agents can learn about the strategies of their opponents in a similar way that they can learn the dynamics by observing the networks. In order to do so, we design an algorithm that extracts information about the behaviour of an agent as it gets through a component. However, several agents roaming through the component simultaneously, or partly random strategies, remain to be captured by the algorithm.

- The connections between the agents that allow them to communicate observations may be disrupted because the network is dynamic. If an agent was in contact with exactly one other agent, it should establish a new route, and some agents may be located at key points that are easier to find under certain navigation schemes. We propose an algorithm to find key points that establish good coverage of the network,

but it is not completely local and thus needs to be parallelized in order to be run by agents.

Finally, we also propose a new theoretical problem relating graph minors and the scale-free property. Scale-free networks can be obtained by vertex contraction, and we present an algorithm that allows a fine tuning of the degree distribution. As vertex contractions are related to minors, there could be base networks on which the vertex contraction process inherits interesting properties, particularly for navigation and labelling. We show how to inherit the navigation properties from a hypercube, and this suggests a thorough study of families of networks for which properties may be inherited through a vertex contraction process.

1.2 Outline

In Chapter 2, we propose a rigorous approach to complex networks. In Chapter 3, we identify key parameters, find typical values from real-world cases, and examine the impact of the parameters through a 2^k factorial design using a custom-made simulator. In Chapter 4, the design of intelligent agents in a dynamic system is explained, with emphasis on the notion of agents and their capabilities as well as the requirements of such a system. In Chapter 5, we present an algebraic framework that allows an efficient exchange of information between agents and characterizes it formally. In Chapter 6, we suggest numerous directions for future work based on research at an early stage, ranging from a new scale-free model that uses vertex contraction to finding key positions of agents if communications are disrupted. We also provide three appendices: Appendix I explains the formalism of percolation theory that can be found in most of the papers cited in Chapter 3, Appendix II completes the discussion about agents from Chapter 4, and Appendix III explains concepts and tools of data streams that are useful if the observed system is highly dynamic and of massive size.

Chapter 2

Background

The basic question that disease models study is: how does the disease *spread*? This finds a strong correspondance in computer networks, in which the last ten years have witnessed a greater use of the mathematical models for diseases in the context of worms. Furthermore, a spread is, after all, one of the many names for the dissemination of an information through a population, such as marketing campaigns. In order to analyze accurately a spread in a population, a natural question to ask is what this population *looks like*. In other words, to better understand the dynamics of a process such as a spread, we take into account the topology of the network on which it takes place. The emergence of the new field of *complex networks* sheds light on these topological aspects, through its broad study of properties found in many real-world networks and its numerous models.

In Section 2.1, we introduce the fundamental concepts of mathematical biology that are often used to analyze spreads. Then, Section 2.2 presents the theory of complex networks and its implications in the context of spreads: we start by discussing the main properties, such as the now well-known *scale-free* and *small-world* properties, and show how they can be modelled with a particular emphasis on deterministic approaches. In Section 2.3, we explain how this vast knowledge of complex networks is applied of two aspects of spreads. Firstly, we review fundamental questions, such as "*Is there an epidemic threshold in scale-free networks ?*", for which erroneous claims are often made. We also suggest further analysis, for example, of the influence of the starting point of a spread. Secondly, we propose four categories for immunization strategies and we review recent work for each of them. We also suggest improvements by pointing out links to methods not yet exploited that are known to be promising for very similar situations.

2.1 Mathematical modelling in epidemiology

The epidemiology models currently in use are based on the ones proposed by Kermack and McKendrick [81] in 1926. A model has two components: labeled compartments, or *epidemiological classes*, standing for a certain category of the population, and rules specifying the flows of population between the compartments. From a computer science perspective, the rules specifying the movements of individuals among classes can be modelled by a finite state automata: each compartment in which individuals can be classified is mapped to a state of the automaton, and a "rule specifying the flow" is simply a transition between two states. When modelling the spread of a disease, one typically starts with the states: for example, the SIS model developed by Kermack and McKendrick has two states: Susceptible $\mathbb{S}$ (a healthy individual that may contract the disease) and Infected $\mathbb{I}$. Then, all possible transitions are specified: the probability $p(\mathbb{S} \to \mathbb{I}) = \alpha$ that a healthy individual get infected, and $p(\mathbb{I} \to \mathbb{S}) = \beta$, the probably that an infected individual can be cured. This yields the automaton illustrated in figure 2.1(a). The states most commonly used are [64]:

- $\mathbb{M}$, for new-born infants protected by maternal immunity a few months after birth.
- $\mathbb{S}$, for individuals susceptible to get a disease because they have no particular immunity.
- $\mathbb{E}$, for individuals exposed to the disease but not infectious yet.
- $\mathbb{I}$, for infectious individuals who can transmit the disease.
- $\mathbb{R}$, for individuals removed from the possibility of being infected because they have been isolated or immunized. Immunization can result from vaccine policies or because certain diseases confer immunity against reinfection, which is generally the case for the ones transmitted by viral agents such as measles.

Individuals can also die for various causes regardless of the state in which they are, thus the automaton can be equipped with a sink state $\mathbb{D}$ standing for dead individuals. This is not common practice in epidemic models, mainly because they are traditionnaly represented with diagrams rather than automata. The number of states depends on the number of aspects that one is willing to take into consideration when modelling a disease, as well as the particular properties of this disease. For example, in the SIS model illustrated in figure 2.1(a), we ignore individuals of class $\mathbb{R}$, *i.e.* immunization or quarantine policies are not taken into account. On the other hand, the situation illustrated in figure 2.1(b) considers

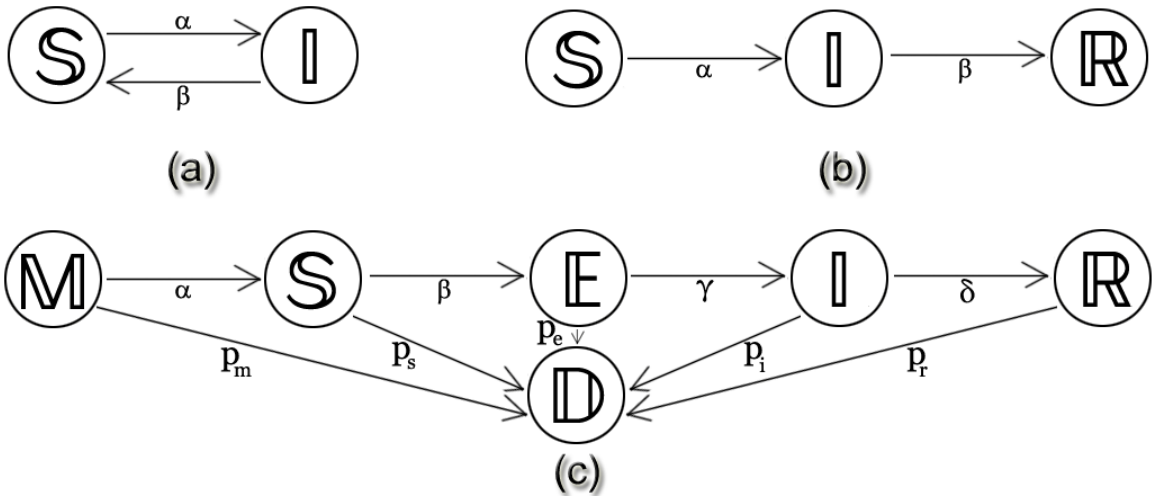

Figure 2.1: Three epidemic models represented as finite automata.

that all infected individuals can eventually acquire immunity to the disease. Acronyms for epidemic models are generally based on the ordering of transitions, thus 2.1(b) is refered to as SIR, while 2.1(c) is known as MSEIR. In the latter, if the acquired immunity for individuals in state $\mathbb{R}$ was only temporary, then there would be another transition from $\mathbb{R}$ to $\mathbb{S}$ and thus the model would become MSEIRS.

The goal of epidemic models is to study the dynamics of a disease, *i.e.* how the system changes over time. As the change from one time step to another corresponds to a derivative, a model is typically expressed by a system of differential equations. For example, the SIS model of figure 2.1(a) is expressed with the following two equations, where S and I stand for the size of the population in states $\mathbb{S}$ and $\mathbb{I}$ respectively, and $\alpha \times S \times I$ is the effective contact rate between the population in states $\mathbb{S}$ and $\mathbb{I}$:

$$\begin{cases} \frac{dS}{dt} = & -\alpha \times S \times I + \beta \times I \\ \frac{dI}{dt} = & \alpha \times S \times I - \beta \times I \end{cases} \tag{2.1}$$

The main quantity of interest is the *basic reproduction number* R_0, defined as "the average number of secondary infections produced when one infected individual is introduced into a host population where everyone is susceptible" [64]. This quantity is an important threshold in most models: if $R_0 > 1$ then each infectious individual will infect more than one other individual on average and thus the disease will become an epidemic. On the other hand, if $R_0 < 1$ then the disease will die out. Typical values for R_0 can range from

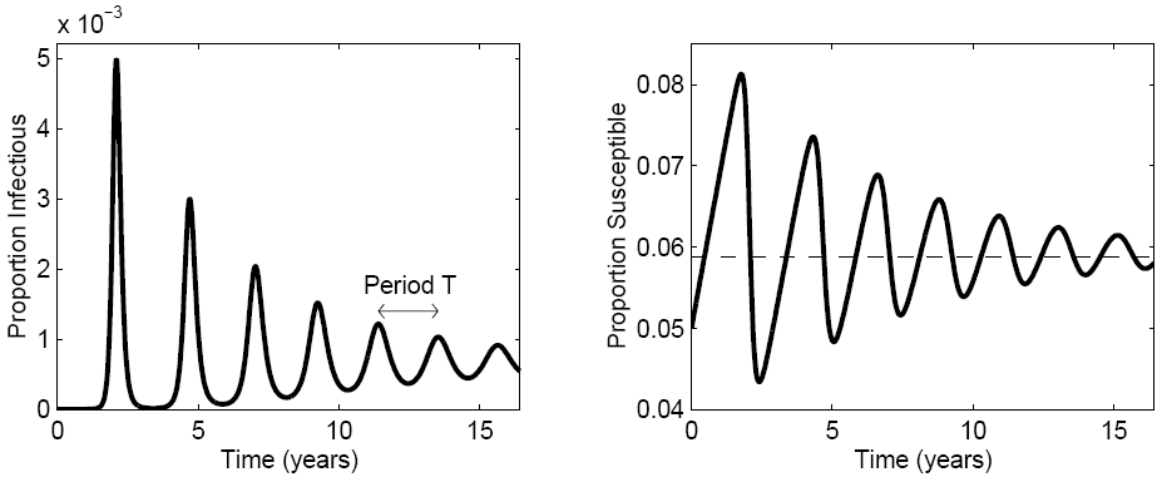

Figure 2.2: Inter-epidemic oscillations of an SIR model taking into account birth and death.

4 for smallpox to 100 for malaria [78]. As the SIS model has very simple dynamics, R_0 is found analytically by studying the (S, I)-plane [21]; in general, systems of two differential equations can be studied by finding and solving the nullclines (boundary where one of the derivatives changes of sign) algebraically, using them to find stable points (*i.e.* equilibria), and determining the stability of those points. However, finding all equilibria and studying their stabilities requires numerical simulations for larger systems of equations, as there can be complex phenomena such as nonlinear oscillations (especially if delays are introduced in the model). For example, "a chain of removed [states $\mathbb{R}_1, ..., \mathbb{R}_n$] can delay the return of temporarily immunize individuals to the susceptible [state $\mathbb{S}$], and thus lead to periodic oscillations" [65]. The dynamics of an SIR model using parameters for measles in England and Wales is shown in figure 2.2 from [78], and already exhibits simple oscillations.

2.2 From automata to complex networks

An important assumption of the models presented in the previous section is that of a homogeneous population. Indeed, we assumed that all individuals that fall within a certain state have the same probability of going through a transition. However, the contacts between individuals play a key role in the transmission of infectious diseases. Consider a graph in which individuals are vertices, linked by an edge if they are in contact with each other. At the graph level, the assumption of a homogeneous population of size n means a

complete graph K_n, in which each single person is in contact with all others. This clearly over-simplifies the topology of a population and, as a result of this low accuracy, only very broad questions can be answered by such models. For example, only random immunization strategies can be designed, as all individuals are considered equal and none of them would present particular advantages.

Research in the structure of real-world networks has grown tremendously within the last ten years, and now offers particularly valuable insights to better understand the topology of a population. First, *complex networks* emerged as a new field of physics that determines the properties of real-world networks, as diverse in origins as citation networks, protein interactions or the network of Hollywood's actors. Then, once the properties found in broad families of networks were known, phycisits as well as computer scientists and graph theorists have designed models to generate networks, and studied the consequences of those properties in more depth. Finally, there are important repercussions to the analysis of networks of individuals, which is referred to as Social Network Analysis (SNA), and is now a multi-discipline active field, encompassing researchers and methods from sociology as well as computer science, geography (in particular geographical information systems), and criminology.

The reminder of this section will explore recent advances in complex networks. We first present the main properties shared by real-world networks, and then show how they can be matched by models. Applications of these properties to epidemiology are reviewed in the next section.

2.2.1 Properties of real-world networks

Our approach is bottom-up: we start from the basic blocks networks are made of, and we progressively move toward higher-level topological properties. Although uncommon, this approach has recently been advocated: "traditionally, complex networks are classified on the basis of their global properties, but taking into account the modular structure [...] leads to a better understanding of how the underlying systems work" [93].

Using basic blocks to establish a profile

A default assumption in use for many years has been that networks have no particular properties, and thus they were simply considered as being random. While this is far from

the truth, it serves as a *null hypothesis*: given a network A, we can compare it to a random network B in order to discover significant differences, that will indicate the presence of various properties. One of the new findings resulting from this method is the presence of *motifs* [97, 98]. A motif is a block, or a *subnetwork*, that occurs at a significantly different frequency[1] from what would be expected in a random network.

The *significance profile* (SP) of a network is a summary of the differences in occurences of motifs between this network and an ensemble of random networks that have been normalized in size and degree. For example, the *triad significance profile* (TSP) will summarize the differences in frequency for each of the subnetworks of three nodes that one is interested in. An illustration is given in figure 2.3 simplified from [98], for three microorganisms networks (bacteria *Escherichia coli* and *Bacillus subtilis*, yeast *Saccharomyces cerevisiae*), three networks of web pages (related to university, literature, or music), and three social networks (inmates in prison, sociology freshmen and college students). These profiles are useful for two reasons. First, the presence of basic blocks indicates at a low level what the system is attempting to do efficiently: for example, in microorganisms motif 7 is very frequent and corresponds to a feedforward loop that has been shown to be useful to perform particular tasks. Finally, models of networks are based on high-level topology properties, and a few parameters are usually required to generate them: a significance profile can be used to tune a model by making it also match low-level features.

Clustering coefficient and communities

As shown in Figure 2.3, motif 13 is very frequent for web pages and social networks. This motif corresponds to a small community: three individuals that know each other. Communities are of particular interest in epidemiology because diseases spread faster within a set of densily connected individuals than among individuals having few contacts. Furthermore, identifying communities in a population is the basis of community-wide quarantine, in which a community is isolated to prevent a disease from reaching it. For example, figure 2.4(a) shows that if the links between the communities are cut, then a disease cannot spread outside of the community from which it originated. To estimate how close is a node to being a member of a community, we compare its immediate neighborhood to a complete graph

[1] In [97], the authors define motifs as subnetwork that occurs at a "higher" frequency. However, we think that what truly matters is the difference, and not whether it is higher or lower: in both cases, it indicates particular network's properties. Thus, we use the more general definition from [98].

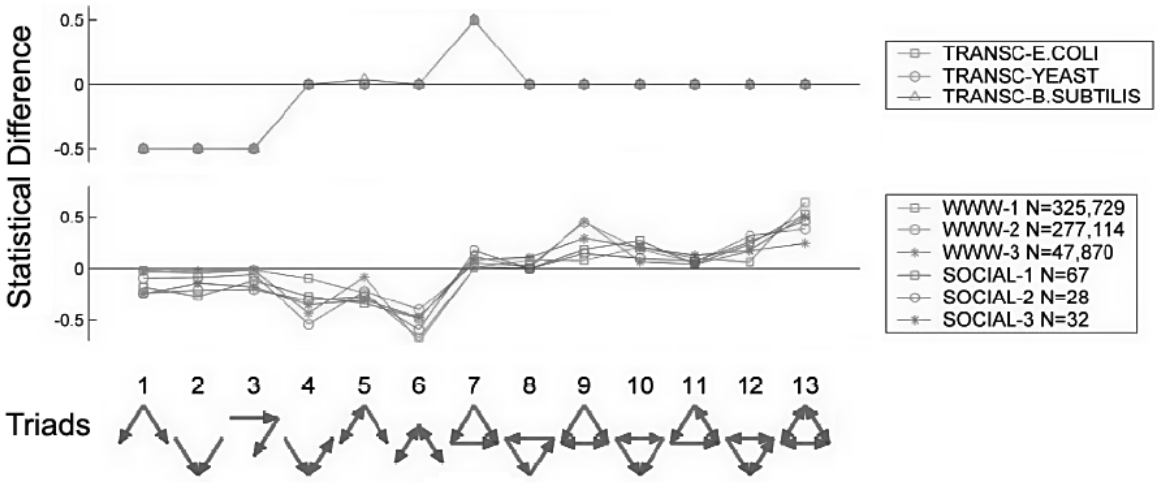

Figure 2.3: Triad significance profile (TSP) for microorganisms networks, web pages and social networks.

K_n: we count the number of edges in this neighborhood, and divide by the number of edges found in a complete graph of the same size. These notions are formalized in Definitions 1, 2 and 3; generalizations are discussed in [123], and Figure 2.5 shows the number of nodes per clustering coefficient in samples of a blog network.

Definition 1. Let $G = (V, E)$ be a graph with a set of vertices V and edges E. Then, the neighborhood N_i of $v_i \in V$ is:

$$N_i = \{v_j | e_{ij} \in E \vee e_{ji} \in E\}$$

Definition 2. The clustering coefficient of a vertex $v_i \in V$ is:

$$\begin{cases} C_i = \frac{|\{e_{jk}\}|}{|N_i| \times (|N_i|-1)}, v_j, v_k \in N_i, e_{jk} \in E \text{ if } G \text{ is directed.} \\ C_i = \frac{2 \times |\{e_{jk}\}|}{|N_i| \times (|N_i|-1)}, v_j, v_k \in N_i, e_{jk} \in E \text{ if } G \text{ is undirected.} \end{cases}$$

By convention, we set $C_i = 0$ if $|N_i| = 0$.

Definition 3. The clustering coefficient of a graph is:

$$C = \frac{1}{|V|} \times \sum_i^n C_i$$

Communities have a long history in sociology. For example, in the early 70's, it was found that one's friends are likely to share the same information regarding a job, and thus are of little help when looking for work; on the other hand, individuals from outside the community

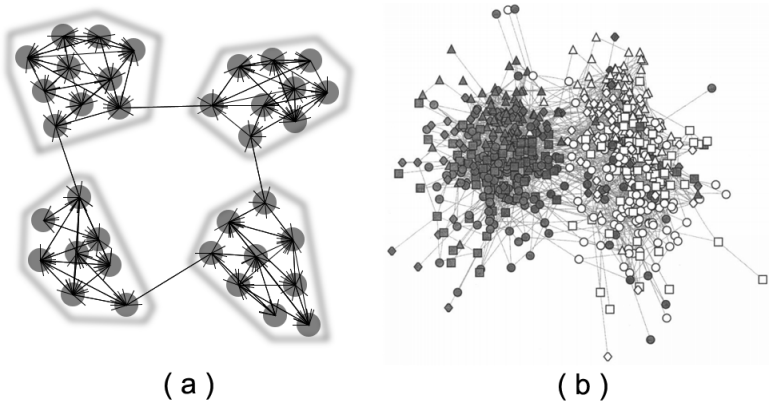

Figure 2.4: Four easily separable communities (a). Friendship relations in "Countryside High School", with shaded figures representing non-white students (b).

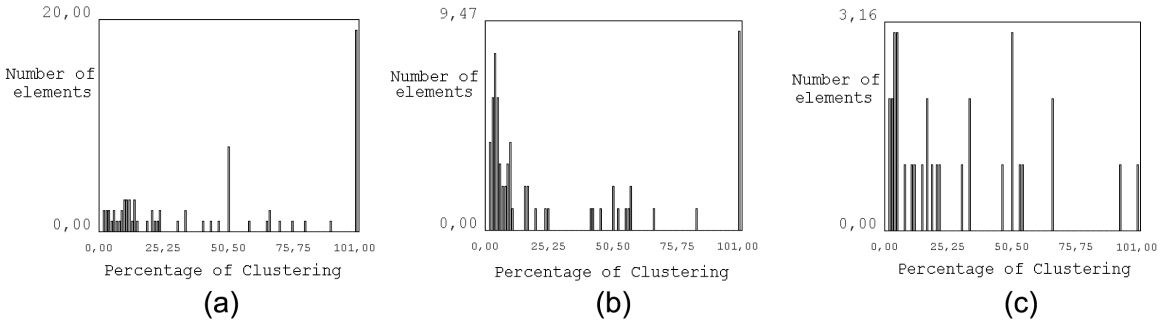

Figure 2.5: Number of nodes versus clustering coefficients in three samples of a blog network, of respective sizes N = 11965, 9401, 4165.

are likely to be aware of other offerings and are thus more efficient for job search [58]. A more recent example showed how racial segregation was underlying the social networks of children in a US school (see figure 2.4(b) from [101]). From a computer science perspective, detecting communities is also a well studied problem, referred to as *graph clustering*; we shall discuss that topic in section 2.3.

Average path distance and the small-world effect

We first considered small blocks of the networks and then larger ones with communities. We are now looking at the overall network with the following question: what is the average distance between any two nodes? While the question is straightforward, the results from Watts and Strogatz turned out to be surprising: in networks as large as 225,226 actors

from the Internet Movie Database, the average distance was only 3.65 [136]. This has an immediate effect on the spread of diseases: starting from a random actor, it takes an average of 3.65 'steps' for any other actor to be infected.

A similar finding resulted from Milgram's experiments [96] in 1967: letters were sent to random inhabitants of Wichita (Kansas) and Omaha (Nebraska). They were asked to forward the letter to some designated inhabitants of the other town, with the rule that they could only send the letter to somebody they know. The letters that successfully reached their targets used an average of only 6 intermediates, which was surprisingly small for random inhabitants in distant towns[2].

In honor of Milgram's article *The Small World Problem*, Watts and Strogatz used the term Small-World for their findings. However, there is a major difference between the two meanings of *Small World*, which can be a source of confusion. First, a small average distance is actually normal for random networks: in the randomized version of the actor network, the average distance would be even smaller at 2.99 [136]. What really was unexpected was the clustering coefficient: while the randomized network has a low clustering coefficient of $C = 0.00027$, the actor networks had $C = 0.79$. Thus, not only is a spread efficient at a global level, but also at the local level. In this thesis, we will use the term *small-world* to refer to a spread that is efficient at both the global and local levels: the average distance must be at most logarithmic in the number of nodes, and the clustering coefficient must be high[3].

While the clustering coefficient and average path length of the network provide enough information to detect the presence of the small-world effect, it is often interesting to observe

[2]Milgram's experience became popular culture when John Guare turned it into a play in 1990, which eventually became a movie starring Will Smith and Donald Sutherland. This belief that we are linked to any person in the world by a few intermediates can even be dated back to the fictious novel *Lancsemek* (Chains), published in 1929 by the popular Hungarian author Frigyes Karinthy. However, from a scientific perspective, such experiences are rather debatable. Indeed, in Milgram's case, only 42 of over 160 letters eventually reached their targets, which he explained as "people didn't bother sending the letters on", while it would seem more likely that they never found a path [84]. Furthermore, the participants were not aware of the whole topology of the network, and thus the letters that reached their destinations did not use the shortest path. Thus, six intermediates is a very rough approximation from which little can be cluded about the network's properties.

[3]We insist on this distinction as it is often unclear in the literature. For example: "this feature is known as the small-world property and is mathematically characterized by an average shortest path length [...] that depends at most logarithmically on the network size N. [...] The small-world property in real networks is often associated with the presence of clustering." [15], or "the small-world property refers to the fact that in many large scale networks the average distance between vertices is very small compared to the size of the graphs. [...] More interesting is the fact that, in close analogy to many social and technlogical networks, the small-world effect goes along with a high level of clustering." [23]

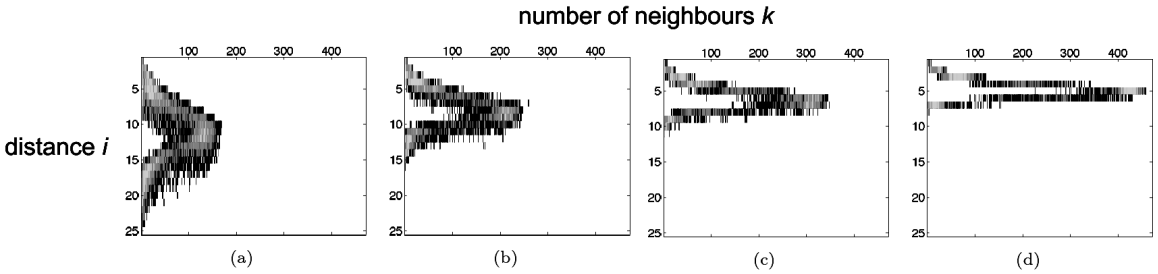

Figure 2.6: Decreasing average path length from (a) to (d), plotted in a B-matrix.

it at a smaller scale, as irregularities might provide additional information. The clustering coefficient can be observed in graphs such as those displayed in figure 2.5. For the average path length, it has recently been proposed to plot the number of nodes versus the number of neighbours k at distance i. As this technique, called a *B-matrix* [9], has three parameters, it is convenient to visualize it either in three dimensions or through the use of colours. An illustration is given in figure 2.6 from [9] to show a progressively decreasing average shortest path length (colours are available in the online version of the paper). A less accurate picture of the average path length can also be obtained by plotting the number of nodes versus average path length.

Degree distribution and the scale-free effect

Examining a general parameter as simple as the distance between two nodes revealed the small-world effect. Similarly, we can examine the degree distribution, *i.e.* the distribution of the sizes of the neighborhoods of nodes. The general intuition when looking at the percentage of objects with a certain property is that it peaks around a typical value: for example, the typical height of a male individual should be between 150cm and 200cm, with a peak around 175cm [104]. However, numerous distributions contradict this intuition: they are *right-skewed*, *i.e.* most of the distribution has fairly small values while the left-most values are several orders of magnitude higher; other names for this distribution are heavy-, fat-, or long-tailed. This is illustrated in figure 2.7(a) from [104]: most cities have a fairly small population, while a few can reach a population 150,000 times larger. In figure 2.7(b), this distribution is plotted on a log-log scale and shows a line. The equation of this line is of the form $ln\ p(x) = -\alpha \times ln(x) + c$, which leads to $p(x) = e^c \times x^{-\alpha}$. A distribution of the form $p(x) = C \times x^{-\alpha}$ is called a *power-law with exponent alpha*. Thus,

the distribution of city populations follows a power law. In 1955, Herbert Simon showed that this "class of distribution functions [...] appears in a wide range of empirical data – particularly data describing sociological, biological and economic phenomena" [124]. For example, the distribution of scientists by number of papers published, and the distribution of incomes in the general population, were both found to follow a power law.

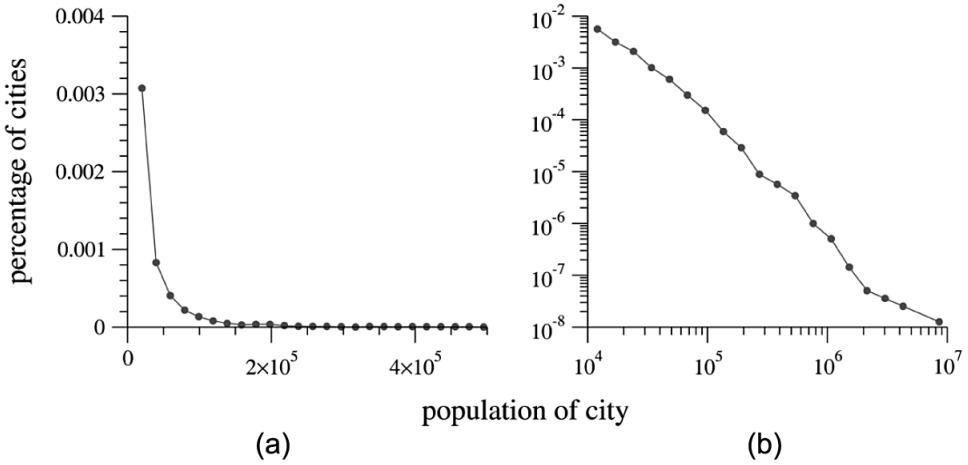

Figure 2.7: Distribution of city populations (a), and the same distribution in a log-log scale (b).

Functions $p(x)$ that follow a power-law are the only ones that satisfy the equation $p(a \times x) = g(a) \times p(x)$. In other words, "an increase by a factor a in the scale by which one measures x results in no change to the overall density $p(x)$ except for a multiplicate scaling factor" [90]. Thus, those functions are the only one that look the same, regardless of the scale at which we look. For this reason, the term *scale-free* is equivalent within the physics community to a function following a power-law; by extension, a network is *scale-free* if its degree distribution follows a power-law. One usually considers two cases: (1) $2 \leq \alpha < 3$, and (2) $3 \leq \alpha \leq 4$. Most networks that have been observed to exhibit a power-law have an exponent of the first class [106], for which we also have the second moment $\int P(x)x^2dx = \infty$, in other words the fluctuations in degree are unbounded; this led to another explanation of the term *scale-free*, but it should be made clear that it only holds for $2 \leq \alpha < 3$: "the absence of any intrinsic scale for the fluctuations implies that the average value is not a characteristic scale for the system" [23].

Scale-free networks gained interest as a result of the claim by Barabasi and Albert that "[the power-law distribution of degree] was found to be a consequence of two generic

mechanisms: (i) networks expand continuously by the addition of new vertices, and (ii) new vertices attach preferentially to sites that are already well connected" [12][4]. Since this claim, a rich literature on scale-free networks has emerged with a variety of different definitions. While some definitions simply encompass consequences of a power-law degree distribution, others involve characteristics that are not consequences and should be considered to be additional assumptions. Consequences and additional assumptions are clarified in [90].

Additional properties

Numerous properties have been found in complex networks. While a complete list is beyond of the scope of this thesis, three are worth mentioning. Firstly, individuals "tend to associate preferentially with people who are similar" [106]; this effect is known as assortativity, and is useful when looking for communities on the basis of similar characteristics, or determining how new individuals will connect to an existing population. This notion is directly connected to the spread of diseases within communities, or transmitted by migrant populations. Secondly, it was found that the distribution of cycles of length h in a network of size N peaks around a characteristic value $h_* \ N^{\alpha}$; thus, h_* and α were proposed as an additional way to characterize families of networks [121]. As cycles are one of the parameters of navigation in a network, further research could explore the relations between h_*, α and the properties of a spread. It can also be noted that studies of cycles generalize the ones on transitivity (cycles of length 2). Finally, it was advocated that self-similarity may be a third commonly shared properties to networks, with small-world and scale-free [125]; a network having the self-similarity property is refered to as a *fractal network*. *Hierarchical networks*, to be discussed later, use a similar approach in which the network is built by repetition of basic structures organized in a hierarchical fashion.

2.2.2 Models for real-world networks

How good is random?

The concept of random network was first proposed by Rapoport [118] and independently discovered later, in a formal approach, by Erdos and Renyi [44]. A random network $G_{N,p}$

[4]The claim started a very rich literature because it suggested that a phenomenon such as preferential attachment could underlie networks such as the Internet. Thus, the popularity of the subject is mainly due to its potential applications rather than to the discovery of new methods. Indeed, "models of preferential attachment giving rise to power law statistics actually have a long history and are at least 80 years old" [90].

has N nodes, and each pair of nodes is connected by a link with independent probability p. Those networks were studied to determine when properties such as "K_4 is a subnetwork" or "there is a giant component whose diameter is the diameter of the network" hold. More formally, there is a critical probability p_c such that if $p < p_c$ then a property Q almost certainly hold, and if $p > p_c$ then Q almost certainly does not hold. This threshold is defined as a function of the network's size N, and denoted by $p_c(N)$, or $p_c(N \to \infty)$ in the limit for large networks. The main goal in the study of random networks was to determine $p_c(N)$ for a given Q. Thus, although random networks have been used to represent a population's topology, they were neither designed nor studied for that goal. Indeed, they do not have any of the properties found in real-world networks. First, they do not exhibit the small-world property: albeit they exhibit a small average distance $\frac{ln(N)}{ln(\bar{k})}$, where $\bar{k}$ is the average degree, they do not qualify because of their low clustering coefficient $C \overset{N\to\infty}{\to} 0$. Furthermore, they are not scale-free either, as their degree distribution for large N can be approximated by a Poisson distribution $P(k) = e^{-\bar{k}} \times \frac{\bar{k}^k}{k!}$. However, we will show in the following paragraphs that random networks can be modified, or "generalized", to exhibit such properties.

A random network can be generalized by providing the degree distribution $P(k)$ as an input, and thus one can use a power-law distribution to obtain a scale-free effect. Such generalization has mainly been achieved in three different ways. The oldest one is the *configuration model*: a degree sequence $D = \{k_1, ..., k_N\}$ is provided, such that $\sum_i k_i = 2|E|$, and the degree distribution will tend to $P(k)$ for N large enough [100]. An alternative was proposed, based on *probability generating functions*. The idea of a generating function is to wrap a sequence into a single mathematical object: the sequence is indexed by natural numbers, and we use derivatives to access one specific element. As the function contains all the information, we say that it *generates* the sequence. For example, the first function used in [107] encodes the degree distribution with $G_0(x) = \sum_{k=0}^{\infty} p_k x^k$, and the probability p_k is accessed by taking the *k-th* derivative of G_0: $p_k = \frac{1}{k!} \times \frac{d^k G_0}{dx^k}$. A recent alternative was of a *given expected degree sequence* $D = \{w_1, ..., w_N\}$, in which a link between two nodes i and j exists with probability $p_{ij} \propto w_j \times w_j$.

Although not often used, given numerous alternative models that will be presented throughout this section, it is possible to transform a graph in order to increase its cluster-

ing coefficient[5]. The functional requirements of this transformation can be specified as follows [14]: given a connected network G and a target clustering coefficient C_T, the goal is to produce a new network G' such that

(1) G' is connected.

(2) G and G' have the same degree distribution $p(k)$.

(3) G' still has low diameter.

(4) $C_{G'} \geq C_T$.

Conditions (1) - (3) indicate invariants under the transformation and (4) is the goal. Clearly, condition (2) imposes rewiring as the only solution that does not modify the overall degree distribution. An algorithm was proposed in [14], by rewiring all links on loops that do not favor the clustering coefficient.

It is also possible to transform random networks to obtain other effects, such as a higher assortativity based on the degree. Indeed, a simple three-step process was proposed [143]: randomly choose two links, order their four end-nodes with respect to their degree, rewire with probability p to connect small degree vertices together (similarly for high degree vertices).

Given the constructions that we presented, *there is no reason not to use a generalized random network if all we want is a topology respecting certain properties.* However, there are two reasons that motivate the development of purely deterministic models (*i.e.* in which there are no probabilities): deeper analytic work becomes feasible (such as techniques from spectral theory), or one wants to control precisely the structure that is generated (for example for simulations). The former is a motivation for our work. In the remainder of this thesis, we will not discuss models involving probabilities unless it is necessary to understand how a deterministic version has been designed. Indeed, the result of such models can as well be achieved by generalized random networks and thus they are not particularly interesting. The classification used in complex networks considers static models (all nodes are present at the beginning and the algoritm adds the edges) or dynamic/evolving models (nodes and edges are added throughout the algorithm)[6]. We use a different distinction, motivated by a

[5]Methods have recently been proposed to "turn a graph into a small-world" [42]. However, the goal of such methods is not to increase the clustering coefficient but rather to create a graph in which decentralized navigation can be as good as in a small-world.

[6]The distinction is sometimes blurry, for example: "the network is both static and dynamic, since new vertices can be added to the structure at any stage" [23]. If such a distinction is used, we rather advocate for the viewpoint of expansion's flexibility: if it is possible to add a few nodes to the network generated by

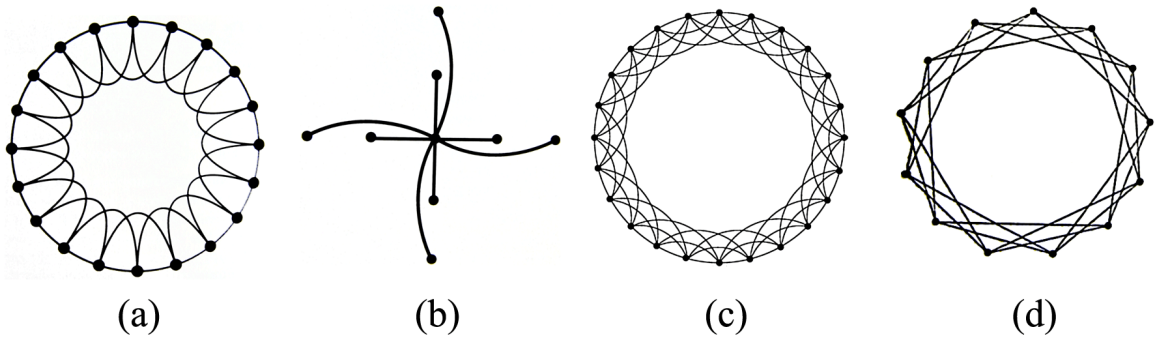

Figure 2.8: 1-lattice with k = 4 (a). 2-lattice with k = 8 (b). Circulant graph $C_{24,6}$ (c). Double step graph (d).

computer science perspective: either an algorithm is deterministic, or it is randomized.

Deterministic small-world model

Watts and Strogatz [136] first proposed a non-deterministic small-world model. As deterministic versions are based on a similar structure, we introduce the Watts and Strogatz (WS) model for the sake of clarity. This model relies on a lattice structure, as defined below and illustrated in figure 2.8(a-b).

Definition 4. A lattice in d dimensions is called a d-lattice. There are N nodes, all of degree k. Each node v is connected to its lattice neighbours u_i and w_i such that:

$$u_i \equiv v - i^{d'} \pmod{N}$$
$$w_i \equiv v + i^{d'} \pmod{N}$$
$$1 \leq i \leq \tfrac{k}{2},\ 1 \leq d' \leq d, \text{ and commonly } k > 2 \times d$$

The WS model starts with a low-dimensional lattice and rewires a link (*i.e* modifies the endpoints of the link) with probability p. This probability allows for an interpolation between a regular lattice with a strong clustering coefficient (case $p = 0$) and a random network with a low average distance (case $p = 1$). Experiments found that a good value of p for the small-world effect is $0.01 < p < 0.1$ (see figure 2.9 from [136]). Thus, rewiring a few edges is enough to guarantee the small-world effect. As these rewirings are reducing the

the same process without significantly changing the properties, then such process is dynamic, and otherwise static. In that sense, random networks without an imposed degree sequence are highly flexible/dynamic, whereas a hierarchical network may require the creation of a whole new layer in order to not significantly change its properties and is thus more static.

average distance, they are creating shortcuts in the network. Thus, the idea in [35] is to start with a circulant network (see figure 2.8(c)), similar to the 1-lattice, and to create shortcuts by adding a double step network (see figure 2.8(d)). Since shortcuts are *added*, nodes having a higher degree than average are clearly the ones providing the shortcuts: this hint can be exploited in simulations, which would lead to artificially optimistic results. Thus, the degree is normalized by rewirings. The base networks are formalized by Definitions 5 and 6, and their combination by definition 7.

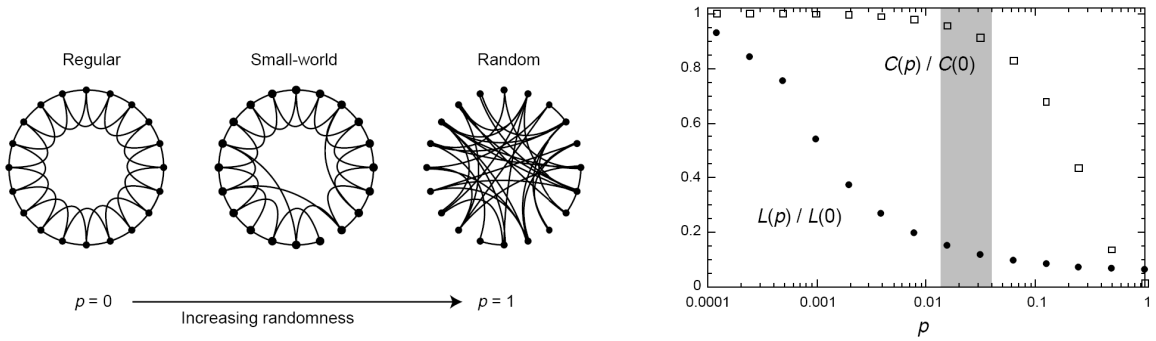

Figure 2.9: Ideal values of p to obtain the small-world effect in the WS model.

Definition 5. The circulant network $C_{n,\Delta}$, Δ even, has n nodes labeled by the integers modulo n. Each vertex i has Δ neighbours $i \pm 1, i \pm 2, \ldots, i \pm \frac{\Delta}{2} \pmod{n}$.

Definition 6. A *double step network* $C(h; a, b)$ is a circulant graph with h nodes such that each node i is connected to $i \pm a \pmod{h}, i \pm b \pmod{h}$.

Definition 7. The network $C_{n,\Delta,h}$ is based on the circulant network $C_{n,\Delta}$ in which shortcuts are provided by selected h equally-spaced nodes and interconnecting them with a *double step graph* $C(h; a, b)$. For each node i, we normalize its degree to Δ by removing the links $(i, i \pm (\frac{\delta}{2} - 1))$ and $(i, i \pm (\frac{\delta}{2} - 2))$, and reconnecting them as in figure 2.10.

The diameter (*i.e.* largest distance between two nodes) of $C_{n,\Delta}$ is reduced to D if we use a double step network with $h \approx \frac{2 \times n}{\delta \times (D - D_h)}$. The original clustering coefficient of $C_{n,\Delta}$ is $C \approx 0.75$, and the reconnection scheme reduces it by $\frac{12 \times h}{n \times \Delta}$, which is small as $n >> h$. Thus, the clustering and distance can be controlled precisely by this model. Furthermore, analyzes showed that they are better than in the WS model.

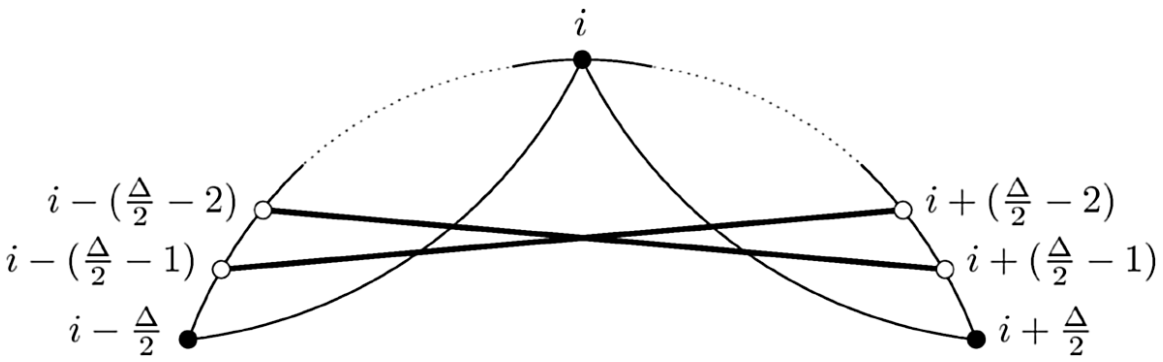

Figure 2.10: Reconnection scheme in $C_{n,\Delta,h}$ to normalize the degree of all nodes to Δ.

Deterministic scale-free model

A deterministic construction was proposed in [119], using a hierarchical structure. The authors start with K_5, in which one node is considered to be the *root*. Then, at each iteration, the network is duplicated 4 times, and the root is connected to all other nodes but the duplicated roots. The scheme can be seen in figure 2.11(a-c) from [119]. Experiments showed the scale-free effect. While the distance is small, there is no small-world effect because the clustering coefficient of a node s is $C(s)\ d(s)^{-1}$, which is too low. This hierarchical technique was generalized in [13] by the family $H_{n,t}$, in which we start with the complete graph K_n and create t hierarchies; a minor change consisted of connecting all the roots of the new duplicates in a complete graph. The scheme can be seen in figure 2.11(d-f) from [13]. It was shown through analytic work that "the number of vertices with a given degree z, $N_{n,t}(z)$, decreases as a power of the degree z and therefore the graph is scale-free", the clustering is still $C(s)\ d(s)^{-1}$ and the diameter is $D = 2 \times t - 1$.

Deterministic small-world and scale-free model

The family $K_{n,t}$ was proposed in [33] to generate networks that are both small-world and scale-free[7]. As in the previous section, we start with the complete graph K_n and iterate up to t. For each iteration, a node is added for each subnetwork K_n and connected to all nodes of this K_n; the principle of the iteration process is shown in figure 2.12(a-b) from [32], and an example on $K_{3,2}$ is shown in figure 2.12(c) from [33]. It was proven in [33] that the

[7]The family was denoted by $K_{q,t}$ in [33] and $K_{d,t}$ in [32]. Here, we denote it $K_{n,t}$ to highlight the similarities in construction with $H_{n,t}$ defined in the previous section.

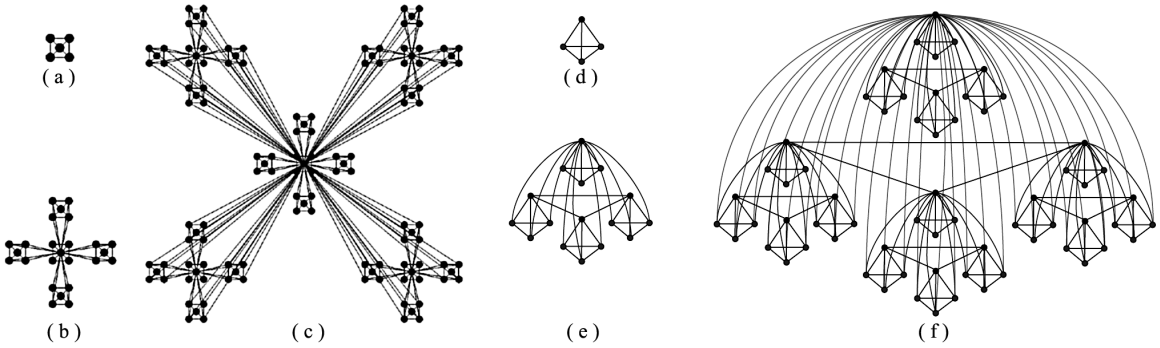

Figure 2.11: Construction of hierarchical networks on 3 levels, starting with K_5 (a-c) or K_4 (d-f).

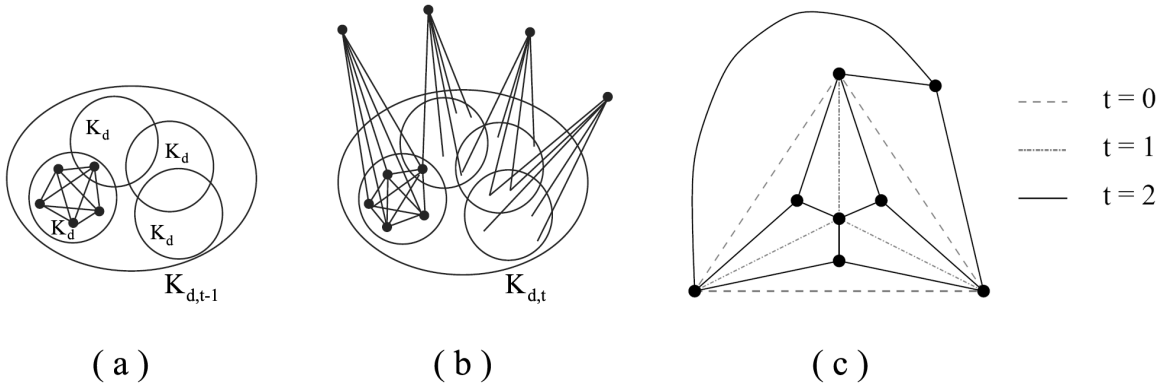

Figure 2.12: Construction of $K_{n,t}$ from $K_{n,t-1}$ (a-b). Three first steps to construct $K_{3,2}$.

clustering coefficient is high: for $t \geq 7$ and $n \geq 3$, $C \geq \frac{3\times q-2}{3\times q-1}$. For t large enough, the diameter is $D \approx \frac{2\times t}{n}$ and the exponent of the power-law is $2 < \gamma < 2.58496$.

2.3 Using complex networks in epidemiology

Through the emerging field of complex networks, more accurate modelling techniques are now available and can be applied to the study of epidemics. This application is twofold:

> ‘‘In its more passive application, modelling can aid in predicting the course of a particular epidemic, so as to plan what resources will be needed to deal with the problem. A more [active][8] role for modelling [...] is to use it to determine

[8]The original quote was of a “more agressive role for modelling”. We prefer here the distinction between passive, in which we observe the network, and active, in which we act on the network.

> the optimal policy for controlling the course of a particular epidemic by isolating or immunizing the population at appropriate times." [82]

We first present the 'passive' application. Given an epidemic model and a property, for example SIS and scale-free, we study the two main quantities of interest for epidemics: the epidemic threshold (introduced in Section 2.1) that states conditions for which a disease can become an epidemic, and the sizes of infected subnetworks, that show how the disease spreads over time. Then, we present the 'active' application dealing with strategies to counter a disease. As noted in [102]:

> ''traditional epidemiology suggests that the most important factors determining the spread of an infectious pathogen are the vulnerability of the population, the length of the infectious period and the rate of infection. These translate into three potential interventions to mitigate the threat of [viruses]: prevention, treatment and containment".

We shall not consider treatment as a solution within this thesis for the reasons found in [102]. We will instead focus on *prevention* and *containment* immunization strategies: in the former, we already possess an antivirus and wish to deploy it to the population before the disease occurs, while in the later we deploy the antivirus while the disease is already spreading; the term competition will be used instead of *containment* in order to focus on the means and techniques implied rather than on the goal. We shall also distinguish two cases imposed by constraints on the access: in a global access, we can access any set of nodes in one time step, whereas in a local access the nodes accessed at a time step are chosen among the neighbours of the nodes accessed at the previous step. Concretely, a global access allows knowledge of the whole structure of the network, which can be used to design elaborate mechanisms; furthermore, as it becomes possible to act on any set of nodes, there is complete freedom of the choice of nodes that are immunized. On the other hand, a local access imposes an incremental approach: if a set S_i of nodes is accessed at time i, then we are only allowed to access the nodes v at distance at most d from any node in S_i; note that under this definition, a global access is a local one such that d is at least the diameter of the network. The combination of global and local access with preventive and competitive settings defines the four cases that will be studied in the remainder of this section.

2.3.1 Understanding the quantities of interest

Is there an epidemic threshold in scale-free networks?

Pastor-Satorras and Vespignani have formulated the now famous claim that there is no epidemic threshold for scale-free networks. In other words, "scale-free networks are prone to the spreading and the persistence of infections, whatever virulence the infective agent might possess" [114]. Their proof relies on four assumptions:

(1) SIS epidemic model.

(2) Only the scale-free property exists, *i.e.* we assume random mixing.

(3) Exponent of the power law is $2 < \alpha \leq 3$. *In the case of* $\alpha > 4$*, the behaviour of the epidemic threshold is as in a random network.*

(4) Infinite network, *i.e.* $N \longrightarrow \infty$.

Under (1), the evolution of the density $p_k(t)$ of infected nodes with degree k is given by the following equation:

$$\frac{dp_k(t)}{dt} = -p_k(t) + \lambda \times k \times [1 - p_k(t)] \times \Theta(p(t)).$$

We consider that all previously infected nodes are healed ($-p_k(t)$). All susceptibles nodes ($1 - p_k(t)$) can be infected, with a probability proportional to the infection rate λ, their degree k, and the probability that one of their links connects to an infected node $\Theta(p(t))$. If we impose a stationary condition $\frac{dp_k(t)}{dt} = 0$, the equation yields:

$$p_k = \frac{\lambda \times k \times \Theta}{1 + \lambda \times k \times \Theta}.$$

By considering the non-trivial solution of this equation using (2), we obtain the epidemic threshold $\lambda_c = \frac{<k>}{<k^2>}$. Under (3-4), we have $< k^2 > \overset{N \to \infty}{\longrightarrow} \infty$, hence $\lambda_c = 0$. Clearly, the question now is: what happens if we remove one of the assumptions?

It was proposed in [43] to relax assumption (2) by introducing as another structural property a high clustering of $C = \frac{5}{6}$. The presence of an epidemic threshold was observed through simulations: in figure 2.13 from [43], a significant prevalence of the disease is observed only if the infection rate λ is increased above a value λ_c. Furthermore, the authors showed by an analytical approach[9] that there is a threshold $\lambda_c = \frac{1}{<k> - 1}$.

Other studies concluded that an epidemic threshold reemerges by removing assumption (4), or by removing assumption (1). In particular, it was shown in [116] that if we consider a

[9]Note that the approach of [114] is based on a BA model, while [43] uses another model to generate scale-free networks. References to the models can be found within the articles.

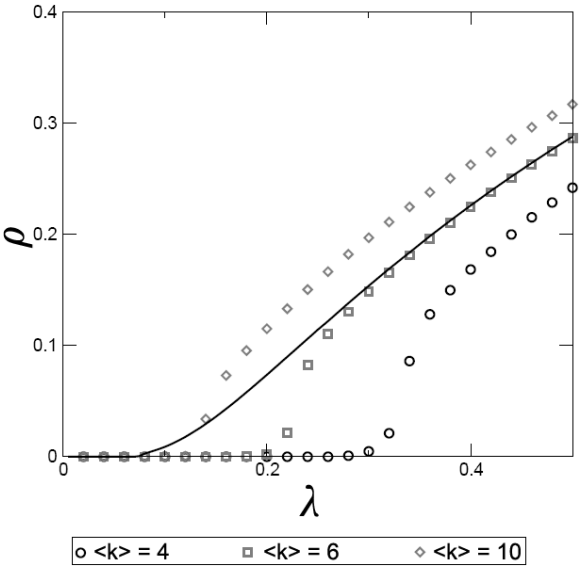

Figure 2.13: Fraction ρ of infected individuals in the stationary state as a function of the spreading rate λ in a scale-free network with high clustering. Simulations for 10^5 nodes, averaged over 100 realizations.

model "different from the standard SIS model, not only does an epidemic threshold reapper even in theoretical SFNs of infinite size but, more important, SFNs can be much less efficient than HNs in favoring the disease spread." Overall, research demonstrated that four strong assumptions are needed to lead to the non-existence of an epidemic threshold in scale-free networks. Thus, one cannot generalize that scale-free networks are more or less prone to epidemics than another type of network[10].

How do the quantities of infected individuals vary?

In an SI epidemic model, the quantity of infected individuals only increases until reaching a stationary point. However, models in which transitions are possible from the Infected state back to the Susceptible state can exhibit a wider variety of behaviours, as we saw with

[10]The very specific conditions for which the claim from [114] holds were often neglected, and even recent publications simply state that scale-free networks do not have an epidemic threshold. For example, the introduction of the special issue of *Mathematical Population Studies* on *Networks in Epidemiology* claims that "in a scale-free network there is no epidemic threshold which implies that the elimination of a [sexually transmitable disease] is not possible" [86].

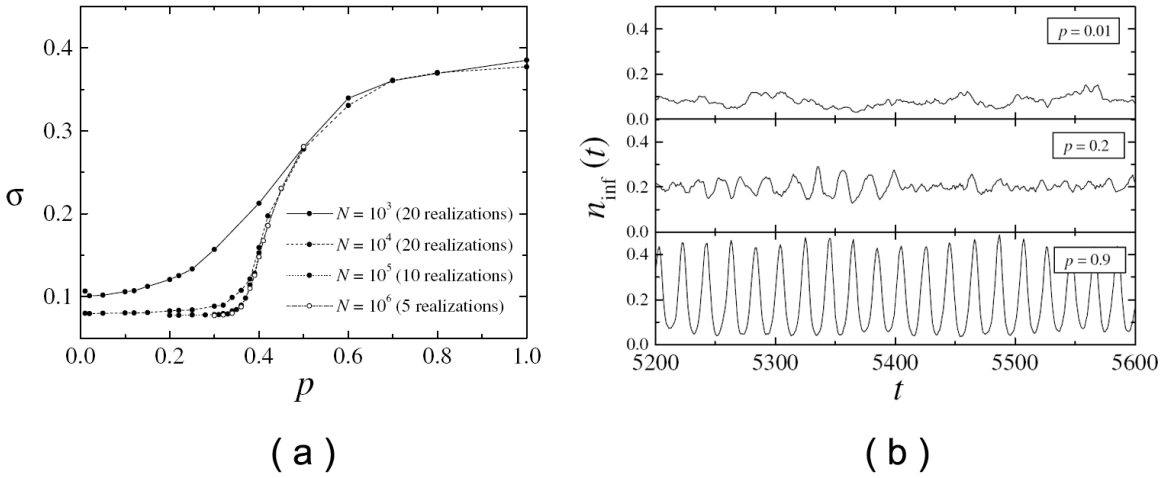

Figure 2.14: Synchronization of the system as a function of p (a). Fraction of infected elements as a function of time (b), for a network of size $N = 10^4$.

oscillations in Section 2.1. In [87], the SIR model was considered in the case of a small-world network, generated using the WS model with probability p. A node becomes infected with probability $\frac{d_{inf}(i)}{d(i)}$, where $d_{inf}(i)$ is the number of infected neighbours of i and $d(i)$ its total number of neighbours. Then, a timer is used rather than probabilities: the node remains infected until time τ_I, when it recovers with temporary acquired immunization; this immunization lasts until time τ_0, when the node becomes susceptible again. The authors studied the evolution of the fraction of infected individuals with respect to time, for different values of p (see figure 2.14(b) from [87]). While oscillations were expected, a synchronization phenomenon was also found: "The formation of persistent oscillations corresponds to a spontaneus synchronization of a significant fraction of the elements in the system. [...] They go through the disease process together, becoming ill at the same time, and recovering at the same time." It was found that the synchronization of the system does not increase smoothly with p but goes through a sharp transition for a critical value $p_c \approx 0.4$ (see figure 2.14(a) from [87]). This phenomenon of synchronization in complex networks has recently gained interested in the community. The interested reader will find the latest developments in a special issue of Chaos solely devoted to this topic [112], or an introduction to the subject in a casual form by [127].

What is the influence of the starting point of the disease?

Due to the heterogeneity in the topology, the situation in the short term is clearly different depending on where a disease starts. For example, in a simple setting a central node (*i.e.* one with minimum graph eccentricity) can infect a population faster than a node 'at the border' (*i.e.* with maximum eccentricity), but the two would lead to the same situation in the long run. Some elements to take into consideration were recently proposed in [37]. First, the authors studied the influence of the degree of the node where the infection starts, with respect to the variability of the resulting outbreaks. They found that, for some categories of scale-free networks, the higher the degree of the initial seed, the lower the variability of the outbreak: "when the seed is a hub, the number of infected [nodes] becomes rapidly very large and thus leads to smaller relative variations of the prevalence". They also showed that nodes at the same distance from the starting point can be infected faster in those scale-free networks, and proposed that "the reason for this behaviour lies in the difference of the numbers of shortest paths in these networks"; this observation matches the basic definition of *maximum flow* used to evaluate the strength of a connection in social network analysis: "one notion of how totally connected two actors are, asks how many different actors in the neighborhood of a source lead to pathways to a target" [71]. It is likely that further interesting results can be found by studying the correlation between the strength of the connection (expressed with Taylor's Influence or Hubbell and Katz cohesion) between an infected node and a susceptible one, and how fast that node becomes infected.

2.3.2 Immunization strategies

Global access and preventive setting

In this situation, we study an immunization strategy that prevents a virus from spreading by breaking the network into disjoint subnetworks, so as to minimize the size of a population in which the virus can spread. Thus, for a given network, we select nodes that will separate it into disjoint subnetworks. In terms of objective function, we have two inter-dependent quantities: we want to minimize the fraction q of nodes selected, but we also want to minimize the size f of the largest remaining subnetwork. Clearly, f is inversely proportionnal to q: the more nodes we select, the more we can divide the network, hence the smaller each subnetwork will be. In the context of epidemics, f is fixed to some small quantity, and we want to minimize the fraction q of nodes to remove to achieve it. This is a direct application

of the search for efficient *graph separators*, defined below.

Definition 8. Let $G = (V, E)$ be a graph with a set of vertices V and edges E. A *separator* is a set $V_S \subset V$ such that $V - V_S$ results in disjoint subgraphs $G_1, ..., G_n$.

Finding efficient separators is a well-known problem, with applications in numerous fields:

> "Research has shown that essentially any nontrivial notion of a graph separation decision problem is NP-complete. [...] As a consequence of this putative computational intractability of graph separation [much research] aimed at discovering tractable approaches to the problem. One well-studied direction is to seek algorithms that discover provably good separators for specific families of graphs rather than for general graphs. [...] While heuristics provide no guarantees [...], several of them have been found to be very efficient in practice." [120]

For the application to epidemics, we cannot fully specify a family of graphs, but we are aware of certain properties (degree distribution, clustering, average path length, ...) and we exploit them when designing heuristics. Early heuristics approached the problem as follow: given that we want to disconnect the network, rank the nodes by their importance with regard to the connectivity, and delete them in decreasing order. In other words, we use a measure of centrality for each node, and apply a greedy method. Up to recently, a simple *Degree Centrality* technique (DC) was used[11] and widely studied. First, it was shown that for very small values of q, this technique quickly increases the average distance[12] in a network, especially compared to a technique in which nodes are chosen randomly (see figure 2.15 adapted from [5]). While choosing nodes randomly seems far from being wise, it is worth noting that it is how immunization policies are designed: a massive fraction q of

[11] This method is known as *targeted strategy* but the name lacks rigour: by definition, all strategies are 'targeting nodes' based on some measure of centrality. Thus, this term will not be used here. Furthermore, the results have to be interpreted carefuly: scale-free networks can be generated in many ways, and a high efficiency of this strategy *on one* construction does not mean that it is efficient *on all* constructions (see [16] for the impact of micro-level differences on networks that have identical macro-level features such as degree distribution). This strategy was mainly tested on the Barabasi-Albert (BA) model, in which new nodes are added and connected to old ones with a probability proportional to their degree: as a result of the BA model, a high-degree node is very likely to be the cornerstone of the network.

[12] The authors claimed to be studying the diameter, and this conclusion was subsequently used in a number of papers. However, they define the diameter "as the average length of the shortest paths between any two nodes in the network" and thus we replaced the term 'diameter' by 'average distance'.

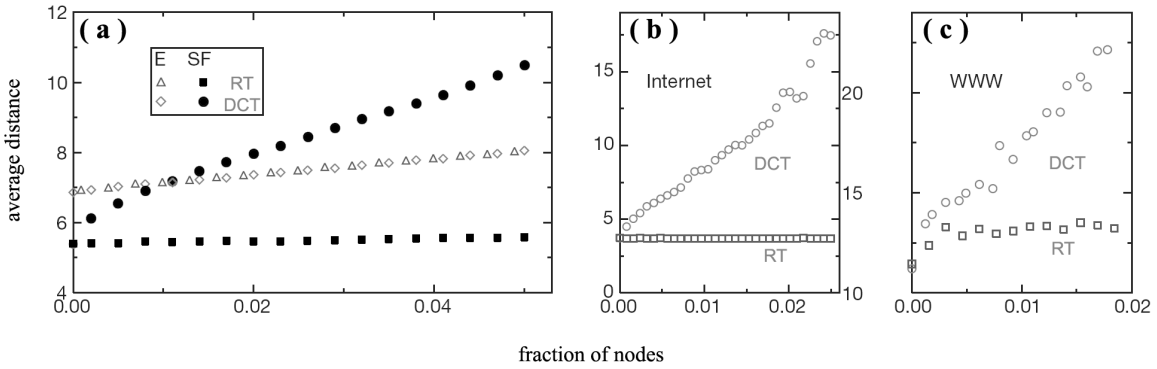

Figure 2.15: Changes in the average path length as a function of the fraction q of removed nodes, chosen by the decreasing degree technique (DCT) or randomly (RT) in random (E) and Barabasi-Albert scale-free (SF) networks (a), a topological map of the Internet (b) and the World-Wide Web (c).

the population is usually immunized, regardless of any individual's characteristics. Thus, this study suggested major changes: *if* a population network is scale-free and we can find individuals with high degree, *then* the DC technique can tremendously reduce the amount of vaccine to distribute.

Under the DC technique, a small q suffices to quickly decrease the size f of the largest subnetwork. However, it was pointed out for scale-free networks that "one can remove all vertices with degree greather than k_{max} and still have [high values of f] even for surprisingly small values of k_{max}" [24]. While the latter seems to contradict the former, they are simply looking at the phenomenon from different angles [24]: in scale-free networks, the highest difference of degree is of several orders of magnitudes, thus removing a few of the highest-degree nodes quickly reduces the highest degree in the network. For example, figure 2.16 from [24] shows that, for an exponent $\alpha = 2.7$ of the power-law, removing a fraction $q = 0.1$ reduces the highest degree to only 10.

An early alternative to the DC technique was the *Acquaintance Immunization* technique (AIT): "choose a random fraction of the nodes, look for random acquaintances with whom they are in contact" [31] and immunize the acquaintances. This technique is "less sensitive to manipulations than [DCT, because it depends] on acquaintance reports, rather than on *self-estimates* of number of contacts": if a node is lying about its degree then it can easily mislead the DCT and obtain the vaccine, whereas several nodes have to lie about a common

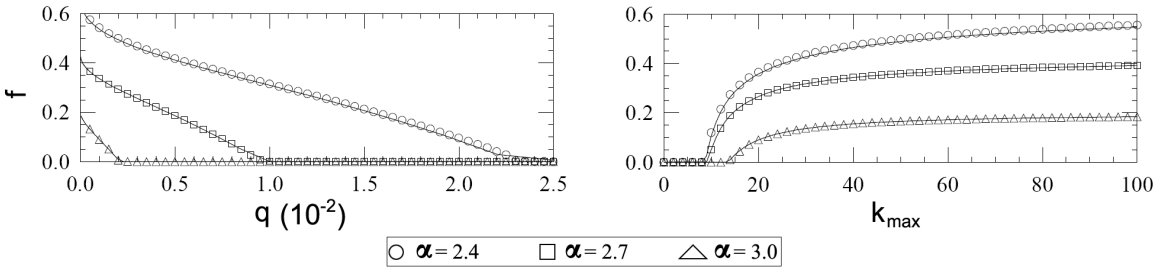

Figure 2.16: Size f of the largest subnetwork versus fraction q of nodes removed under the DC technique, and f versus the maximum degree k_{max}, in scale-free networks with exponents α. Simulation results are showed by data points and solid lines are exact solutions.

acquaintance with AIT in order to obtain the vaccine[13].

Given that this is a problem of finding the best separators in a graph, it has recently been proposed in [29] to simply apply a heuristic employed for general networks[14]. While this is a straightforward solution, it turned out that it actually performs better than the best heuristics employed so far in complex networks. In other words, all of the previous techniques failed, given that they were aiming to benefit from 'complex network properties' but were outperformed by a general heuristic. Thus, a heuristic is yet to be found that really benefits from complex networks properties. To suggest a few possibilities, we propose to look back at the nature of the problem: we want to break a network into disjoint subnetworks, and it all started by studying the effects of a greedy method selecting nodes based on their degrees. The relation is straightforward with *link removal methods* for community structure identification: for example, an algorithm in [57] is based on the calculation of betweenness centrality for all edges of the network, and then repeatedly deleting the edge with the highest betweenness and recalculating the betweenness centrality.

Thus, the literature on community structure identification through greedy algorithms complements the approaches undertaken so far, and can be used in conjunction with the

[13]Nodes that deliberately lie when queried have been investigated in the context of graph searching [62] but not yet for epidemics. However, there is a strong incentive for an individual to lie: by exploiting the right leaks in an immunization technique, one can be offered a vaccine. It can thus be of particular interest to study the robustness of a technique with respect to individual lies: in the case of DCT, a lie guarantees the acquisition of vaccine.

[14]Notice however that the article must be read with caution. While the authors claim that their technique can "separate a network into two clusters with arbitrary size ratio [...], with the number of separators minimized" and that "the computational complexity [is] found to be close to linear in N", this problem is NP-complete and thus the authors' claim of minimization is erroneous.

myriad ways of computing a centrality that have not yet been tried [18]. Furthermore, methods such as *spectral clustering* have surprisingly not been applied yet to this field, although they were outlined for their highly promising potentials in the general case of graph clustering [91]. They also offer a desirable degree of flexibility in the characterization of the subnetworks to be separated: "the basic isoperimetric problem for graphs [consists of] removing as little of the graph as possible to separate out a subset of vertices of some desired 'size', [where] the size of a subset of vertices may mean the number of vertices, the number of edges, or some other appropriate measure defined on graphs" [30]. While simple applications of such techniques may already offer valuable results, further research relating the properties found in graphs (such as the degree distribution in [75]) and the spectra of such graphs can offer a better understanding of complex networks and therefore be beneficial to the design of immunization strategies[15].

Local access and preventive setting

With the preventive setting, we are still able to select any node to receive the antivirus, but the local access constraint imposes less sophisticated techniques for the selection of such nodes. A heuristic flooding[16] was proposed in [126]: we start from a node u, designated as the *originator*, and the vaccine is forwarded to a node v with a probability given by a heuristic function $h(d(u), d(v))$; the process is then repeated, while keeping in mind the tradeoff presented previously between the fraction of immunized nodes and the fraction of nodes that could be infected if an infection occurs. If we consider a strictly local constraint with $d = 1$, then the degree is the only topological information on which the heuristic can be based: by the relaxation $d = 2$, clustering information become available, while for $d = 3$ and above local versions of centrality may be used. However, this approach is not appropriate in all situations. Firstly, different vertices may forward the vaccine to the same node: in a computer network setting, implementing a method that *asks* a node if it was already immunized can be simple while saving an important amount of bandwidth compared to the

[15] One has to be particularly careful with conjectures about the spectra. Indeed, the temptation to generalize power-laws as a universal mechanism to explain many other features [79] is also found in the study of the spectra. For example, measurements have led to the conjecture that "the power law of the degrees determines the power law of the eigenvalues [but it was shown] that we can construct a scale-free graph with non highest eivenvalue power law distribution [as well as a] regular graph with eigenvalue power law distribution" [49].

[16] 'Broadcast' is the standard terminology from a graph-theoretic perspective, to which "dissemination" and "flooding" are equivalent.

uploading of a security patch; furthermore, this would allow the vaccine to be forwarded to a node that may not receive it at all otherwise.

Secondly, we might consider that there is a higher cost for *deploying* the patch at a node than only going through it: for example, one might think of nodes as being villages through which an ambulance can drive quickly, and the deployment being the distribution of vaccines within a village with a sigificantly higher cost in time. Thus, we do not automatically forward the vaccine between nodes, but we rather *explore* the network to deploy the vaccine to the best targets: the tradeoff is then between the quality of the nodes found and the time spent in finding them. Graph exploration is a well-studied problem and recent studies on the topic as well as references related to the formalism can be found in [54]. Furthermore, we can consider that the graph is being explored by *agents*, which introduces a shift in perspective from pure heuristic functions to agents with more complex features such as cooperative behavior. Indeed, while we pointed out that a strictly local constraint only allows the use of degree for the selection of a target, that can be slightly relaxed if we allow cooperative behavior: at any time step, the only *immediate* information about a node is still its degree, but more time can be spent in exploring the neighborhood to collect further information, or exchanging information by a rendez-vous mechanism between agents. As explained in [18], "it may not be possible for a vertex to compute shortest paths [necessary for many centrality indices] because of a lack of global knowledge [and thus] a random-walk model provides an alternative way of traversing the network": the conjunction of *biased random walks* [144], supporting alternative versions of betweenness centrality and closeness centrality, with cooperation between agents can allow the constraint of locality to be partially waived.

Global access and competitive setting

The example of the ambulance already introduced a shift in perspective from pure heuristic functions to include more complex agent reasoning. In the case of global access with competition, we can consider agents as *players* in a network game [50]. Indeed, an increasing number of studies have benefited from the extensive study of competition in game theory by applying it to networks. Among those, it was proven in [85] that selecting two disjoint subsets of nodes to initiate the spreads of the virus and the antivirus, in order to maximize the number of nodes reached, is NP-hard for both players; thus, heuristics are necessary. Furthermore, the authors showed by a counter-example that "in a two player [spread] game

where both players select one node to initiate their [spreads] in the graph, the first player does not always win": in other words, if the virus and the antivirus are given identical capacities and the antivirus can only be deployed after the virus has been detected, then it is not enough of an advantage for the virus to systematically win.

Local access and competitive setting

In a local access and preventive setting, we could afford to spend more time on collecting information to increase the quality of the selected target. On the other hand, the intensity of the competition in this setting tends to discourage such tradeoffs , particularly for computer networks. For example, virulent outbreaks of computer viruses have sometimes made the front page of daily newspapers, with a recent last example being the Conficker virus in January 2009, which already infected millions of computers in the space of a few weeks [108]; similarly, Code-Red is known [145] to have infected 359,000 machines in 14 hours on July 19, 2001. In computers networks, epidemics (*i.e.* worms) fighting each others have already been experienced[17]: the 'helpful worm' Welchia tried to remove the worm Blaster [47], although with undesirable side effects such as rebooting the user's computer and slowing down Microsoft's servers through the download of security patches. Such approaches have been called *active defenses* [109] and one early example, although not commonly deployed, is *predators* [132] defined as "good will mobile codes which, like viruses, travel over computer networks, and replicate and multiply themselves".

This approach is well motivated: a few servers on which security patches are available would not be able to accomodate millions of computers simultaneously, but a distributed 'spread' of software patches could support this demand (note that Welchia does not fall into this category as the patch is not embedded). However, the methods in [132] are not sufficient for the use of predators for real-world cases: they are supposed to be monitoring all packets in the network, looking for signatures of the virus, "entering [an infected] machine in the same way as the virus" and then multiplying to randomly selected machines. Indeed, not all communications can be monitored. Furthermore, a virus may close the backdoor that it exploited and thus it is not safe to assume that it can be used again. More practical assumptions are that only susceptible machines can be immunized, and that the patch is

[17] Claims such as "to the best of our knowledge no work exists on epidemics that fight each other", as formulated in [85], are thus erroneous in general.

spread in a *topological* fashion [137], finding its next targets by accessing the local information of the node (such as /etc/hosts on a Linux machine or machines running the same compromised network service within the subnetwork).

While predators were first targetting infected machines, the *Friends Protocol* [110] targets what it believes to be susceptible machines, with an additional threshold mechanism. A node sends *warnings* to its friends upon detection of the virus, and those friends can then take two actions when they have been convinced of the menace by receiving enough warnings: they block activities that have been reported as suspicious (*i.e.* become immunized) and/or broadcast the warnings to their own friends. Furthermore, a class of *removed* individuals (see Section 2.1) is included in the model, through a recovery process with acquired immunization. This approach is not limited to computer viruses: for example, it can be used in public health situations with local authorities that would start taking measures after receiving a few reports from other authorities. However, simulations were conducted only on simple topologies with a constant number of friends for each node. Similarly, mathematical models for active defenses in [109] were based on differential equations as in section 2.1 and ignored the fluctuations in the topology, although the comparison of different techniques shed light on the tradeoff between the end result and the cost of the antivirus. The Counter-measure competing strategy (CMC) [28] is essentially a simpler version of the Friends Protocol: warnings are broadcasted, adopted with a constant probability rather than through a threshold and, if adopted, get automatically broadcasted. More interesting however are the simulations for CMC on networks for which the average path length, clustering coefficient, and other properties were measured: this starts to link the effectiveness of a strategy to topological properties of networks.

Chapter 3

Modeling competing local broadcasts

In Section 2.3, we presented four approaches to immunize a network against a virus. In this chapter, we will focus on the case of local access and competitive setting. Firstly, we would like to know what matters the most for the final outcome of the virus spread. While previous studies have looked only at topological features [28, 37], we extend our study to include the influence of the design of the antivirus. Thus, we first formalize what an antivirus is, and establish a list of parameters for the virus and the antivirus in Section 3.1. Then, Section 3.2 shows how to construct networks on which the competing broadcasts will take place: in order to study the influence of their topologies, we have to normalize them in size and this required the development of a new small-world model. Finally, in Section 3.3 we present simulation software designed specifically for this situation and also equipped with modeling and analysis tools; we analyse the results of the simulations in Section 3.4 and, in contrast to most previous experiments which were based on simple correlations, we use a factorial design that includes first-order interactions between factors and thus allows a finer analysis. We find that the design of agents can result in similar average efficiency among different types of complex networks, and thus focus on improvements in the design. In particular, we present a tradeoff between memory consumption and efficiency by using mixed strategies in which partial randomness is introduced.

3.1 Parameters of the virus and the antivirus

The probability that a virus is transmited from a node v to another node w is generally abstracted as a constant β, called the *infection rate*. This constant simplifies the particular situation of the nodes, such as their health if they stand for individuals, and allows a wider range of situation to be addressed. The infection rate can be very different even within a same family of viruses. For example, across parasitic worms [115], *Choanotaenia iola* has only $\beta = 0.025$ while *Capillaria ovopunctatum* has $\beta = 0.775$; similarly, computer viruses based on similar techniques can be unnoticed (*i.e.* small infection rate) or devastating in the case of Code Red for which simulation studies [145] suggest $\beta = 0.8$. For the statistical study presented in Section 3.4, we consider two very different yet representative values of β. We consider $\beta = 0.80$ as an upper bound, as it is the highest reported single value in [115] as well as the value used to model Code Red. By approximating the extent to which a virus is infectious into equal-size categories, we obtain the classification in Table 3.1; representative, yet significantly different values, are thus given by the averages $\beta_{high} = \frac{0.48+0.64}{2} = 0.56$ and $\beta_{low} = \frac{0.16+0.32}{2} = 0.24$.

Table 3.1: Classification of virus virulence

Category	**Value of β**
Very high	$(0.64, 0.8)$
High	$(0.48, 0.64]$
Common	$(0.32, 0.48]$
Low	$(0.16, 0.32]$
Very low	$(0, 0.16]$

Viruses in health science propagate simply by following contacts between individuals, and the same goes for computer viruses using internal lists (called *topological viruses*) or, to some extent, those based on local subnet scanning (*i.e.* the first octets of the target and the initiator are the same). However, the distribution of anti-viruses is a more elaborate process. For example, a parasitic worm cannot form complex goals with other worms or study the topology of a population to spread more effectively; on the other hand, vaccines are distributed by humans who are able to carry on such reasonings. In the case of computer viruses, a vaccine is a patch that solves a software vulnerability, and is distributed by the company that owns the software; thus, it is possible for an 'intelligent' patch to communicate with other machines on which the same software is installed and obtain information

that would normally be denied to a virus. As the anti-virus is able to reason based on topological properties of its neighbourhood, it can order the neighbours to which it wants to communicate by decreasing preferences. Thus, the *strategy* of the antivirus is first specified by a ranking function, defined below.

Definition 9. A *ranking* is a function that orders a set of vertices of a graph $G = (V, E)$:

$$\textit{rank:} \quad \{v_1, \ldots, v_n\} \subset V \rightarrow [v_{\sigma(1)}, \ldots, v_{\sigma(n)}]$$

where σ is a permutation of the integers $1, 2, \ldots, n$ and square brackets indicate an ordered subset.

For example, random behaviour means that the ranking function returns a random permutation, and 'targeting the hubs' means that the ranking is based on the degree. The main motivation for a ranking function is that flooding is not always possible (due to limited bandwidth, or vaccine doses, etc.), and thus partial-flooding dictates that the antivirus chooses between targets. The percentage of neighbours that can be accessed within one time step is denoted by the constant α, and flooding is obtained if $\alpha = 1$. The rationale behind a percentage rather than a fixed quantity is that the 'bandwidth' of a node is usually proportional to its degree [138] in the case of computer networks; similarly, in a population the number of persons that an individual meets daily depends on the total number of persons he knows rather than on a fixed quantity. For $d(s)$ large enough, selecting $\alpha \times d(s)$ neighbours is equivalent to selecting $d(s)$ neighbours with a probability α for each of them. Thus, in order not to confer any advantage or disadvantage to the antivirus, low and high values for α have to be identical to β, hence $\alpha_{low} = \beta_{low} = 0.24$ and $\alpha_{high} = \beta_{high} = 0.56$.

As soon as the antivirus is constrained to partial-flooding, it will take several steps to contact all neighbours, and ideally a neighbour should not be contacted twice (given that one contact is enough to guarantee immunization). Thus, the antivirus needs to store in memory all nodes previously contacted. As memory is sometimes constrained, especially in distributed settings, the quantity of memory available and the way it is managed become an important part of a behaviour. We represent the memory as a collection of nodes denoted by M; in order to abstract details of memory managment, such as data compression, we consider two limiting cases: unbounded, or *full* memory, and missing, or *no* memory.

In a local approach, two successively visited nodes have to be neighbours of each other. Thus, the network imposes a set of *mobility* constraints, in which no jumps are allowed.

However, the knowledge that the antivirus has of the network does not have to be limited to its immediate neighborhood. For example, one can know the most prolific authors in an academic community without being in touch with them; if the aim is to spread a rumor in that community, then those prolific authors are typical targets and one should look for chains of acquaintances that lead to them. In order to encompass such situations within our framework, we use an $\ell - neighborhood$ as defined below.

Definition 10. The *ℓ-neighborhood* of u is the set of nodes at distance at most ℓ from u.

The usual restricted sight in which we can only know about the immediate neighbours corresponds to $\ell_{low} = 1$. The average path length of the networks considered is this chapter is 3.30, thus a value of $\ell = 3$ already offers direct knowledge of most of the network, and is too much to correspond to a real situation. Thus, a representative high value is the intermediate $\ell_{high} = 2$.

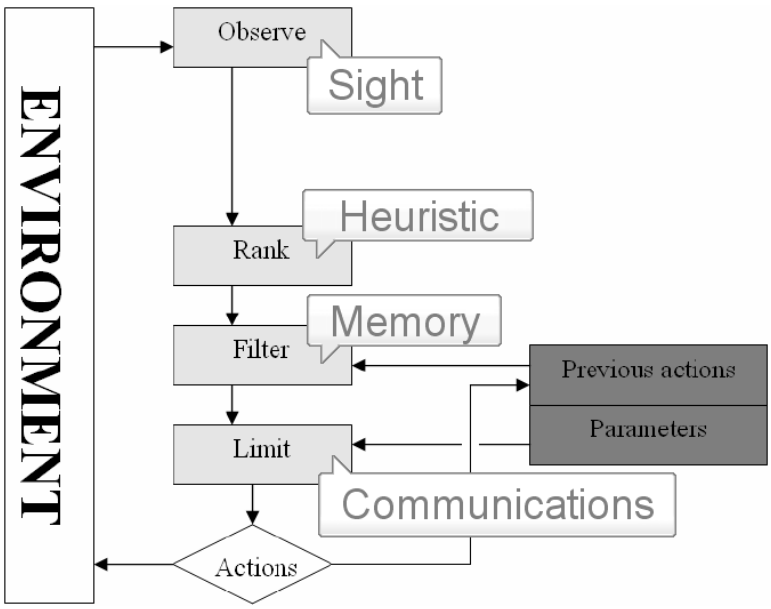

Figure 3.1: Steps of a behaviour and associated variables.

In a nutshell, a behaviour consists of the following steps: the neighbours up to distance ℓ are observed and ranked according to a heuristic. Then, they are filtered using the content of the memory M and finally a fraction x of them will be contacted where x depends on communication constraints. This is summarized in Figure 3.1 and extended in Figure 4.3, Section 4.1.

We initialize the simulation by assigning the virus and the antivirus to each control one node. The choice of those two initial nodes influences the outcome of the process, and thus has to be controlled. A neutral choice consists of choosing two nodes at random, and repeating the process several times to obtain an average. A different and representative choice found in newly developed immunization techniques (see Chapter 2 Section 2.3) consists of choosing hubs. The six parameters introduced in this section are summarized in Table 3.2. In the factorial design that we use in Section 3.4, the parameters that are varied during the experiments are called *factors*.

Table 3.2: Factors for the antivirus and the virus

Factor	**Values**
Infection rate β	{0.24,0.56}
Percentage α of neighborhood accessed by the antivirus	{0.24,0.56}
Sight ℓ of the antivirus	{1,2}
Ranking heuristic of the antivirus	Random, decreasing degree
Memory M of the antivirus	None, unbounded.
Choice of the initial nodes	Random, hubs.

3.2 Instances of networks and their properties

In order to study the interactions of strategies and topologies, we generate one instance of the network families introduced in Section 2.2.2 for each of the four possible combinations of the small-world and scale-free effects. The parameters of these instances are chosen so that they are normalized in size $N \approx 15500$ and average degree $\bar{d} = 12$ (the latter being imposed by the former in one of the models and thus used as a reference). Furthermore, we also have to ensure that instances sharing a property will express it to the same extent. For example, an instance of a hierarchical network, that is both scale-free and small-world, should have similar values of average path length and clustering to the instance with only the small-world property. However, the best possible instance obtained from the deterministic small-world construction presented previously does not compare well with the one from a hierarchical network: it has an average path length $\ell \approx 40$ and a clustering $C \approx 68\%$, versus $\ell \approx 2$ and $C \approx 89\%$ for the hierarchical network. This motivated the development of new deterministic small-world networks, with improved ℓ and C. These new models are presented below, and a summary of the properties of all four instances is provided at the end of this section.

3.2.1 Development of deterministic small-world models

Increasing the clustering

Similarly to the coefficient of transitivity which can be increased by simply connecting a new node to both ends of a edge (thus creating a triangle), it was suggested in [32] that the clustering coefficient could be increased either by replacing nodes by (small) cliques, or by connecting each node to a clique. We shall refer to the latter technique as *clustering augmentation*, and we specify it formally in Algorithm 1. The algorithm takes as input the graph $G = (V, E)$ whose clustering has to be increased, and a parameter δ specifying the degree that *all* node of the cliques must have; this constant degree is designed so that the algorithm can be applied on δ-regular graphs as described in 2.2.2 and conserve the degree regularity. By definition, each node of a clique K_δ initially has degree δ: however, one node r_i of each clique i has to be connected to a node in the original graph and thus $d(r) = \delta + 1$. In order for the degree to remain constant over all vertices, a link is deleted from r_i to another node n_i of its clique i, and each n_i is connected to exactly one copy n_j in another clique j. If the number of cliques is even, then all nodes have degree δ; otherwise, all have degree δ except one that has degree $\delta - 1$. An illustration of the algorithm is provided by figure 3.2.

Algorithm 1 *ClusteringAugmentation* : $\{G = (V, E), \delta\} \mapsto G'$

Let $A = \emptyset$
for $s \in V$ **do**
    Let $H = K_{\delta+1}$, $r \in V(H)$, $n \in V(H), n \neq r$
    $V(G) \leftarrow V(G) \uplus V(H)$
    $E(H) \leftarrow E(H) \setminus \{e_{rn}\}$
    $E(G) \leftarrow E(G) \uplus E(H) \uplus \{e_{sr}\}$
    $A \leftarrow A \uplus \{n\}$
while $|A| > 1$ **do**
    Let $v, w \in A, v \neq w$
    $E(G) \leftarrow E(G) \uplus \{e_{vw}\}$
    $A \leftarrow A \setminus \{v\} \setminus \{w\}$
return G

Applied to the deterministic small-world model presented in 2.2.2, this technique increases the clustering from $C \approx 68\%$ to $C \approx 94\%$. Having dense communities also reduces the average path length from $\ell \approx 40$ to $\ell \approx 14$: indeed, the same number of nodes is now

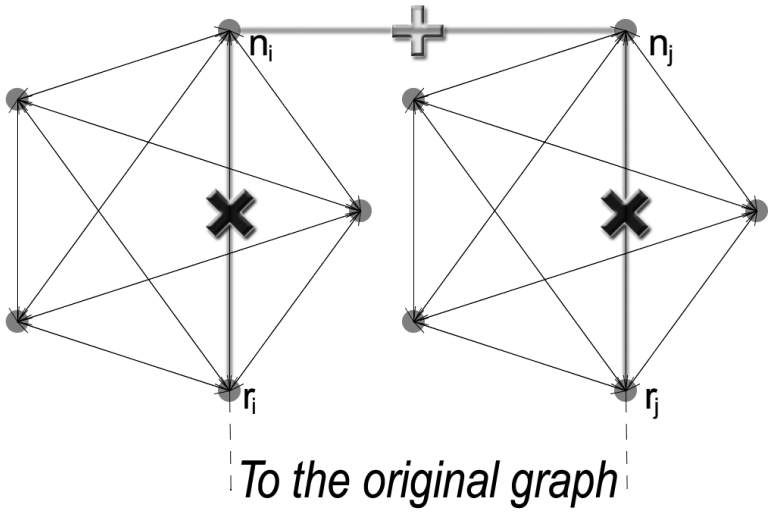

Figure 3.2: Two cliques i and j, each connected to the original graph through nodes r_i and r_j. A link is removed from r_i to an n_i, and similarly from an r_j to an n_j. A link is added between r_i and r_j.

obtained by starting with a smaller initial graph to which the cliques are added.

Decreasing the average path length

To obtain a regular degree in the deterministic small-world model, a reconnection scheme depicted in figure 2.10 was used for all nodes i providing shortcuts: the links $(i, i \pm (\frac{\delta}{2} - 1))$ and $(i, i \pm (\frac{\delta}{2} - 2))$ were removed and rewired. First, we notice that this network already provides many short-range links, as it is based on a circulant network, and that this rewiring process results in short-ranges links as well. Furthermore, what the double step network provides is medium-range links, thus the resulting network lacks long-range links. We create such missing links by using all shortcut vertices for which a link is deleted. In order to do so, we store all shortcut vertices for which we delete a link into a list L. Then, we create a link $e_{i,i+\frac{size(L)}{2}[\text{mod L}]}$ for all nodes $i \in L$. This link is guaranteed to connect nodes the farthest apart as the offset $\frac{size(L)}{2}$ corresponds to a node diametrically opposed in the underlying circulant structure. Using this technique, the average path length decreases from $\ell \approx 14$ to $\ell \approx 9.5$.

In order to reduce further the average path length, we have to provide better coverage of medium-range links. Since all of them are provided by the double step network, we

use alternating distances between two nodes of this network instead of a constant spacing. Thus, instead of connecting each shortcut vertex i to $\{i \pm a\}$ for a given constant a, we alternate between $\{i-a, i+b\}$ and $\{i-b, i+a\}$ for two given constants a and b. All possible combinations of a and b were explored, and for each of them the average path length l was recorded; the result is displayed in figure 3.3, and the best value $l \approx 5.2$ is obtained for $\{a = 33, b = 67\}$.

A new model

Based on the findings described in the two previous sections, a new model was designed and is formally defined by Algorithm 2. We start with a cycle C_n, which is the regular structure for which the network is completely connected with a minimum degree for each node. Then, links providing shortcuts are added progressively from short-range to long-range. Finally, the clustering is increased by applying the augmentation process from Algorithm 1. The resulting network is not $\delta-$regular, but the average degree is kept close to δ: line 4 ensures that no more shortcuts will be started from a node that is already saturated, and the clustering augmentation process will add a large number of nodes with degree δ, which brings the average closer. This construction provided an average path length $l \approx 3.77$ and a coefficient of clustering $C \approx 0.91$.

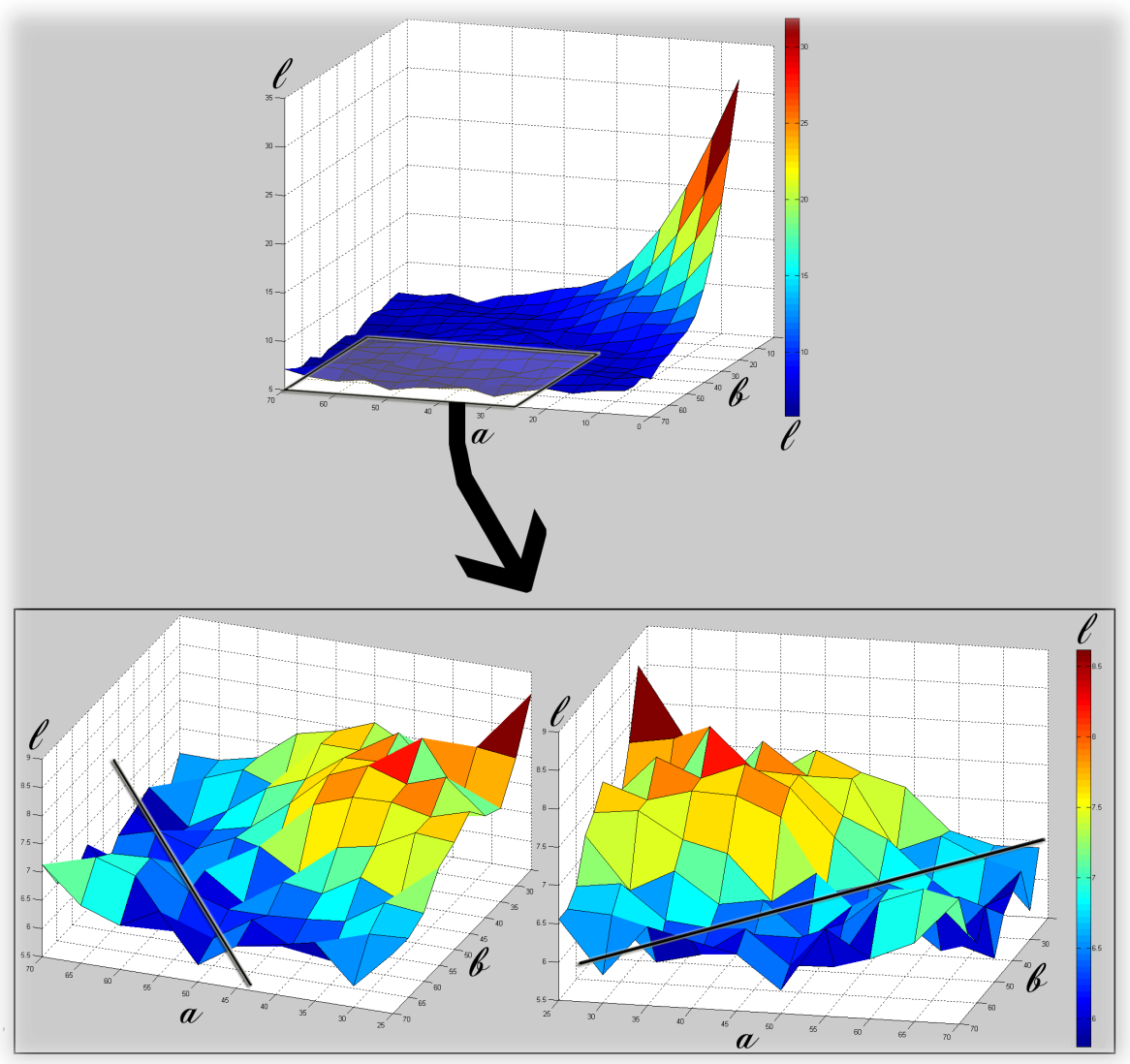

Figure 3.3: Values of the alternating spacing a and b, and resulting average path distance ℓ. The two bottom figures are different views from the part of the plot offering the best values of ℓ. The minimum is showed by a solid dark line.

Algorithm 2 $SmallWorld : \{n, \delta\} \mapsto G = \{V, E\}$

Let $G \leftarrow C_n$
for $i = 1..n$ **do**
    Let $k \leftarrow 2$
    while $d(s_i \in V) \neq \delta - 1$ **do**
        Let $j \equiv i + 2^k[n]$
        $k \leftarrow k + 1$
        $E \leftarrow E \uplus \{s_i s_j\}$
$G \leftarrow ClusteringAugmentation(G, \delta)$
return G

3.2.2 Summary of properties

The overall features of each instance are summarized in table 3.3. The parameters were chosen to obtain similar sizes and average degrees, and the models were chosen so that the values characterizing the same property are close. Indeed, the purely small-world model was designed to obtain a clustering coefficient close to the one of the small-world and scale-free model, as well as obtaining an improvement of the average path length given the limits that no node can be used as a hub (to avoid the emergence of a scale-free effect).

Table 3.3: Properties of instances

Small-world	**Scale-Free**	**Model**	**Parameters**	**Size**	$\bar{d}$	ℓ	C
		$G_{N,p}$	$N \leftarrow 15500, p \leftarrow 0.00039$	15500	12.12	4.13	0
X		Alg. 2	$n \leftarrow 1128, \delta \leftarrow 12$	15792	12.07	3.77	0.91
	X	$H_{n,t}$	$n \leftarrow 6, t \leftarrow 4$	15246	11.99	3.31	0.06
X	X	$K_{n,t}$	$n \leftarrow 5, t \leftarrow 6$	15626	12.19	2	0.889

Table 3.4: Number of nodes per degree class in scale-free instances

network type	**4-11**	**21-25**	**117-199**	**613-614**	**726**	**774-780**	**3109+**
Scale-Free	15120	0	0	0	112	14	0
Scale-Free & Small-World	15000	500	100	20	0	0	5

The features of these models are detailed in Figure 3.4 (average path length), Figure 3.5 (degree distribution= and Figure 3.6. For the sake of clarity, the degree distribution of the two scale-free networks is summarized in table 3.4.

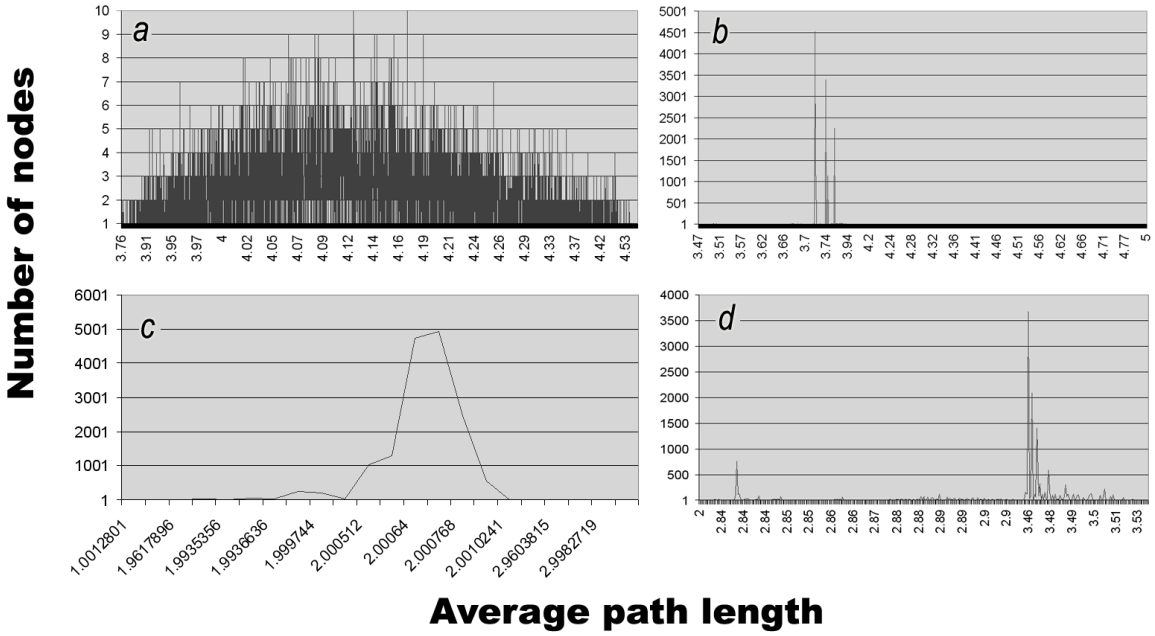

Figure 3.4: Nodes per average path length for the random network (a), small-world network (b), scale-free network (c), small-world and scale-free network (d).

3.3 Simulation Software

3.3.1 Components and goals

In order to carry on the simulations, software was programmed in Java. While the choice of a language is always a matter of personal taste, one of the reasons that we opted for Java was the simplicity of loading classes at runtime. In other words, one may try a behaviour such as *targetting the hubs with a sight of 2* and decide to make modifications without having to restart the software and thus regenerate the graph. Furthermore, new behaviours can be written as a simple extention of other ones and will be loaded by the software as it starts. As shown on the diagram in figure 3.7, the behaviours are an important component as they define the processes taking place on the graph. The graph itself is defined in a separate package, stored as a list, and remains the same throughout the simulation. A simulation consists of a set of steps in which two lists, containing infected and immunized vertices, are expanded based on their respective behaviours over the graph. The fraction of immunized vertices at each step is returned as a raw value in the *results* tab and can be copied into spreadsheet software such as Excel for further analysis and various plots. As

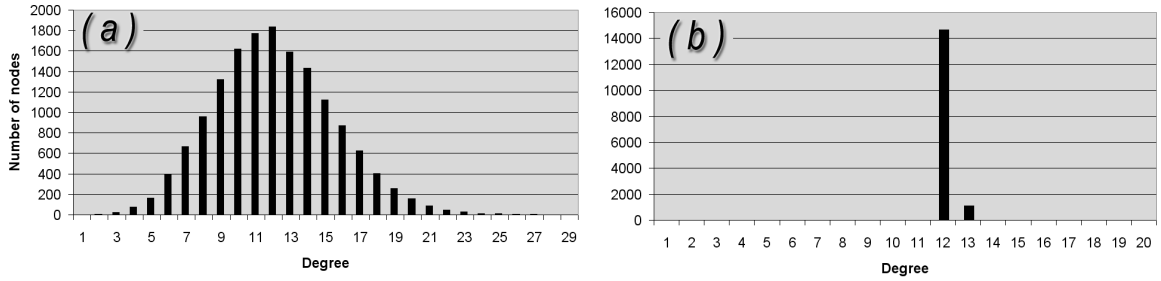

Figure 3.5: Degree distribution for the random network (a) and the small-world network (b).

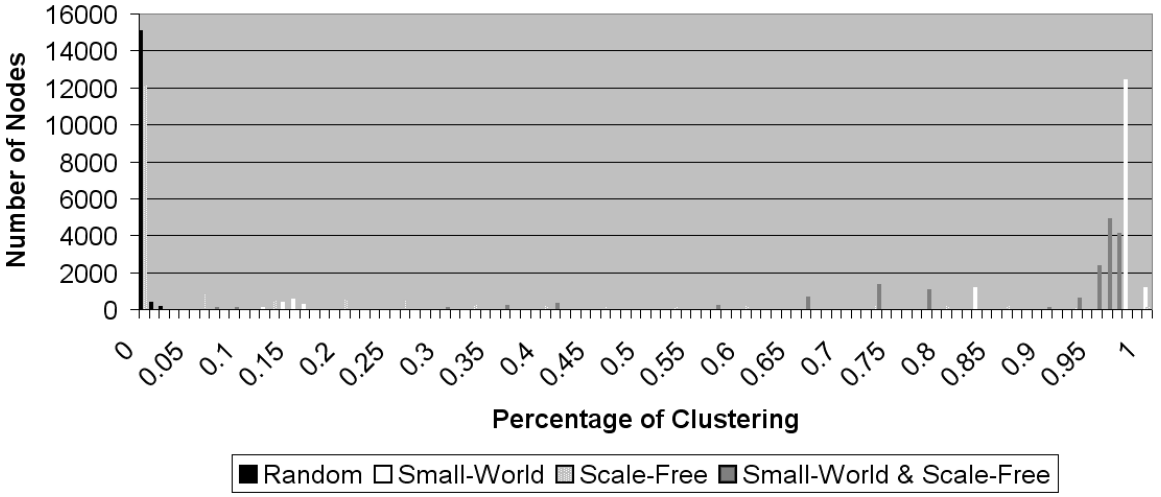

Figure 3.6: Number of nodes per percentage of clustering for each instance.

shown in Figures 3.8 and 3.9, the parameters for the simulation are available in one tab and complementary parameters can be used to specify dynamic aspects of the network which will be discussed in chapter 4 along with agent features. The progression of the spreads is shown by default for each step, which can be useful for educational purposes or demonstrations. As the animations slow down the process, they are usually deactivated when doing simulations purely for their results.

In addition to be *simulation*, the software was also designed to support modeling and analysis. All classical tools are available for the modeling aspects: adding and removing vertices, adding and removing edges, moving vertices, inverting the directions of edges or making the graph undirected; they are complmented by a few more general tools to automatically create paths, cycles, complete graphs, and n-ary trees. Up to ten different graph

models can be generated, among which random graphs, hierarchical complete graphs, the Watts & Strogatz model, recursive clique trees, grid graphs, toroidal meshes, and our model for small-world graphs presented in the previous section. Graphs can also be gathered automatically from webpages through a crawler, saved and loaded; in this last case, crawls in massive graphs can lead to a 'boundary' situation in which only a small core of vertices has been explored but most of the neighborhood is unvisited, and thus we only have incoming edges toward these boundary vertices. For analysis purposes, one might only be interested in the properties of the visited vertices only, thus we offer the user the possibility of compacting the graph: a sink vertex is created to which all vertices of degree 0 are mapped, and all edges are redirected toward this sink. Using this technique, we also succeeded to load a sample of around 400 000 vertices on a machine with 1 Gb of memory, with the small cost of 2 minutes for preprocessing.

The available analysis features can be seen in figure 3.9. The clustering coefficient and degree distribution are among the most fundamental measures, as they are required to assert the presence of small-world and scale-free properties. The average path length is implemented in a *path analysis* tool, that also provides the diameter, the highest average path length, the distribution of average path lengths, and the fractions of the graph that can be accessed from each vertex; that last measurement is particularly useful in the case of directed graphs as it might tell us that, starting from some vertices, there are parts of the graphs that simply cannot be reached and thus it can be taken into account when analyzing the potential of a spread. Centrality analysis will provide a score corresponding to a measure of centrality for each vertex and will rank the vertices; this will be exploited in the next chapter. Finally, the cycle length distribution allows all cycles up to a given length k to be listed. It is also worth mentioning that layout comes in handy when trying to analyze a network, and several were implemented: in figure 3.8, the vertices are gathered into concentric circles, and several layout of a same scale-free graph are compared in Figure 3.10.

3.3.2 Suggested improvements

With the increased interest in social network analysis, there now exists a plethora of libraries that can complement our Java software. Among them, JUNG (**J**ava **U**niversal **N**etwork/**G**raph Framework) offers complementary layouts (such as the force directed Kamada-Kawai and Fruchterman-Reingold algorithms) and metrics (such as pagerank). Metrics such as pagerank are different from the ones we have implemented because ours are based on

distances whereas pagerank is based on *feedback*; the formulas for such measures are given in [19], together with an algorithm for spectral layout. The *visione* software, specified in [19], cannot be used directly in our case because it is implemented in C++ and its sources are not available. Nevertheless, it suggests other tools for graph modeling, such as transformations for links and the ability to group nodes into selections. Pajek, developed in 1996, also provides ideas as two of its distinctive features are to enable comparison/composition of several graphs and to focus on decompositions: clusters can be found according to different criteria, and *islands* can be analyzed; the latter is defined by assigning a (not necessarily unique) number to each vertex, and considering vertices under a given threshold to be submerged.

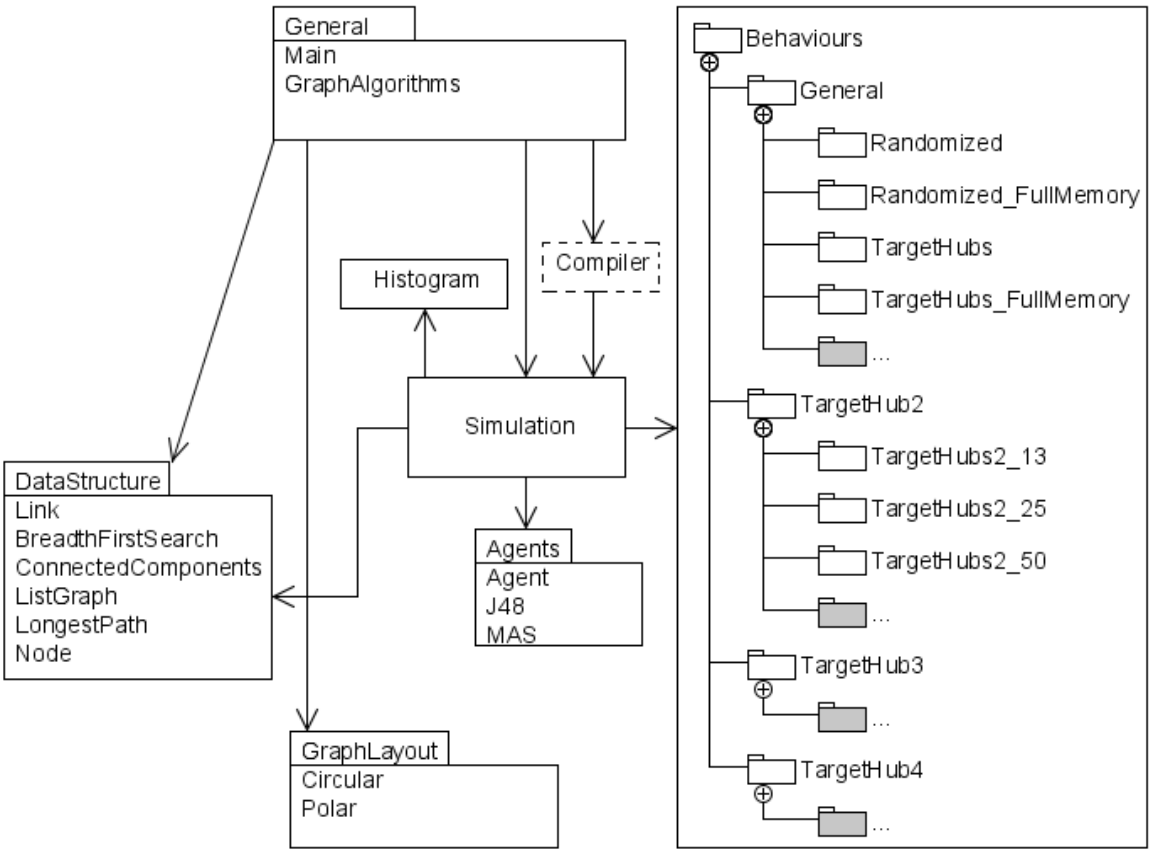

Figure 3.7: Diagram of the simulator, in which arrows pointing to a package indicates a "used-by" relation.

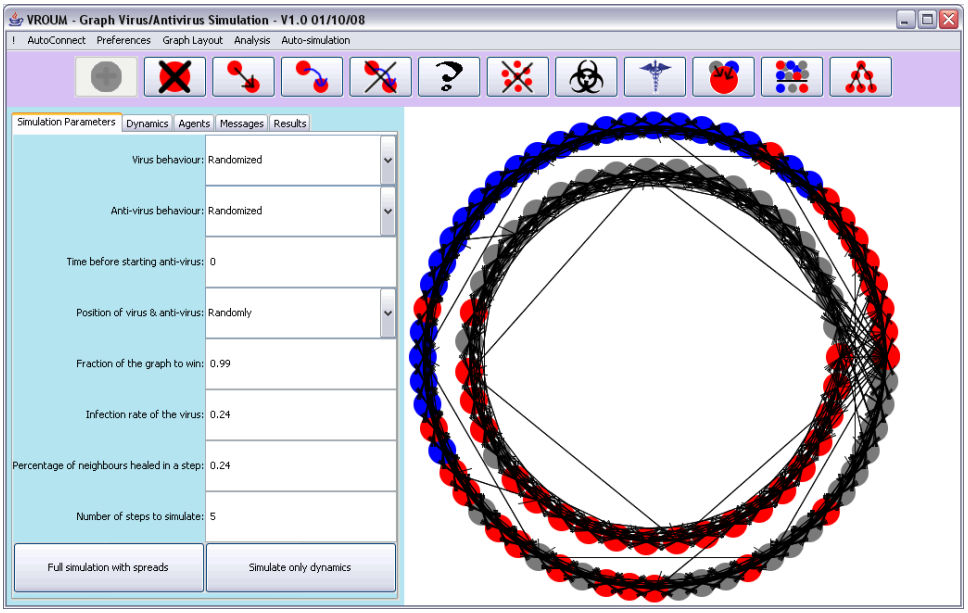

Figure 3.8: Interface showing the parameters for a simulation and three types of vertices in a circulant graph: grey (infected), blue (immunized) and red (susceptible).

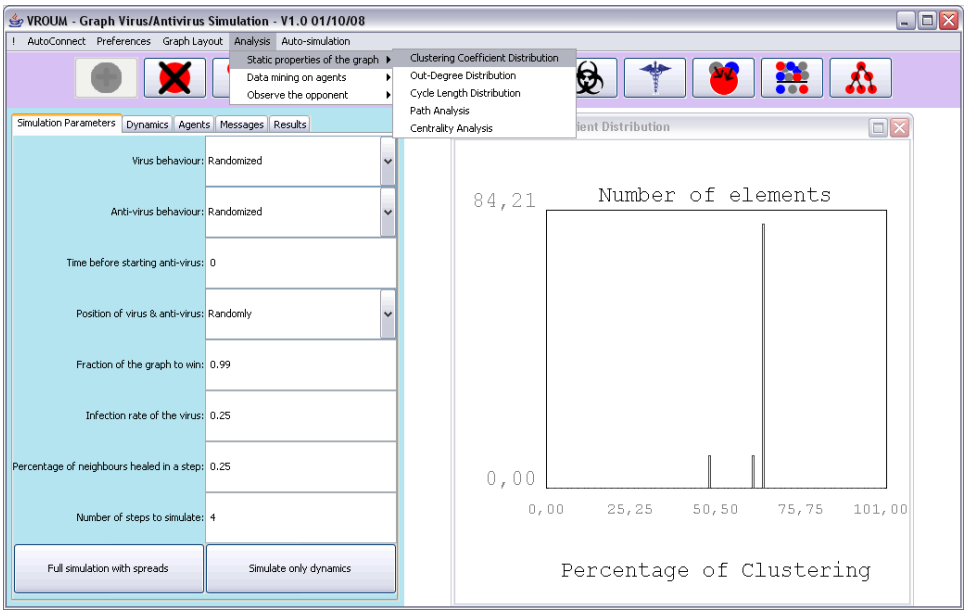

Figure 3.9: Interface showing the parameters for the dynamics of the network and the distribution of clustering coefficient.

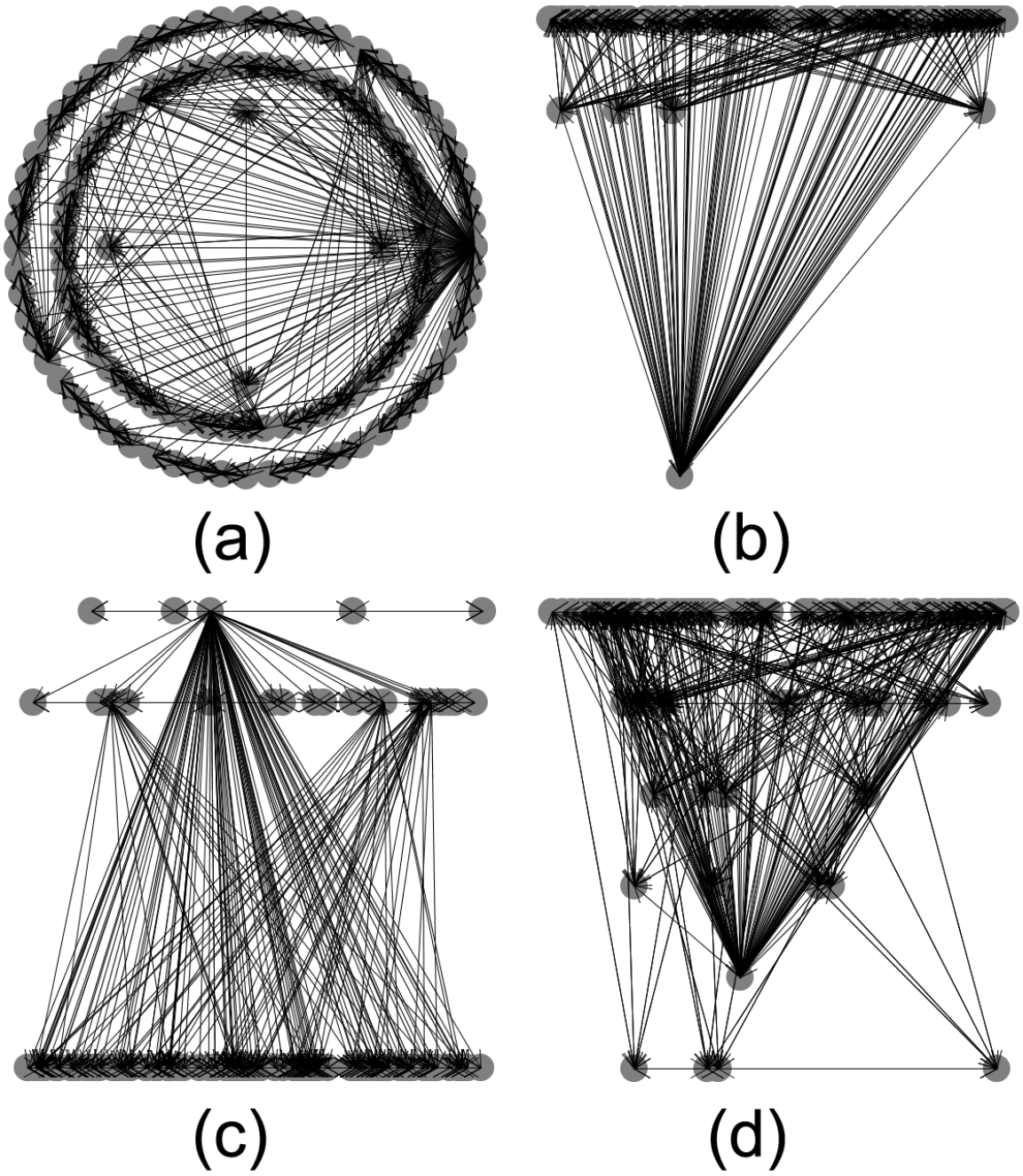

Figure 3.10: Four drawings of a hierarchical graph on 3 levels starting with K_5: concentric circles (a), layered by degree (b), layered by closeness (c) and layared by betweenness (d).

3.4 Impact of the factors

As specified in Section 3.1, there are 6 parameters for the virus and the antivirus, and 2 additional binary parameters for the presence of small-world and scale-free effects. As we are using a 2^k factorial design [70], we have a base of $2^8 = 256$ experiments. Each was performed five times in order to account for variability, hence we have 1280 experiments in total. The simulation results are summarized in Table 3.5 below. The percentage of errors in the table can be thought of as the variability between different runs with the same parameter values, and it shows that the variability remains high with five runs. This is not surprising as the behaviour of the virus is random, the networks heterogeneous, and in half of the cases the initial positions are also random. A convergence toward a more representative average is likely to be obtained for a high number of runs, but this remains challenging as some experiments require significant computing time. Indeed, even on an Intel Xeon CPU X5355 with 2.66GHz, an experiment with targetting heuristics and sight 2 on a scale-free network can require a whole day to complete. However, the same methodology may be used with future machines to obtain a more accurate picture. Despite the variability, it is clear that the communication rates α and β, and the heuristic, play a major role whereas the topology has a moderate impact. This suggests that the design of agents can result in similar average efficiencies among different types of complex networks. Indeed, almost 25% of the result is due to the design of the agents.

Table 3.5: Factors for an Antivirus

Factors	**Percentage**
Primary effects	
β	19.52%
α	15.54%
Heuristic	6.91%
Initial positions	4.45%
Memory	3.37%
Effects of first-order interactions	
Small-world, heuristic	8.80%
Memory, sight	5.58%
Small-world, scale-free	2.13%
Other effects	
Total of other effects	15.17%
Errors	18.53%

3.5 Limited randomness for memory efficient strategies

Having more memory in an agent is always a benefit, since it reduces the likeliness that it wastes time communicating with neighbours that have already been contacted. The experiments showed that the magnitude of this benefit can be significant. For example, in a random network in which the antivirus and the virus agents both start with random positions, have the same rate $\alpha = \beta = 0.24$, and use random heuristics, the antivirus succeeded to win 48.5% of the network on average, *i.e.* neither the virus nor the antivirus has an advantage. If we only give full memory to the antivirus, then its performance rose to 73.6%. Similarly, in a scale-free network with the same setting, the antivirus succeeded to win 40% on average, which rose to 74% when given full memory. Memory is even more necessary for completely deterministic strategies, as the likeliness that communications are wasted is 100% otherwise: the neighbours contacted at each step will always be the same ones that were contacted initially. Indeed, in a random network with random positions and the same rate $\alpha = \beta = 0.24$, if the antivirus always targeted neighbours with highest degree then it only took over 4% of the network, but it rose to 66.8% if provided with full memory; in other words, if we do not have memory then the performances are insignificant, and otherwise they are almost as good as the random approach that best fits this type of network. In a scale-free network with the same setting, we observed that the purely targetting approach also performed poorly at 0.46% but did extremely well at 85.2% with full memory.

Memory is one of the more costly ressources required by an agent. For example, in embedded systems such as sensor networks, the memory is limited and should be managed carefully. On the other hand, the computations are relatively inexpensive: we have generated the list of neighbours to contact either randomly or by sorting by degree[1]. A tradeoff is thus desired between the efficiency that targetted approaches can achieve in some networks versus the quantity of memory required.

[1]Note that the ratio between the cost in space of an approach *without* memory versus an approach *with* memory is not a constant but instead depends on the approach. For example, if the heuristic is random, then we only need space for the list of k neighbours to contact. On the other hand, a strategy purely based on degree requires the neighbours to be sorted thus we need space n for all of them. Providing full memory means that either we have a boolean array of all neighbours indicating whether or not they have been contacted, or we progressively create a list of all contacted neighbours. Ultimately, this structure will reach the size n of the neighborhood, thus we have a ratio $n/k, k << n$ for random strategies versus $n/n = 1$ for targetted ones (*i.e.* only twice as much memory).

In order to explore this tradeoff, we experimented with *mixed strategies*, in which a fraction p of the neighbours is chosen according to the targetting approach, $1-p$ is chosen randomly, and no memory is used[2]. In order to focus on the design of agents, we did not confer any advantage either to the virus or to the antivirus hence the same communication rates $\alpha = \beta = 0.24$; furthermore, they should not have advantages through initial positions, thus these are chosen randomly and the overall result is averaged over five run. The instances for the experiments are summarized in Table 3.6 and the results are shown in Figure 3.11, in which the fraction 0.5 is highlighted as it corresponds to the threshold above which the antivirus starts being better than a random strategy. The results exhibit high variability with the same sight, and there is a discrepancy between different sights. However, it is clear that when the sight is 1, introducing randomness always improves the outcome for the antivirus, and a significant improvement is also found for most cases when the sight is 2. There are also common values of p for which the mixed strategies with sight 1 and 2 achieve their maximum in all three configurations: for the sight 1, we have $p = 0.13$ (thus very high randomness), and for the sight 2, we have $p = 0.75$ (thus low randomness).

Table 3.6: Properties of instances

Network type	**Model**	**Parameters**	**Size**	$\bar{d}$	ℓ	C
Small-world	Alg. 2	$n \leftarrow 200, \delta \leftarrow 8$	4000	8.11	5.23	0.86
Scale-free	$H_{n,t}$	$n \leftarrow 4, t \leftarrow 4$	4690	7.99	3.24	0.08
Small-world and scale-free	$K_{n,t}$	$n \leftarrow 4, t \leftarrow 6$	4096	10.5	2.004	0.86

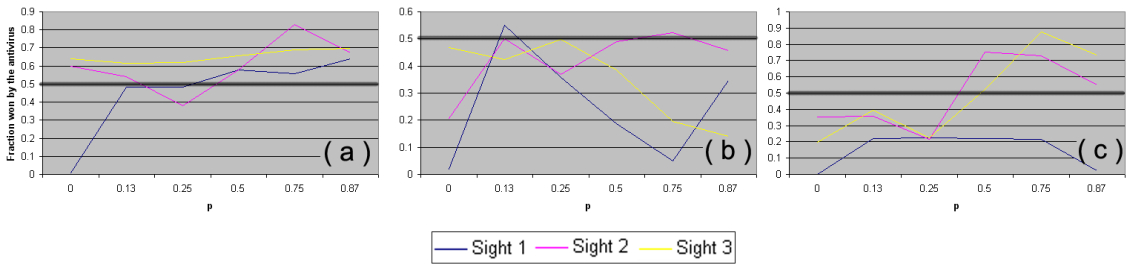

Figure 3.11: Fraction of nodes won by the antivirus when a fraction p of the neighbours are chosen by decreasing degree and a fraction $1-p$ randomly, on small-world (a), small-world and scale-free (b), and scale-free (c) networks.

[2]There is no extra memory except for the strategy itself. Thus, if p is large enough, we should still rank all neighbours and there is a need for space n, but we also save space n because we do not need a data structure to indicate whether a neighbour has been contacted or not.

Chapter 4

Dynamics and cooperative agents

In Section 2.3, we suggested the consideration of agents rather than pure heuristic functions. We concluded in the previous chapter that the main aspects we could influence in the design of an agent are its embedded heuristic and the managment of its memory. However, two aspects of the situation were simplified. Firstly, we only considered static networks, *i.e.* the whole topology is fixed at the beginning. Secondly, we considered an agent to be a *passive* and *solitary* entity: not only did an agent not learn from its environment, or its interactions with the opponents, but it did not attempt to communicate or synchronize with other agents. In this chapter, we will consider the situation as a whole by introducing dynamic networks and agents that try to learn those dynamics through data mining and cooperation.

In Section 4.1 we give an intuitive definition of agents, showing that they are far from being limited to the aspects considered so far, and we present the concept of *multiagent systems* (MAS) with which we will be working from there on; we conclude with the architecture of our system. In Section 4.2, we introduce the idea of an agent applying data mining techniques to learn, and present the dynamics of the network as the application of interest. Finally, we conclude in Section 4.3 with the techniques involved in this specific learning process, suggesting an exchange of data between agents and a process that avoids interference. Note that while no new technique is introduced in this chapter, it is the first time that cooperating agents designed to learn the dynamics of a network to irdero to immunize it are considered. Furthermore, we state the basic requirements of this learning before introducing a new practical technique that can realize it in the next chapter.

4.1 From a single agent to MAS

In the most general definition, an agent is "a computer system that is situated in some environment, and that is capable of autonomous action in this environment, in order to meet its design objectives" [142]. Being capable of an action means that it can "perceive its environment through sensors and act upon that environment through actuators" [122]. However, this would only define a dummy entity that would be going around repeating a fixed task, which is what we considered so far. In addition to autonomy and interactivity, we want agents to possess *intelligence* [130]:

- *Autonomy.* An agent does not need interventions from outside sources such as a human. Thus, it is required for an agent to be self-sufficient.
- *Interactivity.* An agent cannot be entirely controlled by an outside source but may interact with other agents or its environment. In particular, an agent is *reactive* when it perceives the changes in its environments and can react accordingly, and *pro-active* when it takes initiatives to change its environment to accomplish certain goals.
- *Computational intelligence.* An agent may use its interactions with the environment and/or other agents to evolve, which can be considered as its 'intelligence'. If the agent is trained over a given process, we speak of a *learning* agent. If it builds a strategy with other agents in order to reach its goal, the agent is *cooperative*; respectively, there is *competitiveness* when the agent competes with other ones.

Example 1: Monitoring robot. A robot keeping a house has cameras as sensors and wheels as actuators. It is self-sufficient in the sense that, once launched, it does not normally need interventions other than to halt it. It is reactive if it moves toward intruders when detected. It can furthermore be cooperative by developing a plan with other robots to catch the intruder.

Example 2: Computer virus. A computer virus propagating by scanning a range of IP addresses uses the results of the scan as a sensor, and propagation (via uploading and executing) as actuators.

Example 3: Mining robot. A mining robot [113] can be as similar to a monitoring robot with the extra capability of being trained toward more efficient behaviours. For example, it can be rewarded according to the value of the samples it brings back, and it will modify its behaviour accordingly: if it receives little reward, it might decide to switch

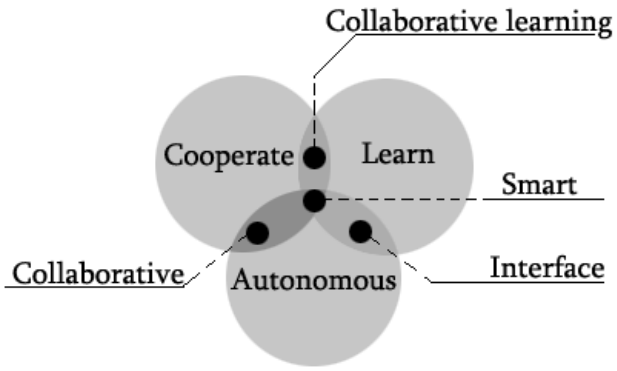

Figure 4.1: Nwana's classification of MAS.

to another place, while it would have an incentive to stay at a place as long as it brings a satisfactory reward. Thus, it follows its own learning process.

Systems with a single agent are studied in Artificial Intelligence (AI) and already show great benefits as is evident by observing the numerous applications of robotics. Our system is more complex as it consists of several agents, and is thus referred to as a *multiagent system* (MAS); the corresponding subfield of AI is called *distributed AI*. MASs are ideal for distributed situations because they are easily extensible (*i.e.* the design is independent of the number of agents) and they are able to deal with a large number of tasks while maintaining performance [130]. As such, MASs are becoming increasingly popular with the emergence of Grid computing, and their theoretical aspects can also be used, for example, in modeling web services. However, they should not be confused with *swarm intelligence*: although this also studies decentralized self-organized system, it is more interested in the emergence of a collective behaviour and has its roots in algorithms modeling ant colonies.

When designing a MAS, a number of choices have to be made regarding the cooperation or the use of learning methods. A classification offers a compact overview by focusing on a few points, and can be narrowed down as choices are made. Nwana's classification [111] considers three "fundamental characteristics" leading to four types (fig. 4.1): *collaborative agents*, *collaborative learning agents*, *interface agents* and *truly smart agents*; agents combining two or more approaches are classified as *hybrids*. The classification is further extended by considering the *mobility* (can the agents move in the network?) and the logic paradigm employed for the reasoning: if the agent maintains a symbolic model of the world, making decisions via symbolic reasoning and "engaging in planning and negotiation in order to

achieve coordination with other agents", then it is a *deliberative agent*; on the other hand, agents without symbolic models and acting on a stimulus/response base are called *reactive*. So far, the agents that we have considered are reactive agents as there was no attempt whatsoever to communicate with other agents, although it can be argued that the distinction is somewhat arbitrary because the content of an agent's memory is a rudimentary way of keeping a symbolic representation of its environment. Thus, we are introducing another difference into the system: the agents become 'intelligent' *and* deliberative.

To better understand the system we are about to present, we use a classification based on six features from [63], which considers autonomy as a requirement as well as temporal continuity ("agents must run continuously rather than simply perform a task and terminate"):

- *Pro-activeness*, how it reacts to the environment. At one end is the *pure reaction* of a stimulus/response, at the other end is *pure planning* involving a symbolic approach made of beliefs, desires and intentions; a *hybrid* approach combines reaction and planning by using priorities: a reaction can be overridden by a plan.

- *Adaptiveness*, how it modifies its behaviour. The common ones are *non-adaptive* agents that do not change over time and *learning* agents, sometimes restricted by *constraints*.

- *Mobility*. An agent is *physically mobile* if it can move between machines or *logically mobile* if it executes on one machine but can access other ones via the network; otherwise, it is *static*.

- *Collaboration*, either via messages (*communicative*) or by interactions with the resources and the environment (*non-communicative*).

- *Veracity*, how reliable are agents to each other. An agent is *truthful* if it doesn't attempt to mislead another agent and *untruthful* otherwise.

- *Disposition*, the attitude of agents toward each other. A *benevolent* agent always attempts to perform the task that was requested, while *self-interested* agents only collaborate if they have an interest, and *malevolent* agents act in a malicious manner.

Our agents will be *hybrid*, *learning*, *logically mobile*, *communicative*, *truthful* and *benevolent*. This is different from the previous chapter with respect to three of the features, as the agents were previously *pure reaction*, *non-adaptive* and *non-communicative*. A setting

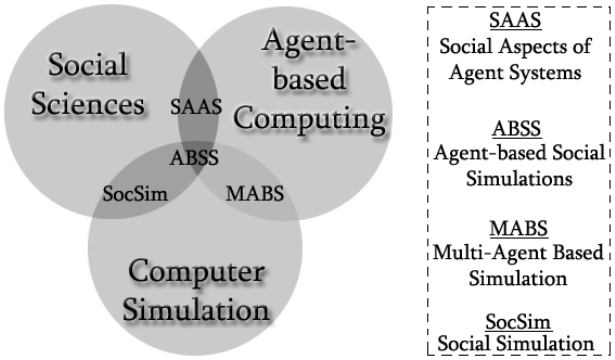

Figure 4.2: Intersection of agent-based computing with other fields.

that we will not explore but of interest is of unreliability, which is found in two cases. Firstly, elaborate viruses may try to disguise as agents and thus we face *untruthful* agents. Secondly, agents of different 'generations' may ignore requests for collaboration from older agents whose design can be thought of as less efficient, thus, agents could be *self-interested*.

Once the type of MAS has been carefully chosen, the next step is to establish the architecture of the multiagent system. Given that the specification of an architecture depends on the purpose of the system and that agents have been used in a wide number of fields, there exists a myriad of possible architectures. In particular, Davidsson [39] has shown that agent-based approaches from computer science can share common interests with social sciences and computer simulations (figure 4.2). Appendix II provides the interested reader with a more complete discussion of agents as *players* in game theory, *actors* in actor theory or *processes* in computer systems.

In Section 3.1, we presented the parameters of an (antivirus) agent: it has sight ℓ in which it sees vertices, then ranks them, filters using the contents of its memory, and finally broadcasts to as many vertices as it can in decreasing order of ranking (*i.e.* it is being limited). These are summarized as *functions* in Figure 4.3, which extends the model shown in Figure 3.1 in Section 3.1. The aspects of intelligent deliberative agents are related to the *knowledge evolution* module: the agent observes its environment, deduces a knowledge model from it, and exchanges models with other agents (by sending its own model to their mailboxes and receiving their models in its own mailbox). This results in a synthesis that will be used for the ranking function. The higher level principles of this module are discussed in the remainder of this chapter, while the algorithms and data structures underlying it are explored in the next chapter.

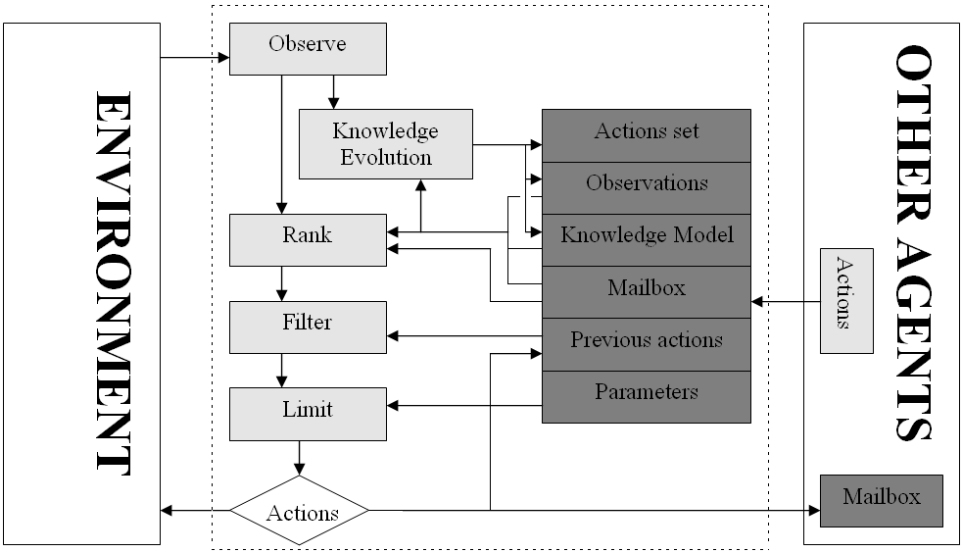

Figure 4.3: Architecture of an agent.

4.2 Agents applying data mining to dynamics

As the architecture consists of abstract modules, let us come back to our concrete goal. A typical network evolves over time, with rules underlying its evolution. For example, when new nodes join the network, they can have explicit preferences about who they will link to (such as a preferential attachment based on degree) or an implicit bias (such as vertices with highest betweenness centrality being the ones more likely to be encountered and thus to befriend). Clearly, agents with fixed behaviour can quickly become obsolete in a dynamic network whereas agents that can adapt to changes should perform much better. Adapting to changes means that agents have to be able to *understand* them, thus our *knowledge evolution* module aims at understanding the dynamics. As we said, an agent observes its environment and deduces a first model from it: this is exactly data mining, which is defined as "the application of specific algorithms for extracting patterns from data" [45]. Data mining can be used mainly in two ways with multiagent systems, for which the frameworks are illustrated in figure 4.4 according to [130]:

- At the behavior level, the sources of data are the actions taken by the agents, either on the environment or between each other (including messages). Goals can be to predict the behaviour of agents, or to improve it. For example, in [128], the agents learn how to identify important messages by analyzing the log of their interactions with other

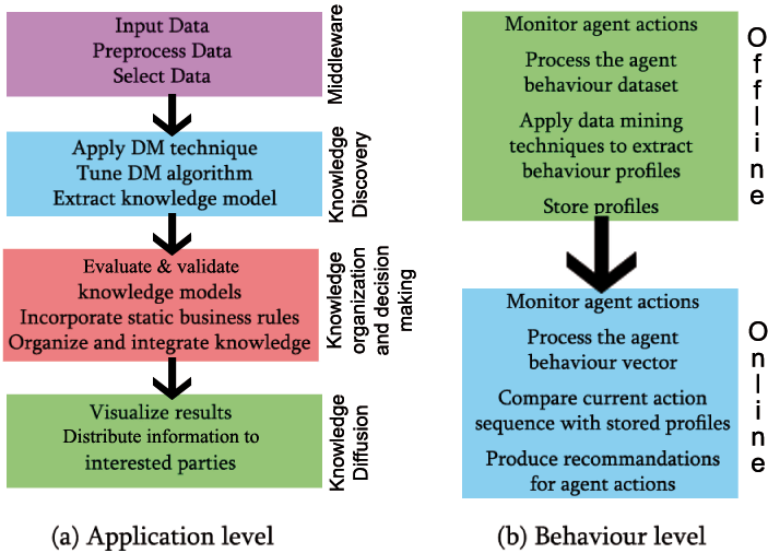

Figure 4.4: Main steps of data mining to improve the behaviour of the agents (a) or to distribute the computations (b).

agents; a consequence of this improvement is better coordination among the agents, as they can postpone the actions in which they were involved when they receive a message that is truly important.

- At the application level, the sources of data are user-provided, *i.e.* any dataset can be given. The goal is to be able to perform a given data mining task, such as producing a decision tree, in a distributed fashion: not only can it process faster, but it can also handle higher volumes of data.

Given this dichotomy, it is clear that our agents will use data mining for their behaviour: by having a model of the network dynamics, they change the way they choose their targets. Concretely, at each step, an agent of the defence system residing at a node scans its immediate neighbourhood to collect basic information such as the degrees of its neighbours or the clustering. If a neighbour does not respond between two consecutive steps, the agent considers it to be dead and changes the neighbour's status accordingly. When enough observations have been collected, each agents runs a classification algorithm: in figure 4.5, we use a C4.5 algorithm [117] which produces a decision tree based on information entropy[1]. In general,

[1]Some minor preprocessing takes place before running the algorithm. At each time step, the agent records the state of its neighborhood in the form *TIME*—*vertex*$_1$-*degree-status*—...—*vertex*$_n$-*degree-status*. This is

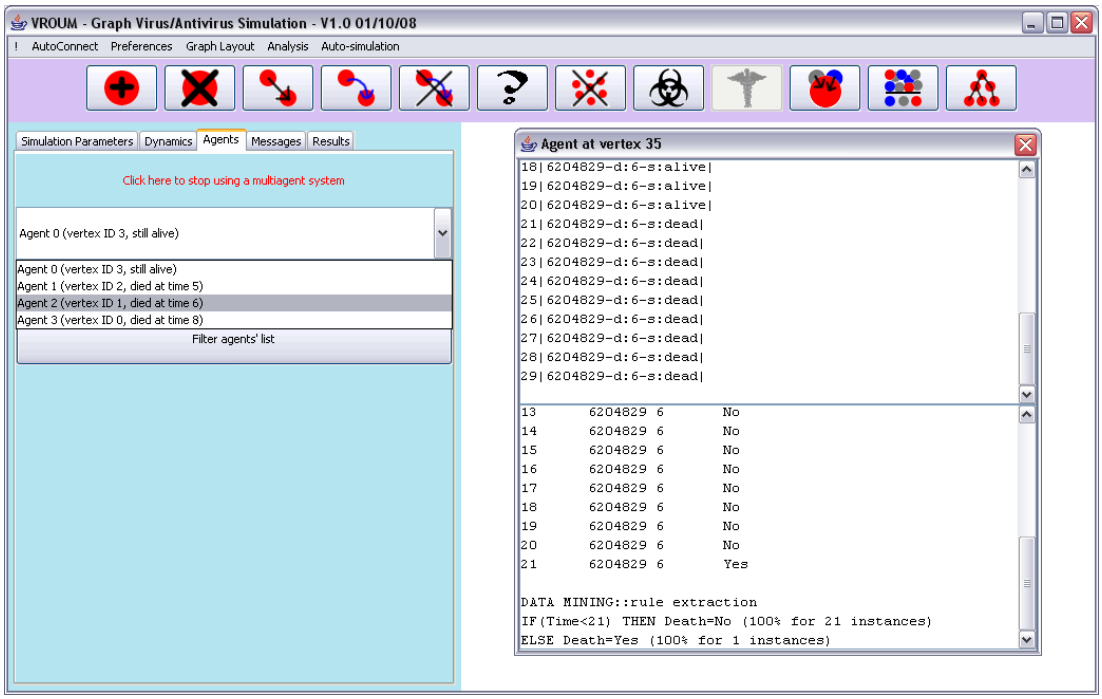

Figure 4.5: In our software, each agent observes its neighborhood and runs a C4.5 algorithm to extract rules from its observations.

agents extract rules from data to predict if a node is likely to die given its attributes, thus the data mining task is to predict how the network *shrinks*. However, a problem typically encountered in multi-agent systems is that the agents have different views. For example, we can consider the configuration in which a small number of nodes die when their age exceeds three while the other nodes keep aging and die later. The rules deduced by the agents will differ on the age of death depending on their views, as we see in figure 4.5 in which an isolated node experienced a delay such that it only noticed some of its neighbours' deaths after 21 steps while the rule was *IF age* > 3 *THEN die*. To deal with this issue, we develop a new algebraic framework called *decision spaces* in the next chapter.

Furthermore, to avoid overfitting the data we should delete any rules concerning too few cases, and this could result in small local phenomena such as the death of a few nodes in the neighbourhood being seen erroneously as noise. To solve issues such as differences of view, it is common for agents to cooperate by exchanging information with agents in their

converted to a table in which columns represent time, IDs of vertices, their degree and their status, thus there is one vertex per line.

neighbourhood. This allows more accurate predictions that are based on a more complete view of the network[2].

4.3 Exchange of data and positions

Exchanging raw information between agents is not efficient from either the communication or computation points of view:

- The amount of data to be transmitted can potentially be overwhelming, whereas decision trees offer a very compact representation of the data.
- As the C4.5 algorithm is not linear in the size of the data, it is more costly to apply it to a single large data set than to several smaller data sets.

Thus, our agents exchange the information in their decision trees and a small amount of additional information instead of their observations. However, while it is trivial to merge two sets of observations, we need to define a merging operator for decision trees. Furthermore, what about agents sending their model with a delay? To solve this problem, we designed a new structure upon which an algebra can be defined to formally specify the correctness of the structure; this structure is introduced in the next chapter.

However, the lack of operators is not the only challenge. Indeed, our approach relies on the idea that agents would observe expressions of the dynamics in different parts of the networks and work together toward a consensus, that is, they do distributed learning on disjoint subsets of vertices. But what if some subsets are observed by several agents? Whatever happens in this subset will be over-represented in the consensus, and this situation should be avoided. In other words, we have to minimize interference. We do not address this problem in this thesis. However, a rich literature in reducing interference, and protocols for gathering data, can be found in sensor networks, although our motivation is to avoid an over-representation of a sample whereas interference in sensor networks causes problems in receiving messages and leads to retransmissions that consume the crucial resource of power. The situation is well summarized as follows:

[2]We assume that same dynamic is governing the entire network, or that the local differences in the expression of this dynamic are reasonably small. While our goal is to derive a consensus about *one* overall picture of the dynamics, approaches that have been suggested to take into account local differences. For example, *power clouds* were suggested in [89] by considering that having differences in a subnetwork is similar to uncertainty in the exponent of the power-law found in a subnetwork.

> *"In many situations, data at nodes are not independent. Due to the correlation present, [...] approaches that take into account this correlation [should] outperform traditional approaches [...] Moreover, jointly exploiting the data structure and optimizing the transmission topology (structure) in the network can provide substantial further improvements. Therefore, it is worth studying the interaction between the correlation of the data measured at nodes and the transmission structure that is used to transport these data."* [38]

Furthermore, two approaches were suggested in [38]: either complex coding with simple routing where we need both global network knowledge and information about the correlation structure, or simple coding and complex routing where we work at a local level by learning the correlation structure. A tradeoff between global knowledge and purely local knowledge, as we have already seen, is to relax the distance to $d > 1$ and thus we can use complex coding techniques within a cluster and complex routing between clusters. This idea of multi-level or hierarchies leads to schemes such as the one in figure 4.6, similar in many aspects to the ones in [60]. Due to heterogeneities in networks, in particular scale-free ones, it can be safe to assume that vertices with a high degree also have higher bandwidth in order to maintain a quality of service to their neighbours. Thus, such vertices with high bandwidth could be chosen as observers with a distance between them of close to 2, in order to limit interferences and also minimize delays in transmission. Then, the vertex with the highest bandwidth can centralize the exchange of data, enabling a hierarchical gossiping; in figure 4.6 the central red vertex would centralize the decision trees from the agents in gray crosses. Furthermore, studies have found that the presence of properties such as scale-free are actually very beneficial for our goal: "an ad hoc network should organize itself according to a scale-free topology when trying to efficiently disseminate information to its members" [52].

4.4 Requirements in highly dynamical massive systems

So far we have assumed that an agent can keep a complete record of all changes taking place in its neighborhood, and we did not impose bounds on the processing time of the records required to extract a knowledge model such as a decision-tree. However, this is challenged by systems in which an agent monitors a very large number of nodes and/or nodes join and leave the network very frequently. Indeed, the limited memory of an agent might not

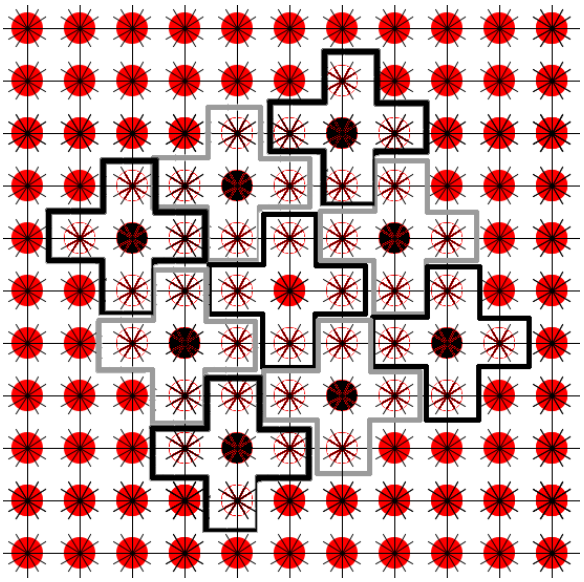

Figure 4.6: Observing agents (black) record changes in their cross-shaped neighborhoods and produce a decision-tree.

be able to store all records, and the speed at which agents have to be able to cope with changes imposes fast approximate answers. This has a strong resemblance with the three constraints of data streams algorithms [1]: they can look at a record only once when it arrives (*single pass*), the data structure used to represent the data stream has a limited storage space (*bounded storage*), and the time spent to process each record must be low (*real-time*). The concepts that our system needs to use for such a setting is a topic on its own, and the interesting reader will find some developments in Appendix III. In particular, we put emphasis on the data structures, formalisms and learning methods; this review is targetted at a conceptual level, thus the primitive operations and their implementations can be found following the references therein but are not an object of our study.

Chapter 5

An Algebraic framework to Combine Classifiers

Learners are distributed by nature in a growing number of systems, such as sensor and peer-to-peer networks. Each learner can run a data mining algorithm to deduce a model from its local environment, but it only has a small picture. However, a more complete and accurate picture of the environment can be obtained by propagating local models among the learners. There are two main approaches to integrating a set of models with the model computed by a learner: either they are all kept separate and weighted (*ensemble classifiers*), or a new model that combines them (*meta-learning*) is learned. In this chapter, we consider the case of meta-learning in which the models are classifiers, potentially of different types. The concepts of classifiers and meta-learning are discussed in Section 5.1.

Classifiers are rather complicated objects, and combining, or *merging* them, is not a straightforward operation. First, classifiers can be of very different structures, such as support vector machines and rule sets. Then, *conflicts* arise when classifiers differ in their predictions, and solving these conflicts requires extra statistical knowledge, or heuristics in the case of pure meta-learning. Most of the proposed solutions have been application specific and were validated through experiments. In contrast, our approach emphasizes the formal aspects. In Section 5.2, we introduce *decision spaces* allowing different classifiers to be merged without loosing information even though the structure might be simplified by the process. Decision spaces are an algebraic framework used for certain operations; they are not classifiers, thus they do not learn data. In Section 5.3, we present a merge operator that

solves all conflicts without requiring extra knowledge, and we characterize its properties using algebraic methods.

We show that the merge operator is not associative: if three or more models have to be merged, then the order in which they are merged affects over the result. Furthermore, the impact of a model decays exponentially with the time at which it was received. This can be a very desirable feature in settings such as *data streams* [103], in which the most recently received data can be considered to be the most representative of the current trends. However, there also are cases in which the ordering should not matter: for example, a massive homogeneous database that is partitioned into blocks distributed for the sake of computability. To ensure that our operator satisfies both needs, we show in Section 5.4 how particular schemes can ensure the same result regardless of the order in which the models are merged.

Our formal framework can also be used to characterize other common, but difficult, problems of data mining. For example, a learner can generate a sequence of models and analyze this sequence to find patterns of changes in the underlying system. In a data stream setting, this is refered to as a *blind method* operating over a sliding window. In Section 5.5, we define an intersection operator that simplifies the models in a sequence to permit easier analysis of the sequence, and we characterize its properties using an algebraic approach. We also briefly discuss how more complex operators can be defined using compositions of the restriction and merge operators.

Finally, we propose a few heuristics in Section 5.6 to reduce the time and space complexities of the framework. This opens up possibilities for efficient implementations that adapt this framework for specific applications.

5.1 Background.

In the classification problem, data is of the form $(a_1, \cdots, a_n, y)$ where a_i is the value of the *i-th* attribute, and y is a class label for this data. For example, we have two instances $(a_1 = 30, y = \text{“}no\text{”}), (a_1 = 70, y = \text{“}yes\text{”})$ where attribute a_1 is an age in $[0, 150]$ and the class label is either *yes* or *no*, corresponding to an individual being classified as old or not. The goal is to find the best approximation to the function $f(a_1, \cdots, a_n) = y$ that determines the label of an unclassified instance given the values of its attributes. Three techniques commonly used to train a classifier, *i.e.* to deduce an approximation of f given

a set of instances, are:

- Consider that a data point is an n-dimensional vector, and that all data points are classified into two possible classes. A Support Vector Machine (SVM) classifies the data points by separating them with the $(p-1)$-dimensional hyperplane that leaves the maximum margin between the two classes [129]. It is also possible to obtain non-linear classifications using the kernel method.

- One of the most popular classifiers is the *decision tree*. A *decision tree learner* [117] applies a divide and conquer technique: a node splits the data using the value of an attribute (maximizing a metric such as the information gain or Gini index), and the procedure is repeated recursively. Thus, a path in the tree corresponds to a set of conditions on the data, and leads to a class distribution.

- A classifier can also be a set of rules based on the values of the attributes; this is commonly referred to as a Rule Set [46]. A rule is a conjunction of conditions on the attributes that results in a class distribution vector expressing the percentage of instances for each given class. An attribute can be repeated at most twice in a rule, to specify an interval with a lower bound and an upper bound.

In this chapter, we focus on combining the results of multiple learners, which was defined as *metalearning* in [27]. In metalearning, each learner builds a complete model of its local environment, such as a decision tree, and exchanges it with other learners; each learner then disposes of a set of classifiers and combines them to create only one classifier. A number of researchers have focussed on merging decision trees: it was noted in [10] that *"a kind of decision tree induction [...] efficient in a wide area system employs metalearning, [in which] each computer induces a decision tree based on its local data and then the different models are combined to form the final tree"*. For example, it was proposed in [61] to transform decision-trees into the set of rules that they represent and to combine those sets. A rule not in conflict with other rules can be kept intact and otherwise the conflicts are solved using a heuristic. However, there are two problems with this approach. Firstly, the authors argue that conflicts that are not handled by the heuristic are *"unlikely if the training sets contain similar distributions of examples from a coherent larger training set"*. This assumes that the data are uniformly and independently distributed, which is not the case in a data stream setting. Secondly, one of the steps might not only need to perform data mining

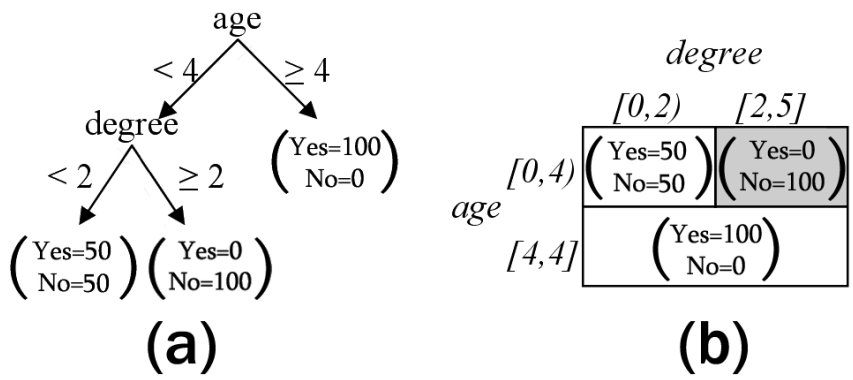

Figure 5.1: Comparison of decision tree (a) and decision space (b) representations.

again, but will ask a learner to send all the data on which there is a conflict. This is not purely metalearning, and is not doable because it requires all examples to be stored and communication can be prohibitably expensive. In the next section, we introduce decision spaces to investigate this problem in a rigorous framework, solving *all* conflicts without requiring any data outside of the models.

5.2 A Framework: Decision Spaces.

5.2.1 Introducing the Structure.

Intuitively, a *decision space* is an n-dimensional space, in which each dimension corresponds to the range of an attribute. It contains a set of non-overlapping elements which, if they cover all the ranges, form a partition of the space. A geometrical interpretation of an element is a specific region defined by an *n-polytope*. Each element, or polytope, has a class distribution vector specifying the number of instances that fall into each class of the space that the polytope covers. An example of a decision space is shown in Figure 5.1(b): it has two attributes, *degree* and *age*, and three elements, each with a class distribution vector of size 2 (with classes Yes and No). These concepts are formalized in Definitions 11 and 12.

Definition 11. A decision space is an m-dimensional space $D_{attr_1} \times \cdots D_{attr_i} \times \cdots \times D_{attr_m}$ where m is the number of attributes and D_{attr_i} is the range of the *i-th* attribute specified as a bounded poset (*i.e.* a partially ordered set with a least and a greatest element).

Definition 12. An element of a decision space D is a subspace of D, *i.e.* a polytope of m dimensions (*m-polytope*) where m is the number of attributes. It is identified by a set of

coordinates for each attribute, and contains a class distribution vector of c elements in [0, 100], where c is the number of classes. The *i-th* value of the vector is the probability that an instance with values that are within the element's ranges is in class c_i; thus, the vector's content sums to 100%.

5.2.2 Conversions.

As a decision space is a mathematical structure with few constraints, classifiers can be converted effectively into decision spaces. Furthermore, having fewer constraints does not mean that information is lost. Indeed, the constraints on the structure of a classifier are used in the data mining process to guide the search, while our decision space is a framework and does not result directly from a data mining process. Prior to converting a classifier, we also require the ranges of attributes on which the classifier was trained. These ranges can be trivially deduced in one pass over the dataset by scanning for the maximum and minimum values. The ranges can also be user supplied, but should not be smaller than what is found in the dataset for consistency purposes. Thus, given a classifier and the ranges of the attributes, the main task of the conversion is to extract the individual elements, or polytopes. The polytopes for elements of the three classifiers defined in Section 5.1, in order of increasing constraints on the shapes, are:

(1) The regions in an SVM can have the most general shapes because the data can be separated into regions in a non-linear fashion.

(2) A rule in a rule set defines an axis-parallel rectangle.

(3) As shown in figure 5.1, each path of a decision tree can trivially be converted to a rule, and this rule defines an axis-parallel rectangle.

A decision tree can only generate certain axis-parallel rectangles. Indeed, a decision tree belongs to the data mining family of *separate-and-conquer* algorithms that imposes constraints on the search. Intuitively, a cut in the space along the border of an element, either vertical or horizontal, should not cut any element [48]. An example of a set of rules that violates this constraint is given below, and shown in Figure 5.2:

$$\text{IF } age \geq 0 \text{ AND } age < 4 \text{ AND } degree \geq 0 \text{ AND } degree < 2 \text{ THEN } A$$
$$\text{IF } age \geq 4 \text{ AND } age < 6 \text{ AND } degree \geq 0 \text{ AND } degree < 4 \text{ THEN } B$$

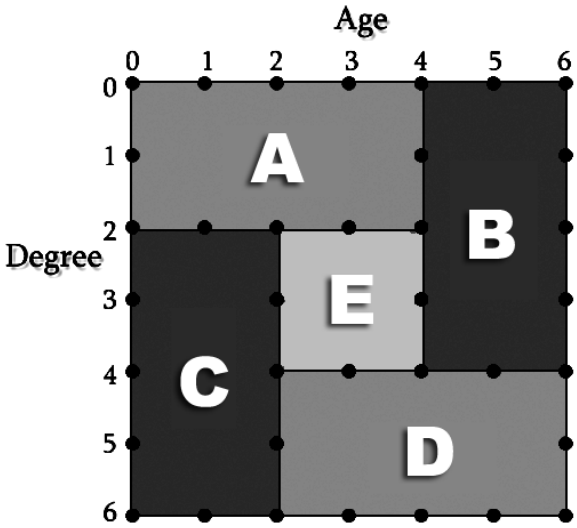

Figure 5.2: A partition of space not allowed by decision trees but allowed by decision spaces.

$$\text{IF } age \geq 0 \text{ AND } age < 2 \text{ AND } degree \geq 2 \text{ AND } degree < 6 \text{ THEN } C$$
$$\text{IF } age \geq 2 \text{ AND } age < 6 \text{ AND } degree \geq 4 \text{ AND } degree < 6 \text{ THEN } D$$
$$\text{IF } age \geq 2 \text{ AND } age < 4 \text{ AND } degree \geq 2 \text{ AND } degree < 4 \text{ THEN } E$$

The rules from (2) and (3) can be converted to elements using Algorithm 3 below. The algorithm uses pattern-matching. For example, in line 4, $attr_k\ OP_1 = \{<, \leq\}\ val_k$ is a pattern for which the value of an attribute has to be lower or strictly lower than a value. If the pattern is found, then $attr_k$, OP_1 and val_k are bound to the actual values. For example, a pattern $age < 5$ will result in the binding $atttr_k = age$, $OP_1 =$ " $<$ ", $val_k = 5$.

For each rule, the algorithm considers all patterns that specify an upper bound (line 4). If there is also a pattern specifying a lower bound for the same attribute (line 6), then the range of the attribute can be specified. If no pattern is found for the lower bound of an attribute $attr_k$ (line 15), then we use the lower bound of the attribute's range which we denote $min(D_{attr_k})$. Finally, if no upper bound is found for $attr_k$, we use the upper bound of the attribute's range which we denote $max(D_{attr_k})$.

Algorithm 3 RULESTOSPACE(Ruleset R, Attribute ranges $D_{attr1} \times \cdots \times D_{attrM}$)

Require: Rules are expressed in the following form:
$r :=$ IF $attr_1 \asymp val_1$ AND $attr_2 \asymp val_2$ AND $\cdots$ AND $attr_m \asymp val_m$ THEN CLASS $= X$

Decision space S $\leftarrow \emptyset$
for $r \in R$ **do**
  element $e.value \leftarrow X$
  $P \leftarrow$ ALL PATTERNS $attr_k$ $OP_1 = \{<, \leq\}$ $val_{k_{up}}$ IN R
  for $p \in P$ **do**
    if THERE EXIST A PATTERN ($attr_k$ $OP_2 = \{>, \geq\}$ $val_{k_{low}}$ IN R **then**
      if OP_1 IS $<$ AND OP_2 IS $>$ **then**
        $e.range \leftarrow e.range \cup (val_{k_{low}}, val_{k_{up}})$
      else if OP_1 IS $<$ AND OP_2 IS $\geq$ **then**
        $e.range \leftarrow e.range \cup (val_{k_{low}}, val_{k_{up}}]$
      else if OP_1 IS $\leq$ AND OP_2 IS $>$ **then**
        $e.range \leftarrow e.range \cup [val_{k_{low}}, val_{k_{up}})$
      else
        $e.range \leftarrow e.range \cup [val_{k_{low}}, val_{k_{up}}]$
    else
      if OP_1 IS $<$ **then**
        $e.range \leftarrow e.range \cup [min(D_{attr_k}), val_{k_{up}})$
      else
        $e.range \leftarrow e.range \cup [min(D_{attr_k}), val_{k_{up}}]$
  if $P = \emptyset$ **then**
    $P \leftarrow$ ALL PATTERNS $attr_k$ $OP = \{>, \geq\}$ $val_{k_{low}}$ IN R
    for $p \in P$ **do**
      if OP_2 IS $>$ **then**
        $e.range \leftarrow e.range \cup (val_{k_{low}}, max(D_{attr_k})]$
      else
        $e.range \leftarrow e.range \cup [val_{k_{low}}, max(D_{attr_k})]$
  $S \leftarrow S \cup e$
return S

5.3 Merge Operator.

5.3.1 Preliminary Definitions.

The most fundamental operation on decision spaces is merge. Given two decision spaces X and Y, we merge them into Z using the following principles:

1. If an element $x \in X$ does not intersect with any element $y \in Y$, then the rule represented by x has no conflicts and can be added to Z. An element $y \in Y$ with no conflicts is treated similarly.

2. If an element $y \in Y$ is strictly contained within an element $x \in X$ with the same value, then it can be deleted. Indeed, the rule represented by y is unecessary and is too specialized.

3. If neither of the first two conditions is satisfied, then the element $x \in X$ intersects with at least one element of Y and conflicts must be resolved.

Before giving an algorithm to merge elements, we need to specify more formally the ways that elements can intersect. The definitions below use the following notation for an element $x \in X$:

- x is defined for a set of attributes $A(x)$.
- x has a class distribution vector $V(x)$.
- Each attribute $a \in A(x)$ has a lower bound $low(x, a)$ and an upper bound $up(x, a)$.

Definition 13. An element $x \in X$ *subsumes* an element $x' \in X'$, denoted $x' \subseteq x$, if $A(x) = A(x')$ and $\forall a \in A(x), low(x, a) \leq low(x', a)$ and $up(x, a) \geq up(x', a)$.

In other words, an element x' is subsumed by an element x when all the ranges characterizing x' are included in the ranges characterizing x. We denote strict subsumption by $\subset$, where the bounds are specified with $<$ and $>$. The main property of subsumption is established by Theorem 1.

Theorem 1. Let X and X' be two decision spaces. There is at most one $x \in X$ such that for any $x' \in X'$, $x' \subseteq x$.

Proof. The elements $x \in X$ and $x' \in X'$ partition the Euclidian space formed by X and X', so one partition can be included in at most one other partition. □

Definition 14. The intersection of an element $x \in X$ with a decision space Y is denoted $x \uplus Y$ and is the set $I = \{y_1, \cdots, y_n\} \subseteq Y$ such that $\forall y_i \in I$, $\exists a_i \in (A(x) \cap A(y_i))$ and $[low(x, a), up(x, a)] \cap [low(y_i, a), up(y_i, a)] \neq \emptyset$. Subsumption is a special case of intersection.

The first two principles of merging can be handled by the notions in Definitions in 13 and 14. For the third principle, we resolve each conflict between two elements $x \in X$ and $y \in Y$ by creating a new element z for each intersection. The value of z depends on the elements x and y that were intersecting. A simple approach would be to assign to z the average value of those elements, but it would not take into consideration the coverage of an element. Indeed, an element could span a broad range of values for various attributes, or only a small range, and thus it is more or less *specialized*. As an element covering a small

range should not have the same weight as one covering a broad range, we use a weighted average based on the specialization. First, we define the following metric of specialization M.

Definition 15.

$$M(x \in X) = \frac{\sum_{a \in A(X)} |up(x,a) - low(x,a)|}{|A(X)|}$$

Intuitively, we sum the sizes of the ranges for each attribute characterizing x and we normalize by the number of attributes so that the minimum is 1. Small values of M indicate specialized rules based on small ranges. Other possible metrics could take into account the number of cases, or the class distribution. However, it must be possible to compute a metric for merging for only some parts of an element, and it is not possible to know either the number of cases or the class distribution in a specific part of an element without using the instances themselves. Thus, we designed M for the particular use of merging. It will become clear as we analyze the algebraic framework that it is possible to tune the metric to better reflect an application without having to modify the algebraic framework itself.

The weighted average based on the specialization is given by the following formula for two intersecting elements $x \in X$ and $y \in Y$:

$$\bigotimes(x,y) = x \otimes y = V(x) \times \frac{M(x)}{M(x)+M(y)} + V(y) \times \frac{M(y)}{M(x)+M(y)}$$

5.3.2 Merging Algorithm.

The merge operator $\bigotimes : (X,Y) \mapsto Z$ is defined in an algorithmic way by Algorithm 4 and illustrated in Figure 5.3. First, we apply principle (2): all elements of $x \in X$ that are strictly contained in an element $y \in Y$ with the same value are deleted. Then, we apply principle (1): if the element x has no intersection, it can be added. Principle (3) is applied as follows:

- We consider all possible intersections of x with $y \in Y$ and handle them one by one.
- We create an element z for which the range is the intersection of x and y using the weighted formula to assign the value.
- There will be a 'leftover' when the process is over if x only intersects partially with Y. Thus, we remove from x the range of each z resulting from the intersection, and we add what is left of x to the result if it is not empty.

Principles (1) and (2) have to be repeated for each element $y \in Y$. However, the elements in the intersection of X and Y have already been computed and thus that step can be *partially* avoided using an auxiliary set A of pairs (y, y'), in which the element y is used as a key and the element y' contains a range updated throughout the process. When an intersection between x and y is found, the part in the intersection is virtually removed from y by changing the value of y' in A. After all the intersections have been computed, A contains the leftover of each element of Y and thus it can be directly added to the result. A can typically be implemented using a hash table.

Example. In Figure 5.3, the pink element x and the green element y intersect. As two spaces can only intersect in one continuous space, the result is the new space z defined by $up(z, age) = 8$, $low(z, age) = 7$, $up(z, degree) = 10$, $low(z, degree) = 3$. Each value of the class distribution vector is computed using $x \otimes y$, hence the value for the class *Yes* is:

$$\frac{\frac{15}{2}\times 40}{\frac{15}{2}+\frac{13}{2}} + \frac{\frac{13}{2}\times 0}{\frac{15}{2}+\frac{13}{2}} \approx 21.$$

The same computation is applied for each component of the vector. In this case there are only two classes, so the value for the class *No* is $100 - 21 = 79$.

Other authors have proposed to model a rule as a rectangle. However, the intersection of two rectangles in m-dimensions (for m attributes) might result in spaces that are not rectangles. For example, the leftover of element x in Figure 5.4 is not a rectangle. Constraining elements to be rectangles requires a heuristic and the results will be biased because virtual partitions of these elements will be made and each one will have a smaller specialization than it really represents. Thus, we use polytopes instead of rectangles in order to provide an exact algebraic framework. As there are tractability issues when computing the intersections of complex shapes that are defined by a growing set of coordinates, we propose heuristics in Section 5.6.

5.3.3 Algebraic Properties.

The goal of a merge operator $\bigotimes$ is to combine the information originally found in two decision spaces, resolving any conflicts that arise. Thus, it has to obey a set of algebraic properties in order to be consistent:

- Merging a decision space with itself does not change anything (*idempotence*), as there is neither new information nor conflicts.

Algorithm 4 $\bigotimes : (X, Y) \mapsto Z$

```
1:  Z ← new decision space
2:  A ← ∅
3:  for x ∈ X do
4:      if not (x ⊂ y ∈ Y and V(x) = V(y)) then
5:          if x ⊎ Y = ∅ then
6:              Z ← Z ∪ x
7:          else
8:              tmp ← x.range
9:              for y ∈ x ⊎ Y do
10:                 z ← new element
11:                 z.range ← tmp ∩ y.range
12:                 z.value ← x ⊗ y
13:                 Z ← Z ∪ z
14:                 if ∄(y, y′) ∈ A then
15:                     A ← A ∪ (y, y.range \ z.range)
16:                 else
17:                     A ← A \ (y, y′)
18:                     A ← A ∪ (y, y′.range \ z.range)
19:                 x.range ← x.range \ z.range
20:             if x.range ≠ ∅ then
21:                 Z ← Z ∪ x
22: for y ∈ Y such that ∄(y, y′) ∈ A do
23:     if not (y ⊂ x ∈ X and V(y) = V(x)) then
24:         Z ← Z ∪ y
25: for (y, y′) ∈ A do
26:     if y′ ≠ ∅ then
27:         Z ← Z ∪ y′
28: return Z
```

- Merging a decision space with a decision space that does not contain any elements (*identity element*) should not change anything.
- Merging a decision space with another one should not depend on which one is first but only on the information, hence *commutativity* is also required.

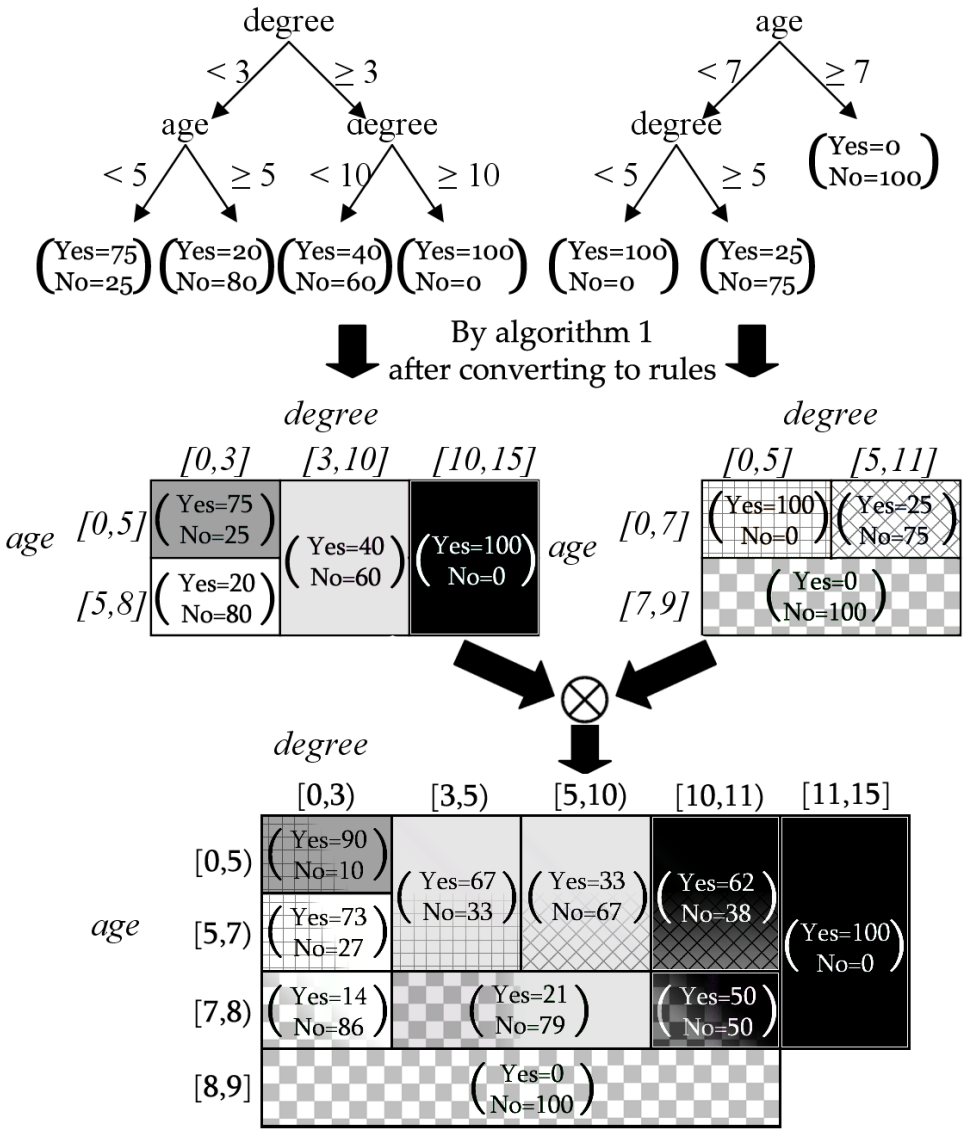

Figure 5.3: Merging two decision trees by converting them into decision spaces and creating a union decision space.

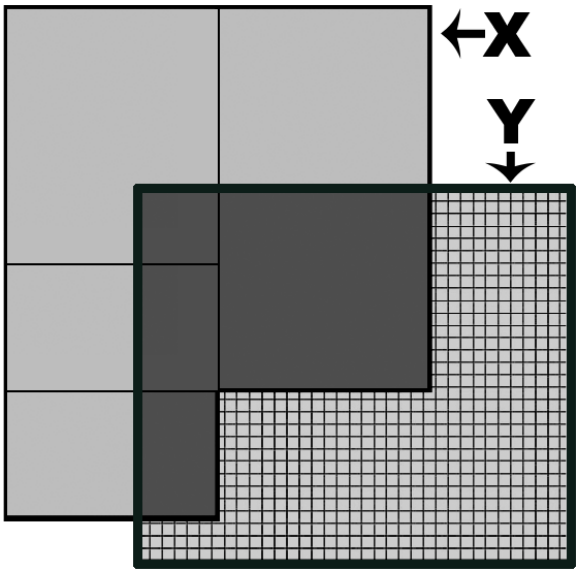

Figure 5.4: Intersection of decision spaces X and Y, showing the leftover of X with cross-hatching.

Theorems 2, 3, and 4 show that these three algebraic properties are satisfied because our definition is based on unions of geometric spaces and the resolution of conflicts encountered for non-empty intersections. This algebraic characterization is summarized in Definition 18.

Definition 16. The set of all decision spaces is denoted by $\mathbb{D}$.

Theorem 2. The $\bigotimes$ operator is commutative, *i.e.* $\forall X \in \mathbb{D}, \forall Y \in \mathbb{D}, X \bigotimes Y = Y \bigotimes X$.

Proof. We use a proof by contradiction, showing that there is no $z \in (X \bigotimes Y)$ such that $z \notin (Y \bigotimes X)$. We consider all possible cases from which such a z can result:

(1) z results from an $x \in X$ that has no intersection. Then, we show that if a $y \in Y$ has no intersection it will also be kept in the result Z.

(2) z results from the intersection of an $x \in X$ with a $y \in Y$. We will show that there are no changes if we consider it as the intersection of a $y \in Y$ with an $x \in X$.

(3) z is the leftover of an element $x \in X$. We show that the leftover of an element $y \in Y$ will also be kept in the result Z.

(1) As $x \in X$ has no intersection, it will be added to the result (line 6). If a $y \in Y$ has no intersection, it will not be in any $x \uplus Y$ and thus we will not create a $(y, y') \in A$ in line 14. As $(y, y') \notin A$ in line 20, y will be considered unchanged. Given that y has no intersection, it will be added to the result in line 22.
(2) As x intersects with a y, the operator $\uplus$ relates x to y (line 9) and an element z will be created, its ranges being the intersections of the ranges of x and y, and its value being $x \otimes y$. Each operation involved is commutative, thus the overall process is commutative.
(3) When an element $x \in X$ intersects with some $y \in Y$, each intersection produces a new element (to resolve the conflict). The part of x that is not included in an intersection with y is the *leftover of x*. For an $x \in X$, we consider each $y \in Y$ with which it intersects (line 9): an element z is created for each intersection, and its range taken out of the range of x (line 19); once the ranges of all intersections have been taken out of x, the leftover is added to the result if it is not empty (line 20). The same process takes place for the leftover of a $y \in Y$: all the $x \in X$ with which it intersects are considered (lines 3 and 9), an element z is created for each intersection, and we keep track of the leftover of y by updating its associated value y' in A (or creating it for the first intersection). An element y will not be considered anymore if it intersected with some $x \in X$ (line 22), and instead its leftover is added to the result (line 25). □

Definition 17. A decision space $E \in \mathbb{D}$ is called the *identity* of $\mathbb{D}$ with respect to the $\bigotimes$ operator if and only if $\forall D \in \mathbb{D}, E \bigotimes D = D \bigotimes E = D$.

Theorem 3. There exists a unique identity $E \in \mathbb{D}$ with respect to the $\bigotimes$ operator, characterized by an empty set of polytopes (*i.e.* the empty space).

Proof. Let X and E be two decision spaces and $E = \emptyset$. We first prove that $\forall X \in \mathbb{D}, X \bigotimes E = X$, which leads to $E \bigotimes X = X$ using Theorem 2 hence E is an identity element. No element x is subsumed by an element $e \in E$, and $x \uplus E = \emptyset$, thus all elements $x \in X$ are added to the result. As there is no $e \in E$, the result is made out of all elements of X and thus is equal to X.

We complete the proof by showing that $Y \in \mathbb{D}$ cannot be an identity for any $X \in \mathbb{D}$ if $Y \neq E$. As $Y \neq E = \emptyset$, there is at least one $y \in Y$. Let us consider $X \in \mathbb{D}$ such that $y \uplus X = \emptyset$. As y has no intersection with any $x \in X$, it will be ignored by the main for loop (line 3 to 21), thus $\nexists (y, y') \in A$. As a consequence, it will be considered by the second loop

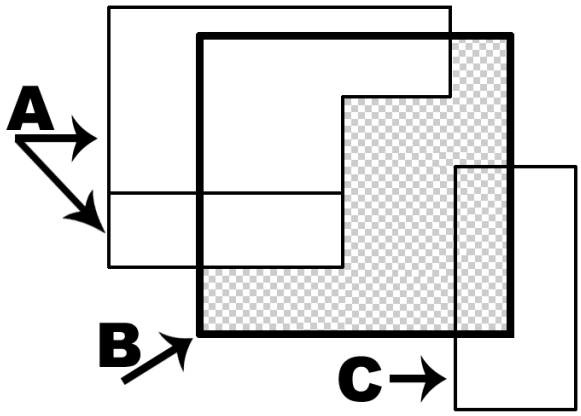

Figure 5.5: Illustration of Theorem 5.

(line 22) and, as it is not subsumed by any $x \in X$, it will be added to the result in line 24; thus, the result is $X \cup y \neq X$ because $y \notin X$. □

Theorem 4. The $\bigotimes$ operator is idempotent as $X \bigotimes X = X$.

Proof. Let X and X' be two decision spaces such that $X = X'$:

- There is no $x \in X$ strictly contained[1] in a $x' \in X'$.
- Each element x intersects with exactly one x'. As they are the same, we will add one element to the result with the following value for each component of the vector:

$$V(x) \times \frac{M(x)}{M(x)+M(x')} + V(x') \times \frac{M(x')}{M(x)+M(x')}$$

$$= V(x) \times \frac{M(x)}{2 \times M(x)} + V(x) \times \frac{M(x)}{2 \times M(x)}$$

$$= \frac{1}{2} \times V(x) + \frac{1}{2} \times V(x) = V(x).$$

- x has been added to the result for all $x \in X$. For all $x' \in X'$, we added a pair $(x', x'') \in A$ such that $x''.range = x.range \setminus x'.range = \emptyset$. Thus, all elements of $x' \in X'$ are skipped because they are in A (line 22), and because the associated value

[1]The use of the subsumption $\subset$ instead of $\subseteq$ is particularly important at this point. Indeed, if $X = X'$ and we were using $\subseteq$, then all elements in X would be discarded because they are subsumed by the same value in X', and similarly for X'. Thus, $X \bigotimes X$ would lead to the incorrect result $\emptyset$.

is empty (line 26). Therefore the resulting decision space contains each $x \in X$ exactly once.

□

Definition 18. The set $\mathbb{D}$ of all decision spaces equipped with the binary operator $\bigotimes : (\mathbb{D}, \mathbb{D}) \mapsto \mathbb{D}$ is a unital and idempotent magma (see [95] for a brief review of algebraic structures such as magmas).

Theorem 5. The $\bigotimes$ operator is not associative:

$$(X \bigotimes Y) \bigotimes Z \neq X \bigotimes (Y \bigotimes Z).$$

Proof. In Figure 5.5, if we first merge A with B, we get the shaded leftover. Then, if we have to merge with C, the value of the intersecting element will be the average of the shaded leftover and C. However, if C was merged earlier, then it would have intersected with an element B spanning a greater space than the shaded leftover. In that situation, the value of the intersection would depend more on B. Thus, the values change with the order in which elements are merged, while the ranges do not (as they result from the intersection of geometrical spaces which is an associative operation). □

5.4 The Impact of Time on merging.

According to Theorem 5, the $\bigotimes$ operator is not associative so the result of merging several decision spaces depends on the order in which they are merged. First, we examine the offline case, in which decision spaces are merged after all of them have arrived. Then, we study the online case in which decision spaces are merged in the order that they arrive. In the online setting, two effects are possible. If the underlying distribution is changing, then recent decision spaces should account for a larger fraction of the result, because they represent recent trends. If the underlying distribution is not changing, then all decision spaces should account for the same fraction. We will show that the first effect is naturally achieved, with the impact of a decision space decreasing exponentially with time. The second effect cannot be guaranteed in an online setting, but we can provide some insight by considering the ideal setting and establishing a scheme that balances the merging efficiently.

5.4.1 Offline merging.

Consider a set $\{X_1, \cdots, X_n\}$ of decision spaces. Our approach extends the approach of Algorithm 2: for each zone where two *or more* elements overlap, we create a new element and we compute its value as a weighted average. In the case of m overlapping elements, the weighted average can be extended naturally:

$$x_1 \otimes \cdots \otimes x_m = V(x_1) \times \frac{M(x_1)}{M(x_1)+\cdots+M(x_m)} + \cdots + V(x_m) \times \frac{M(x_m)}{M(x_1)+\cdots+M(x_m)}$$

The algorithm is based on the principles in [88]: gather all the elements of the decision spaces and compute all the non-empty intersections. Then:

- If an element is strictly included within another one and has the same value, discard it.
- Otherwise, for each intersection, create a new element z with a value that is computed using the extended weighting formula.
- Finally, if the range of an element is different from $\mathbb{D}_\infty$, use an approximation operator to approximate the leftovers and add them to the result.

5.4.2 Online merging.

Consider a set $\{X_1, \cdots, X_n\}$ of decision spaces such that the decision space X_i is received at time i. The goal is to merge decision spaces as soon as possible to avoid having to store all of them as in the offline version. This saving of space also saves time: when the last decision space is received, most of the merging has already been done and only a few extra steps are needed. We propose an approach for a simple setting, in which the elements of the decision spaces all have the same ranges; in other words, the overlap is complete.

A first attempt would be to merge two decision spaces as soon as they are received. If we consider an element x_1 of X_1, it will be merged with an element x_2 of X_2 into an element z_{12} such that $V(z_{12}) = \frac{V(x_1)+V(x_2)}{2}$. Then, X_3 is received and there will be an element x_3 to merge with z, resulting in an element z_{123} with value:

$$V(z_{123}) = \frac{\frac{V(x_1)+V(x_2)}{2}+V(x_3)}{2}$$

In other words, the value of x_3 accounts for as much as the combined values of x_1 and x_2 in the final result. Thus, the process is strongly biased towards the most recently received

decision spaces. We characterize an unbiased result in the definition below, and establish the characteristics of this simple merging scheme illustrated by the tree in Figure 5.6(a).

Definition 19. The merging of n decision spaces $X_1, \cdots, X_n$ is *unbiased* with respect to $\bigotimes$ iff the value $V(w)$ of an element in the resulting decision space is:

$$V(w) = V(x'_1) \times \frac{M(x'_1)}{M(x'_1) + \cdots + M(x'_m)} + \cdots + V(x'_m) \times \frac{M(x'_m)}{M(x'_1) + \cdots + M(x'_m)}$$

such that $w \subseteq x'_1 \in X'_1, \cdots, w \subseteq x'_m \in X'_m$ and $\{X'_1, \cdots, X'_m\} \subseteq \{X_1, \cdots, X_n\}$, $m \leq n$.

Definition 20. A *merging scheme* specifying the order in which a set of decision spaces has to be merged can be represented as a tree in which a leaf represents a decision space, an intermediate node represents the application of the $\bigotimes$ operator (hence an intermediate decision space), and the root represents the final result. A decision space D accounts for a proportion $k = n_1 \times \cdots \times n_p$ of the final result, where n_i is the arity of the i-th intermediate node on the path to the leaf representing D.

Theorem 6. The merging scheme $(((X_1 \bigotimes X_2) \bigotimes X_3) \cdots \bigotimes X_n)$ is biased.

Proof. If we represent this merging scheme as a tree, a leaf at distance d from the root accounts for a proportion 2^{-d} of the final result, as each intermediate node is binary and there are d nodes on the path. As shown in Figure 5.6, leaves are at unequal distances from the root so they account for different proportions and the scheme is biased. More precisely, if a decision space is received at time $t, 1 \leq t \leq n$ then it accounts for a proportion 2^{-n+t-1} if $t > 1$, and 2^{-n+1} if $t = 1$. □

An unbiased scheme is shown in Figure 5.6(b). X_1 is merged with X_2, each one accounting for half of the equation. The result is then be merged with $X_3 \bigotimes X_4$: in this equation, each of $X_1, \cdots, X_4$ accounts for one fourth, and so on. Theorem 7 formalizes this unbiased binary merging scheme, based on the same argument as in the proof of Theorem 6 with the difference that all leaves are at the same distance from the root.

Theorem 7. In the merging scheme $(((X_1 \otimes X_2) \bigotimes (X_3 \bigotimes X_4)) \cdots ((X_{n-3} \otimes X_{n-2}) \bigotimes (X_{n-1} \bigotimes X_n)))$, where $n = 2^k$, each decision space accounts for 2^{-k} of the equation, independent of the order in which they arrive.

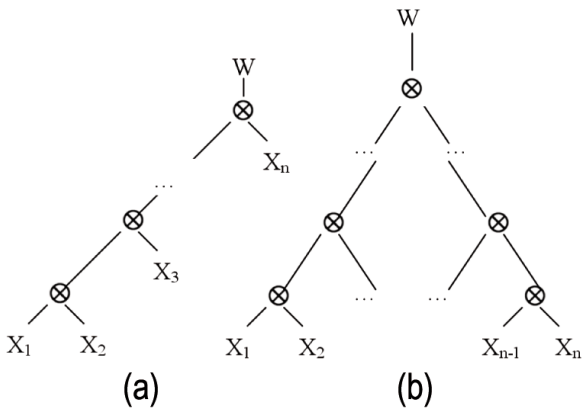

Figure 5.6: Biased (a) and unbiased (b) binary merging schemes.

Since the $\bigotimes$ operator is binary, only binary groupings are possible, and the only possible unbiased merging schemes are for n decision spaces such that $n = 2^k$. We can generalize the $\bigotimes$ operator to allow offline merging for groupings of more than 2 decision spaces. Theorem 8 describes the construction of an unbiased scheme for any number n of decision spaces. An example is shown in Figure 5.7.

Theorem 8. A merging of n decision spaces $X_1, \cdots, X_n$ is unbiased iff the scheme is represented by a balanced tree such that all nodes at a given level have the same number of children.

Proof. First, we consider a merging scheme that is not represented by a balanced tree as specified in the statement of the theorem, and show that the scheme is biased. If the tree is not balanced, there must be at least one element at a different depth from the root. This leads to a biased merging by an argument similar to the proof of Theorem 6. If not all nodes of a given level have the same number of children, there must be two nodes n_1 and n_2 with $d(n_1)$ and $d(n_2)$ children, respectively. These nodes will be merged with an equal weight w, so the children of n_1 will account for $w/d(n_1)$ while children of n_2 will account for $w/d(n_2)$, which is biased as $d(n_1) \neq d(n_2)$. Now, we show the converse: if a merging scheme is represented as specified in the statement of the theorem, then it is unbiased. By construction, the nodes on the path from the root to the leaf all have the same degree sequence $p_1, \cdots, p_d$, where d is the depth of the tree. By Definition 20, all decision spaces account for a proportion $p_1 \times \cdots \times p_d$ of the final result. As time is not a parameter, the result is unbiased. □

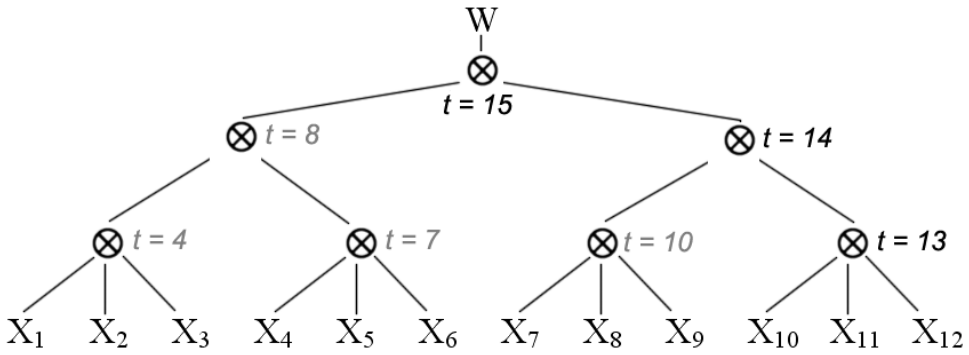

Figure 5.7: Merging scheme from Theorem 13 with $n = 3 \times 2 \times 2$ decision spaces.

In the structure described in Theorem 8, groupings are entirely specified by the degree sequence $p_1, \cdots, p_d$ from the root to the leaves. Different sequences lead to different space consumption, thus efficient sequences are of particular interest. When n decision spaces are merged, the sequence that results in the minimum number of decision spaces stored at a given time is $p_1 \times \cdots \times p_i \times \cdots \times p_d = n$, such that the p_i are prime numbers and $p_d \leq p_{d-1} \leq \cdots \leq p_1$. Indeed, using prime numbers, groupings are as small as possible.

5.5 Developing an Algebraic framework.

5.5.1 Restriction Operator.

In dynamic systems, the accuracy of the rules decreases over time so new decision spaces must be created regularly. A sequence of decision spaces carries information about the evolution of the system, so techniques such as time series analysis potentially could be used. While time series analysis considers a sequence of vectors of *fixed size*, decision spaces can be of varying size: for example the values of attributes can evolve over time to cover a broader space. Thus, a restriction operator can be used to simplify the decision spaces of a sequence so that they all have same size. Formally, the *restriction* of a decision space X by a decision space Y is the decision space Z that only retains elements of X for which the range of the attributes is in Y. The corresponding operator is denoted $\odot$ and defined in an algorithmic way by Algorithm 5.

By definition, we only consider the elements $x \in X$ that intersect with some $y \in Y$ (line 2). For each such element x, we create an element z with the same value (line 6) but a range restricted to the intersection between x and all $y \in Y$ (line 4-5). Intuitively, the restriction

Algorithm 5 $\bigodot : (X, Y) \mapsto Z$

```
1: Z ← new decision space
2: for x ∈ X such that x ⊎ Y ≠ ∅ do
3:     z ← new element
4:     for y ∈ x ⊎ Y do
5:         z.range ← z.range ∪ (x.range ∩ y.range)
6:     z.value ← x.value
7:     Z ← Z ∪ z
8: return Z
```

is simply the intersection of the geometrical spaces, regardless of their values; thus, it benefits from all of the algebraic properties of intersection of spaces, such as idempotency and associativity.

Theorem 9. All identity elements $E \in \mathbb{D}$ such that $X \bigodot E = E \bigodot X = X$ are generated by a family of decision spaces F_v such that $\forall f \in F_v$, $\forall a \in A(f)$, $low(f, a) = -\infty$, $up(f, a) = +\infty$ and $V(f) = v$. The number of elements of F_v is unbounded.

Proof. A decision space X is not restricted by a decision space Y only if all elements in Y have ranges of values at least as large as the range of X, so a trivial identity is the decision space with only one element for which all ranges are $(-\infty, +\infty)$. As the value of this element does not matter, we can define a *family* taking its value v as a parameter. The set of all possible values v can be infinite because the values are taken from a continuous range; thus, F_v contains an unbounded number of identity decision spaces. □

Theorem 10. The $\bigodot$ operator is idempotent as

$$X \bigodot X = X.$$

Proof. Let X and X' be two decision spaces. If $X = X'$ then $\forall x \in X$, $x \uplus X' = x'$ such that $\forall a \in A(x)$, $low(a, x) = low(a, x')$ and $up(a, x) = up(a, x')$. Thus, each element z is created with exactly the range and value of an x, and only one z is created for each x. Hence the result Z is the same as X. □

Theorem 11. The binary operator $\bigodot : (\mathbb{D}, \mathbb{D}) \mapsto \mathbb{D}$ is associative, as $\forall X, Y, Z \in \mathbb{D}$

$$(X \bigodot Y) \bigodot Z = X \bigodot (Y \bigodot Z).$$

Proof. As the values do not matter, the restriction can be considered to be an intersection of spaces, which is associative. □

5.5.2 Composite Operators.

A variety of meaningful operators can be composed from $\bigotimes$ and $\bigodot$. For example, $\bigodot$ can be used to limit the notion of merging expressed by $\bigotimes$: instead of creating each element $z \in Z$ from $x \in X$ *and/or* $y \in Y$, we could create z only from x *and* y. This change has a significant impact: an element derived from only one element is not as precise as an element derived from two, because in the latter case a consensus is obtained through a weighted formula. Thus, this limited merging is less sensitive to noise and, as we know that all elements in Z are derived from exactly two elements, we can have the same confidence in the prediction of all elements in Z. A merging that provides the same confidence in the prediction of each element is obtained through the following composite operator $\bigoplus$.

Definition 21. $\bigoplus : (X, Y) \mapsto Z$ is defined by $(X \bigotimes Y) \bigodot X \bigodot Y$.

This definition ensures that the values are correctly computed based on the specialization of each element of X and Y, and then the overall range is reduced to the intersections with X and Y. A stricter definition that not only restricts the merging to the elements that intersect, but also computes the values based on common ranges of attributes.

Definition 22. $\bigcirc : (X, Y) \mapsto Z$ is defined by $(X \bigodot Y) \bigotimes (Y \bigodot X)$

The main difference from the previous definition is that any part of an element that lies outside the intersection will be ignored when measuring its specialization, and the values are computed based on the same space. The choice between $\bigcirc$ and $\bigoplus$ can be based on the application. Both $\bigcirc$ and $\bigoplus$ have the properties of idempotency, associativity, non-commutativity, and unique identity element.

5.6 Approximations.

5.6.1 Motivation.

In this section, we consider decision spaces as computational objects, focusing on their time and space complexities and on various ways to approximate them. First, as we saw in Section 5.3, the leftover of an intersection of two rectangles does not have to be a rectangle itself. Thus, to keep the specialization meaningful, we need a way to represent the leftover as a single element. As a side-effect, the set of coordinates defining an element can grow each

time the $\bigotimes$ operator is applied. Furthermore, the complex nature of the shapes will increase the complexity of the algorithms in terms of both space and time. We examine this problem through a hierarchical family of decision spaces, show that they have increasing complexity, and discuss heuristics to approximate higher-class decision spaces while benefiting from the smaller complexities of lower classes.

5.6.2 Hierarchy and Heuristics.

Definition 23. A decision space in which each element is defined by at most k pairs of coordinates for each attribute is called a k-decision space and denoted $\mathbb{D}_k$. Two cases of interest are:

- ∞-decision space, in which an element can be defined by an unbounded number of coordinates (hence a polytope). This is the case that we have been considering so far in this chapter.
- 1-decision space, in which each element is defined by one pair of coordinates for each attribute. Thus, the space is partitioned into rectangles.

Theorem 12. The time and space complexities of algorithms 2 (merge) and 3 (restriction) over k-decision spaces are strictly increasing with k.

Proof. The dominant cost in Algorithms 4 and 5 is the enumeration of intersecting shapes, and this cost increases with the complexity of the shapes. □

An upper bound on the time to find all of the rectangles that intersect was shown in [88], but it is an open problem to determine whether the bound is tight: given n iso-oriented rectangles in $d > 1$ dimensions, the algorithm is in $O(n^{d-1})$ time. In our approach, dimensions corresponds to attributes, hence the complexity of a *1*-decision space is already exponential in the number of attributes, which constrains the applications to cases with limited numbers of attributes. While k-decision spaces such that $k > 1$ are interesting theoretical objects, they are impractical and we will concentrate our efforts on the case $k = 1$. In particular, *1*-decision spaces over two attributes can be merged and restricted very efficiently in $\Theta(n \log n)$ time and $\Theta(n)$ space. Two attributes are enough for many applications, such as networks represented by dynamic graphs in which the age and the degree of a vertex are

the main attributes monitored by the system.

Observation: Bounds on Classes.
Let $X \in \mathbb{D}_k$ and $Y \in \mathbb{D}_k$ be decision spaces and let $|Y|$ be the number of elements in Y (resp. $|X|$ for X). Then, $\exists p, p' \in \mathbb{N}$ such that $1 \leq p, p' \leq max(|Y|, |X|) \times k^2$ and:

$$(1) \qquad \bigotimes(X \in \mathbb{D}_k, Y \in \mathbb{D}_k) \mapsto Z \in \mathbb{D}_p,$$

$$(2) \qquad \bigodot(X \in \mathbb{D}_k, Y \in \mathbb{D}_k) \mapsto Z \in \mathbb{D}_{p'},$$

This observation says that when an operator is applied over two decision spaces in $\mathbb{D}_k$, the result can be in a richer decision space $\mathbb{D}_p$, $p > k$. For example, all five of the elements Figure 5.4 (four elements in X and one in Y) are in $\mathbb{D}_2$ while the leftover is in $\mathbb{D}_4$. The upper bounds on p and p' are based on the fact that at most k^2 pairs of coordinates are needed to specify an element that result from the intersection of two elements in $\mathbb{D}_k$. The upper bounds folllow because each element $x \in X$ intersects with at most $|Y|$ elements of Y and each element $y \in Y$ intersects with at most $|X|$ elements of X. The lower bound on p' is achieved if all elements of X are disjoint from those in Y. The lower bound on p is also 1 because the elements in Y can simplify the shapes of the ones in X with ideal cuts.

According to the observation, $\mathbb{D}_k$ is not closed under either $\bigotimes$ or $\bigodot$, as was shown in Figure 5.4. Our goal in introducing this hierarchy is to reduce the complexity of $\bigotimes$ and $\bigodot$ by allowing the use of heuristics to constrain the result to be in an arbitrary $\mathbb{D}_l$, $l \leq k$. In order to do so, we will apply $\bigotimes$ or $\bigodot$ normally, and then transform each element of the result using an operator that approximates them in $\mathbb{D}_l$.

Definition 24. An operator $\approx$: $((x \in X) \in \mathbb{D}_k, l \leq k) \mapsto S = \{s_1, \cdots, s_n\}$ *approximates* the element x in $\mathbb{D}_l$ if it respects the following three conditions:

- $\bigcup_{s_i \in S} s.range = x.range$
- $\forall s \in S, s.value = x.value$
- $\forall s \in S, s \in \mathbb{D}_p, p \leq l.$

In other words, an approximation partitions an element into simpler shapes with the same value. Some partitions are more desirable than others but can be costly to compute. We define the *error* in the approximation as follows:

Definition 25. Given an element x and its approximation $s \in S = \{s_1, \cdots, s_n\}$, the approximation error is:

$$\frac{M(x) - M(s)}{M(s)}$$

.

M is the metric of specialization from Definition 15, which sums the sizes of the ranges for each attribute of x and normalizes by the number of attributes. The elements approximating x have the same value but a smaller specialization, as their shape is smaller. Thus, when using an approximation of x, there will be an error only for the specialization. A straightforward observation is that the error for an element is minimized when its surface is maximized because this surface is as close as possible to the original x. We consider three approximation schemes and the approximation errors that they introduced, illustrated in Figure 5.8:

• Extend the neighbouring shapes to cut the surface, as shown in Figure 5.8(b). Note that not *all* dimensions have to be extended to find a cut. An alternative would be to extend only horizontally and then 1 would be merged with 4, as well as 3 with 5. This is the cheapest method, but no conclusions can be made about the errors; it depends only on the topology.

• Use the most uniform partition into shapes of $\mathbb{D}_l$, leaving the remainders as shapes of decreasing size (Figure 5.8(c)). The distribution of errors is as uniform as possible and only increases slightly for the last generated element, but this approach is costly to compute.

• A greedy partition produces shapes of decreasing surface (Figure 5.8(d)). The errors are inversely proportional to the sizes of the elements, so they are increasing. This approach has a lower computation cost than the previous one.

More complex schemes can be derived from the literature on multidimensional cube packing, optimal rectangular partitions, and the use of *guillotine subdivisions* [99].

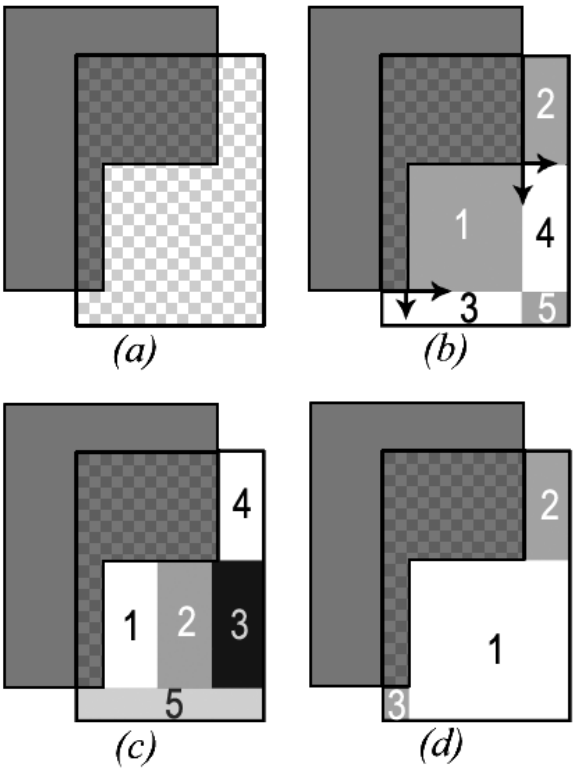

Figure 5.8: Heuristics to partition the original checkerboard left-over space in (a): extending the neighbours (b), uniform partition (c), greedy biggest surfaces (d).

Chapter 6

Conclusions and future work

6.1 Conclusions

We have presented the first steps toward a self-improving immunization system for complex networks, both from theoretical and practical points of view. On a theoretical level, we emphasized a rigorous graph theoretic approach to complex networks throughout our review, introduced a formal framework for two competing broadcasts, and presented a communication efficient approach to exchange knowledge about the network between agents while achieving fast consensus. On a practical level, we conducted a 2^k factorial design showing the contribution of each parameter to the final outcome (including first-order interactions), studied approaches achieving good tradeoffs between memory consumption and performance, and designed software supporting the overall system that may be used as a base or for educational purposes. Numerous directions for future work have been suggested in the review, for example, our approach falls into only one of the four cases that we have identified for immunization systems on complex networks. Complementary theoretical studies are also suggested: dynamic networks call for a new framework that could be inspired by extensions of evolutionary game theory [77], our framework for exchanging knowledge could be extended and applied as a hierarchy to characterize approaches, and we may be able to recognize certain classes of graphs such as scale-free using the outcomes of particular spreading processes. In the next section, we focus on selected future work that stems from early resarch.

6.2 Future work

6.2.1 Scale-free graphs from vertex contraction

Most models for complex networks are non-deterministic, and the ones that are deterministic make little use of graph techniques. In Section 3.2, we improved a deterministic graph model to have a greater small-world effect, and presented a new one with further improvements, while producing graphs that are close to regular. Similarly, there is room for new deterministic models making use of graph techniques regarding the scale-free effect. As mentioned in Section 2.2.2, the Watts-Strogatz model starts from a low-dimensional lattice and uses random rewirings to create the small-world effect. In general, starting from sparse graphs, we can use vertex contraction to increase the degrees of some vertices and also decrease the average distance: this results in a scale-free graph $G_{n,k,\alpha,\gamma}$ with little clustering. Note that our deterministic approach to vertex contraction versus the random mergings in [72] is similar to going from the Watts-Strogatz model to the circulant graph with double-steps [35]. The algorithm generating this graph is specified in Algorithm 6, where $s_i \circ s_j$ denotes the contraction of vertices s_i and s_j, and $replace(s_i, K_n)$ denotes the replacement of vertex s_i by the complete graph K_n while preserving adjacencies; two instances are illustrated in Figure 6.1. k specifies the number of hierarchies (or distinct levels in the degree distribution), α is the amplitude of the effect (*i.e.* the highest degree), and γ is the slope (speed at which the highest degree decreases when creating the next level). Thus, these parameters allow a fine calibration of the power-law degree distribution.

Algorithm 6 $ScaleFree(n, k, \alpha, \gamma) \mapsto G = (V, E)$

1: Let $G \leftarrow C_n$
2: **for** $i = 0...k-1$ **do**
3: $\quad s = s_0 \circ s_{\frac{\gamma^i}{\alpha}}$
4: $\quad$ **for** $j = 2..n \times \frac{\alpha}{\gamma^i}$ **do**
5: $\quad\quad s = s \circ s_{j \times n \times \frac{\gamma^i}{\alpha}}$
6: $\quad replace(s, K_{\gamma^i})$
7: **return** G

Two future directions concerning this model consist of studying the growth of the parameter γ that provides the best approximation to a typical power-law, and having n specify the *final* size of the network rather than the size of the original cycle. Furthermore, this model can be seen as a process. The cycle C_n is the base upon which we contract vertices to

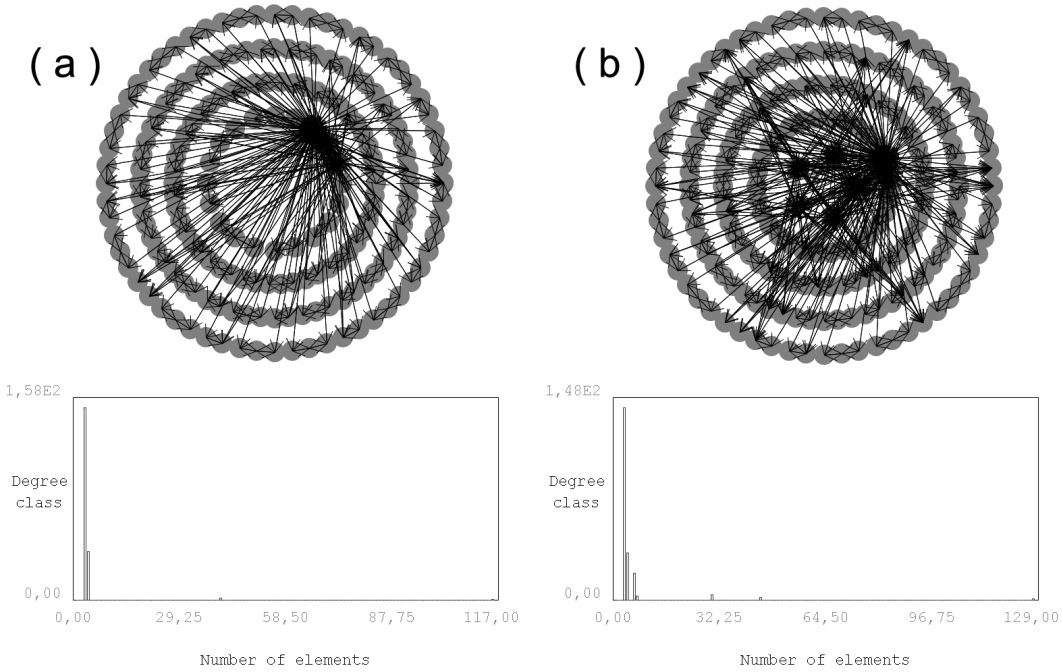

Figure 6.1: The graphs $G_{300,2,0.25,2}$ (a) and $G_{350,3,0.25,2}$ (b), with respectively $N = 191$, $N = 206$ and $\ell = 2.67$, $\ell = 2.64$. Their degree distributions exhibit the scale-free effect.

create higher-degree vertices. The sequence of vertices created has decreasing degrees and each vertex of the sequence is converted to a layer by replacing it with a larger complete graph. This suggests two things: if we also replace the base vertices with a complete graph then we add significant clustering and we obtain a scale-free *and* small-world network. Finally, are there other graphs for which this process is of particular interest? Intuitively, we can see it as an augmentation process that makes a network scale-free, but would particular classes of base graphs provide useful features? As we start using contractions instead of rewirings, are there interesting relations between the minors of a graph and its properties in the extended scale-free version?

To provide a framework for future analysis, we formalize the notion of contractions. The contraction of two vertices $s_1, s_2 \in V(G)$ results in a vertex adjacent to all neighbours of s_1 and s_2. In the following, we will study a sequence $A = \{s_1, ..., s_k\}$ of vertex contractions that results in one vertex s. In other words, if $s_1 \circ s_2$ denotes the contraction of s_1 and s_2 then $s = (((s_1 \circ s_2) \circ s_3) \circ ... \circ s_k)$. An *optimal* sequence is one for which $d(s)$ is the maximum over all possible sequences. A k-sequence is a sequence with k contractions.

Theorem 13. The result of a sequence is independent of the ordering of its contractions.

Proof. The vertex contraction operator $\circ$ is commutative (Corollary 1 from [141]). □

Theorem 14. We denote by s_{i-1} the vertex resulting from a sequence of contractions at step $i-1$. A contraction $s_i = s_{i-1} \circ s_x$ does not improve the result of a sequence if one of the following is true:

(1) s_{i-1} and s_x have all neighbours in common.

(2) s_{i-1} and s_x are adjacent, and s_x has at most one neighbour s_c that is not a neighbour of s_{i-1}.

Proof. (1) If s_{i-1} and s_x are not adjacent, then $d(s_i) = d(s_{i-1}) = d(s_x)$ and there is no improvement. Otherwise, the edge $e_{s_{i-1}s_x}$ is lost and we have $d(s_i) = d(s_{i-1}) - 1 = d(s_x) - 1$ thus the result worsens.

(2) If there is one such neighbour s_c then the addition of edge $e_{s_i s_c}$ is offset by the loss of edge $e_{s_{i-1}s_x}$ thus $d(s_i) = d(s_{i-1})$. Otherwise, the edge $e_{s_{i-1}s_x}$ is lost and $d(s_i) = d(s_{i-1}) - 1$. In both cases, there is no improvement.

□

As an example of a class of graphs that may provide useful features as a base for vertex contractions, we study the n-dimensional hypercube Q_n. Each vertex of the hypercube has a unique label in the vector space $\{Z_2\}^n$, *i.e.* a label $(x_0, ..., x_n)$ with $x_i \in \{0, 1\}$. Two vertices are adjacent if and only if they differ in exactly one coordinate.

Theorem 15. An optimal 2-sequence in Q_n is $A = \{s_1 = (x_0, ..., x_n), s_2 = (\bar{x_0}, ..., \bar{x_n})\}$ and it results in $d(s) = 2n$ when $n > 2$.

Proof. The vertices with labels $(x_0, ..., x_n)$ and $(\bar{x_0}, ..., \bar{x_n})$ are antipodal, *i.e.* at the maximum Hamming distance n, and they do not share a common neighbour for $n > 2$. Furthermore, Q_n is n-regular, so the resulting vertex $s = s_1 \circ s_2$ has n neighbours from s_1 and n different neighbours from s_2, and $d(s) = d(s_1) + d(s_2) = 2n$. □

Example. In Q_4, we first contract vertices with labels (0000) and (1111) into s, since this contraction is optimal by Theorem 15. This is shown in Figure 6.2(b). As all vertices are at distance 2 or less from s, the gain from any further contraction of s with a vertex

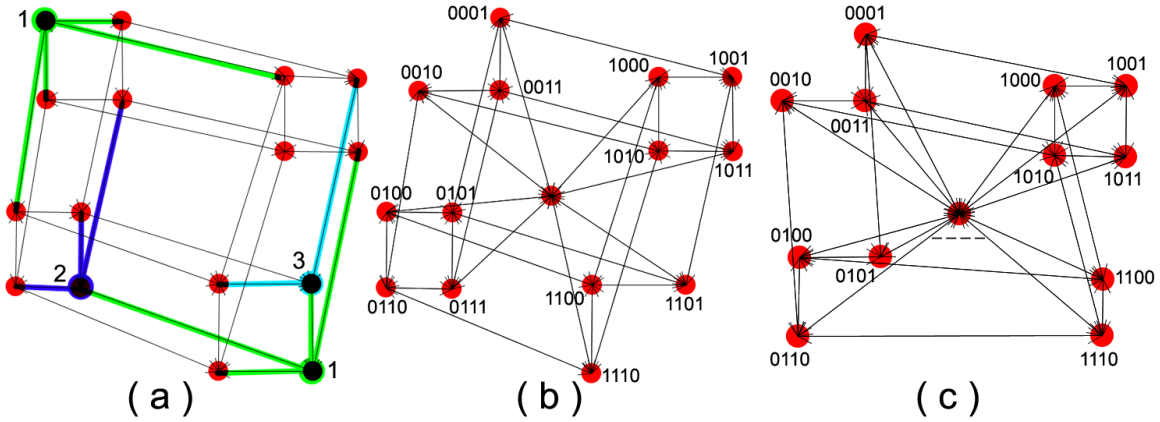

Figure 6.2: The hypercube Q_4 with the four vertices to contract in three steps (a). The result of the first step is shown in (b) and the final result from the third step in (c).

s_c having distinct neighbours from s will be at most $d(s_c) - 1$: either s_c is a neighbour of s and the edge e_{ss_c} is lost, or it has a neighbour s_p in common with s and the edges ss_p and $e_{s_c s_p}$ become duplicates and one is lost. Therefore, the best possible next contraction is between s and a vertex that has at least two neighbours not in common with s. The vertex (0111), numbered 2 in Figure 6.2(a), has 3 neighbours not in common with s and is chosen next, followed by (1101) which has 2 neighbours not in common with s. The result is the sequence $A = \{(0000), (1111), (0111), (1101)\}$, and $d(s) = 11$. The result of this sequence is shown in Figure 6.2(c).

Conjecture. For any optimal k-sequence B, either there exists a $(k+1)$-optimal sequence A that contains it, or no further contraction can improve the sequence.

The intuition behind this conjecture is that an optimal sequence could be constructed by a greedy algorithm in which we select each contraction to be the one that most improves the result. The symmetries of the hypercube permit simple decentralized navigation through the labelling scheme. Thus, a natural question is to ask whether the label of the vertex resulting from a contraction can be chosen in a way that preserves simple decentralized navigation. If there are no constraints on the sequence of contractions, there are three possibilities for navigation: change the navigation algorithm, re-label the graph, or perform additional contractions to recover decentralized navigation. The sequence of contractions can be constrained so that only vertices that differ in one coordinate may be contracted, and this coordinate is replaced by a "don't care" _, indicating that the value can be considered

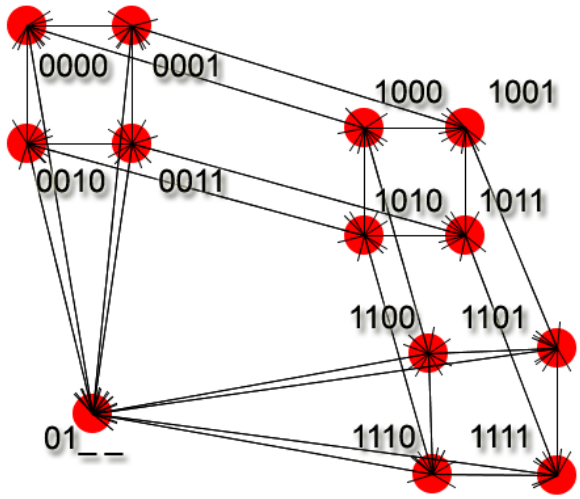

Figure 6.3: A subcube Q_2 in Q_4 is contracted, and two of its coordinates are replaced by "don't care". The result has degree $d(s) = 8$ as it is connected to two other sub-cubes, each with 4 vertices.

to be either 0 or 1 by a neighbour. Note however that the symmetries will not be preserved under contractions and thus the routings will not take the shortest path because they do not know where the shortcuts are. The question is then to find bounds on the sub-optimality resulting from the constraint.

In contractions restricted to minimize navigation changes, we only contract sub-cubes as shown in Figure 6.3. The size of the sub-cube indicates the number of coordinates to replace by "don't care": for example, the vertex resulting from the contraction of a sub-cube Q_2 will have two coordinates replaced by "don't care". Let $d_{Q_p}(s)$ denote the degree of the vertex s resulting from the contraction of the sub-cube Q_p in Q_n, $1 < p < n$. This degree is given by Theorem 16 below.

Theorem 16. $d_{q_p}(s) = 2^p(n-p)$, $1 \leq p < n$.

Proof. A p-dimensional sub-hypercube that is contracted into a single vertex is specified by $n-p$ coordinates. We will say that these $n-p$ coordinate positions are external to the sub-cube and the remaining p positions are internal to the sub-cube. There are 2^p vertices in the sub-cube and all share the same values in the $n-p$ external positions. Each vertex in the sub-cube has p internal neighbours whose labels differ in exactly one internal position and $n-p$ external neighbours whose labels differ in exactly one external position. Thus, there are $2^p(n-p)$ edges between a vertex in the sub-cube and an external neighbour.

We need to show that all of these external neighbours are distinct so that no edges are duplicated when the p-dimensional sub-cube is contracted to a single vertex. Suppose that x and y are any two vertices in the sub-cube and that z is an external neighbour of x. Then the label of z differs from the label of x in exactly one external coordinate position and the labels of x and y differ in at least one internal coordinate position. This means that the labels of y and z differ in one external position and at least one internal position, so y and z cannot be neighbours. □

Thus it is possible for n large enough to create a power-law effect by contracting large sub-p-cubes, and the algorithms for navigation are untouched. In other words, the constraint on the contractions generates a scale-free network from a hypercube and perserves decentralized navigation.

6.2.2 Implementations

Simulation software was designed and presented in Section 3.3. Originally aiming at simple competing heuristics in a network, it was extended to handle dynamic rules and observing agents running a C4.5 algorithm (see Figure 4.4 in Section 4.2). To test the overall system, the methods introduced in the previous chapter have to be implemented, which requires using a library able to handle the intersection of spaces in n dimensions. A compromise could also be found, as typically not more than three parameters would be monitored, and the intersection of 3-dimensional objects is a fundamental operation for many graphics libraries.

6.2.3 Luring component

We define a *luring component* as a subnetwork in which all vertices are monitored by a set of agents. The agents test each vertex periodically, expecting a predefined response; if the response is not as expected, or if there is no response, then the vertex is considered to be corrupted and the agents know that their opponent (e.g. a virus) has been navigating through it. Instead of observing the dynamics, the agents are thus observing their opponents in a component designed to reflect accurately the choices made by the malicious agents that navigate through it. At the end of the observation, the agents expect to be able to characterize the malicious agent with respect to its sight, heuristic and memory.

A (k, d)-tree is designed to reveal if an invader can see up to distance d, with at most $k/2$ neighbours accessed at each round, whether its heuristic is to target high degree nodes, and whether it has memory. It is defined in an algorithmic way by Algorithm 7 and illustrated in Figure 6.4. In the figure, the numbers inside the nodes are their out-degrees. The out-edges that have been omitted to simplify the diagram are all directed towards leaves. If the virus can only see targets at distance 1, then it will choose the level 1 nodes of degrees 6, 5, 4, and 3. If it can see to distance 2, then it will include the degree 2 node so that it can reach the level 2 node of degree 7 and it will exclude the level one node of degree 3. If it can see to distance 3, then it will include the degree 1 node so that it can reach the level 3 node of degree 8. If the virus has some memory, it will avoid targetting some nodes more than once, so the sets of targets will be different in different steps.

Algorithm 7 TreeBuild(k, d)

```
1: Create a new node of degree k called root
2: current ← root
3: assortativity ← true
4: for i = 1 to d do
5:     if outDegree(current) < k then
6:         Create a node with degree i + k − 1 as child of current's nearest sibling
7:         Create a node with degree k as child of current and set it to be the new current
8:     else if assortativity = true then
9:         create k nodes as children of current of increasing degree i to i + k − 1
10:        assortativity ← false
11:        current ← leftmost_child(current)
12:    else
13:        create k nodes as children of current of decreasing degree i + k − 1 to i
14:        assortativity ← true
15:        current ← rightmost_child(current)
```

Agents monitoring different luring components could be exchanging their knowledge in the same fashion as for dynamics, thus the implementation could be adapted for this situation. However, understanding an opponent's strategy is difficult if several malicious agents are roaming the luring component: indeed, our construction allows the observation of an individual agent but being able to dissociate different agents is a major issue. Furthermore, strategies can be mixed, and this should introduce a new parameter that would extend the tree in order to detect specific classes of mixed strategies.

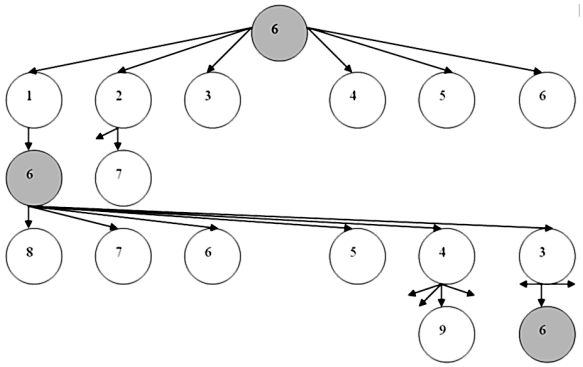

Figure 6.4: A $(6, 4)$-tree.

6.2.4 Key locations for agents

When we discussed ways to exchange data between agents in Section 4.3, an implicit assumption was that two agents would know the path between them. However, paths may get disrupted, particularly in a dynamic network: some of the nodes that constitute it may simply die. In order not to lose information that may result from long observations, an agent should be able to find another agent to receive its information. Given that an agent is not aware of the positions of all other agents, one approach would be that if a path is lost then the agent moves according to *a certain pattern* until it finds another agent. Such patterns might be random moves, or targetted toward high degree vertices, or alternating between high and low degree vertices: in general, a heuristic is used to choose the next target. Thus, the locations for agents in charge of exchanging data should be chosen in such a way that they are found easily using the chosen pattern. In the following, early results are summarized and we discuss the future work that it suggests. We refer to an ideal location as an *entry point*, and we want k entry points.

An obvious first algorithm for placing entry points is a naïve greedy approach (Algorithm 8): rank all of the nodes with respect to a given attribute such as betweenness or closeness centrality and pick the k best ones. The k entry points chosen in this way can be too concentrated to offer good coverage of the network (Figure 6.5), so we designed a second algorithm that ensures better coverage by spacing the nodes (Algorithm 9). Starting from a random node, consider its neighbourhood and select the node maximizing a specified attribute; then select a new random node that hasn't been visited yet and repeat the process. The algorithms use the following notation:

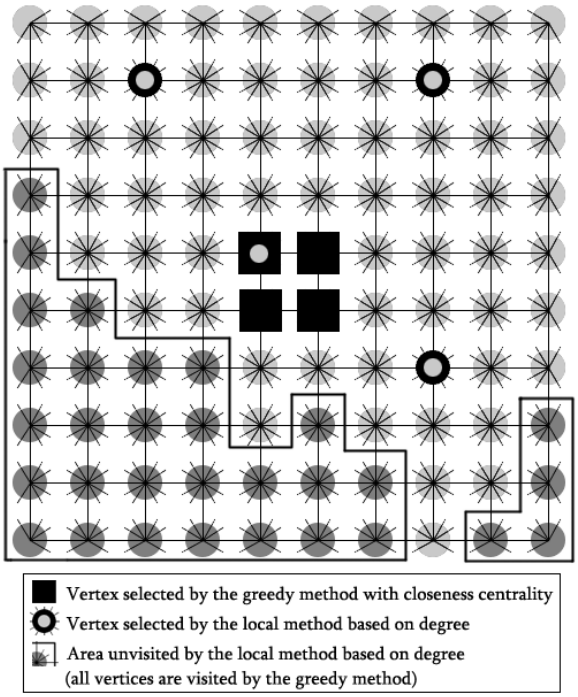

Figure 6.5: Comparing Algorithms 8 and 9 on $G_{10,10}$.

- $G|_{v,s}$ is the subgraph containing the first s nodes that are found by a breadth-first search starting from node v.
- $computeValues(subgraph, a)$ returns an array containing the value of each node of a subgraph with respect to the attribute a. For example, if a is *closeness centrality*, then $value[i]$ is the closeness centrality of the i^{th} node.
- When we look for the next seed, we consider a larger subgraph than the current one, using a coefficient of dilation $\varepsilon > 1$. We used $\epsilon = 1.5$ in our experiments.

Algorithm 8 PLACEGLOBAL(k, attribute a, graph G)

Require: $|V(G)| = n$
1: $values[1...n] \leftarrow computeValues(G, a)$
2: sort $values$
3: **return** the top k $values$

Ideally, each of the k entry points in a graph with n nodes would cover disjoint neighbourhoods with n/k nodes each. This rarely will happen but Algorithm 9 works towards this goal. Note that marking a node as visited simply requires using one bit of the memory

Algorithm 9 PLACELOCAL(k, attribute a, graph G)

Require: $|V(G)| = n$
1: $subgraphSize \leftarrow n/k$
2: ε is the coefficient of dilation
3: $currentNode \leftarrow$ Random in $1...n$
4: **while** $k > 0$ **do**
5: $subgraph \leftarrow G|_{currentNode,subgraphSize}$
6: mark all nodes of $subgraph$ as visited
7: $values[1...n] \leftarrow computeValues(subgraph, a)$
8: $selected \leftarrow selected \cup \{$unselected node with highest $value\}$
9: $k \leftarrow k - 1$
10: $subgraph \leftarrow G|_{currentNode,subgraphSize*\varepsilon}$
11: **if** all of $subgraph$ has been visited **then**
12: $currentNode \leftarrow$ Random node in $subgraph$
13: **else**
14: **while** $currentNode$ has been visited **do**
15: $currentNode \leftarrow$ Random node in $subgraph$
16: **return** $selected$

Table 6.1: Average times for Algorithms 8 and 9 to find a node in S_{opt} for $k = |S_{opt}| = 4$.

Graph	n	Algorithm 8 using closeness centrality	Algorithm 9 using degree
$G_{10,10}$	100	23.97	16.54
$WS_{100,0.003}$	100	23.6	18.45
$R_{100,0.1}$	100	22.04	15.07
$H_{6,3}$	216	6.4	5.44
$C_{100,4,10}$	100	17.83	17.22

available at the node.

Using betweenness or closeness centrality as an attribute for Algorithm 9 did not prove to be efficient as not enough information is provided by a small neighbourhood. To evaluate the centrality measures, we compared a ranking of nodes based on the overall graph with a ranking in which each value was computed in a small neighbourhood. Most of the rankings showed significant differences. In Figure 6.6, the structure of the hierarchical graph and its leaves is clear if computed globally (in black) but remains rather noisy on a local scale (in grey). However, using the node out-degree as an attribute produces good results with Algorithm 9: the average time to find a node in S_{opt} is lower than for Algorithm 8 (see Table 6.1) and a comparison of Figures 6.7(a) and 6.7(b) shows that the time is also more uniform. In

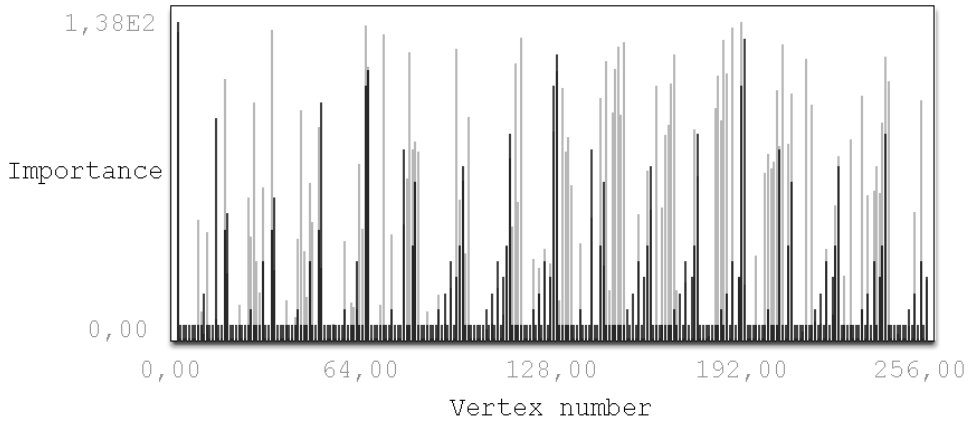

Figure 6.6: Ranking of nodes with betweenness computed globally (black) and at distance 3 (grey)

particular, Figure 6.7(a) shows high variability, while most values in Figure 6.7(b) are clustered in a small range (grey band). Indeed, the algorithm is efficient both in homogeneous networks because of the spacing of the entry points (Figure 6.5) and in more heterogeneous networks because the degree often indicates hubs or shortcuts. The performances of the algorithms were compared on various families of graphs, defined in Section 2.2.2 and representing a broad range of properties: random graph R_p, grid graph $G_{n,n}$, hierarchical graph, circulant graph $C_{n\Delta,h}$ and Watts-Strogatz $W_{n,p}$.

Future work should aim at parallelizing this algorithm so that it is run in a distributed manner by a set of agents, and should compare this distributed algorithm to local-based approaches for centrality introduced in "Local access and preventive setting", Section 2.3. Furthermore, the time needed to find a location clearly depends on the navigation scheme used by the agent who lost a path, thus several navigation schemes could be explored. One can also introduce *guarantees*: every entry point should be within a certain distance of an agent, and a specialized navigation scheme used to find it; if it cannot be found, then the distance acts as a threshold for another navigation scheme.

6.2.5 Decision spaces

A complete implementation of the system as suggested in section 6.2.2 will allow experiments to be conducted for decision spaces. Two types of experiments are of particular interest. Firstly, the accuracy of a decision space obtained from a group of agents can be compared

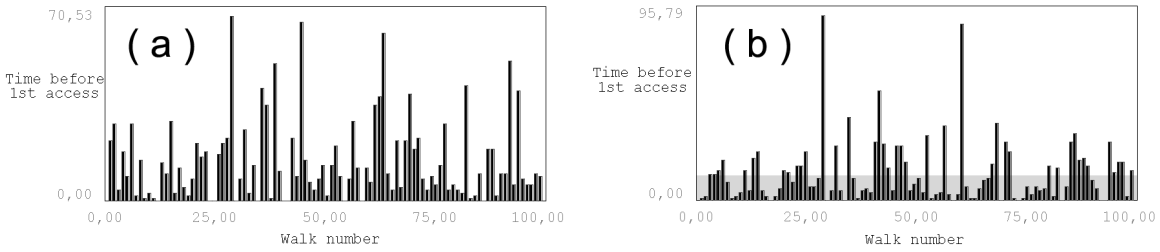

Figure 6.7: Times for Algorithms 8 (a) and 9 (b) to find a node in S_{opt}.

to the classifier of each of these agents, and a factorial design can be used to determine the impact of various factors of the dynamics over the difference in accuracy. Secondly, nodes will keep observing the network even after merging their knowledge with others: the factors obtained from the previous experiments could be used to find appropriate conditions triggering the merging of their new models, with the classical tradeoff between cost and accuracy. Numerous task-specific applications for decision spaces can be found within the context of immunization in dynamic networks: two key families of applications are to decide to whom new nodes will connect (*i.e.* growth) and why nodes die (*shrinkage*). This information can then be used to modify a ranking heuristic that could take into account an estimate of when a node may die, or of the expected growth in a neighbourhood.

Appendix A

Percolation theory

Most of the research on modeling complex networks has been done by physicists working in statistical mechanics (a comprehensive review of this approach can be found in [4]), and one of their most commonly used tools is *percolation theory*. A brief introduction to this theory is important to understand complex networks: not only is it a valuable complement to the theory of random networks described in Section 2.2.2, but it allows an interested reader to have the broad view required by this multi-disciplinary field.

Percolation theory was founded to study the flow of fluid in a porous medium [22]. Concretely, suppose that a porous stone is immersed in a bucket of water [83]: what is the probability that the centre of the stone is moistened? Or, in other words, what is the probability that there exists a path from the centre of the stone to some point on the surface, from which the water can come? To model this problem, we use a network[6]. A *configuration* assigns a weight $w_l = \{0, 1\}$ to each link l: if $w_l = 1$ then the link l is *opened* (*i.e.* water can flow through it); similarly, if $w_l = 0$ then the link l is closed (*i.e.* water cannot flow through it). If we apply this model to two dimensions, we have a grid, or *lattice* (refered to as $\mathbb{Z}^2$ in the literature), in which all links exist with probability p (*i.e.* the weights are assigned indepently from a probability distribution $\mathbb{P}$); see Figure A.1 for two examples of lattices from [59]. This model is similar to the random networks introduced in Section 2.2.2, with a spatial embedding as an additional constraint[7].

[6]Various terminologies can be used. For the sake of clarity, we chose to use network terminology in this thesis. However, "the standard terminology of percolation theory differs from that of graph theory [or networks]: vertices and edges are called *sites* and *bonds*, and components are called *clusters*" [17].

[7]In a network without spatial constraints, any pair of nodes can be connected. In a grid, two nodes can

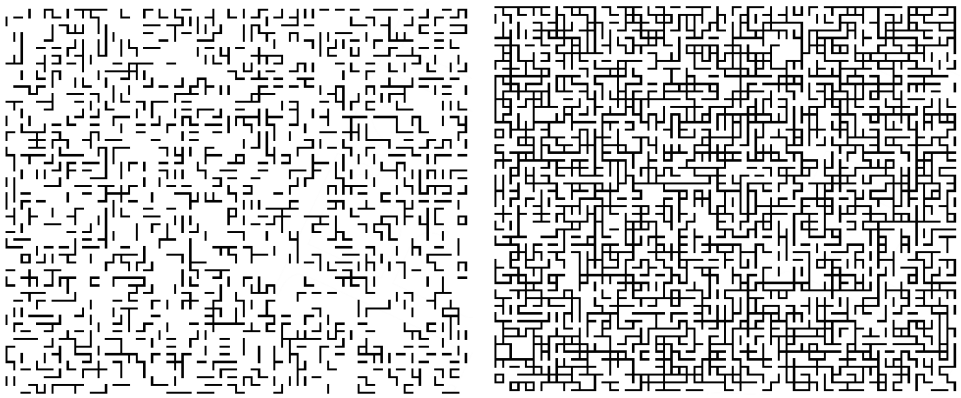

Figure A.1: 50x60 lattices with p = 0.24 and p = 0.51.

A path from node x to node y is denoted $x \to y$. As our aim is to model a porous stone, we are interested in path leading to the surface, and denote them by $x \to \infty$. Our initial question "would the centre of the stone be moistened" can now be defined by "what is the probability $p(0 \to \infty)$ that there exists a path from the centre to the surface". The probability that a node x is on a path leading to the surface is denoted $\Theta_x(p) = p(x \to \infty)$; if all nodes are structurally equivalent, then we can drop x and consider the quantity $\Theta(p)$, known as the *percolation probability*. Clearly, if there are no links for the water to flow through, then the stone cannot be wet; similarly, if the water flows through all links then every node will be wet. Thus, $\Theta(0) = 0$ and $\Theta(1) = 1$. Intuitively, the probability for a node to be wet increases with the probability p of opening a link to the water. Thus, there is a critical probability $0 \leq p_c \leq 1$ such that if $p < p_h$, $\Theta(p) = 0$ and if $p > p_c$ then $\Theta(p) > 0$. The behaviour of the system is very different for $p < p_c$ and $p > p_c$: such sharp transitions are known in physics as *phase transitions* or *critical phenomena*[8]. This phase transition in percolation theory is also seen in random networks as a random network with n nodes has a critical probability $p_c = n^{-1}$ of having a giant component.

---

only be connected if they are physical neighbours. However, this constraint can be waived, as a network with n nodes can be represented with a grid in n dimensions. As random graph theory is interested in the case $n \longmapsto \infty$, we are interested in *infinite-dimension percolation*. Furtunately, there is a critical dimension d_c and results may depend on the dimension d only for $d < d_c$. The results presented in our introduction do not depend on the dimension d and thus are strictly equivalent to what can be found in random graphs.

[8]For a straightforward example of a system with very different behaviours and a sharp transition, one might think about water in a glass: as long as the temperature is roughly above 0C, the water is liquid; when it goes below 0C, the system changes completely as the water becomes ice [11].

Appendix B

Alternative views of Agents

Agents were presented in Section 4.1 in order to understand the characteristics of our system. Numerous aspects of agents were not discussed in Section 4.1 because they were not necessary to understand the design. In this appendix, we will discuss several of these aspects, as they provide a better understanding of the richness of this topic. By reviewing approaches from three different fields, we will show that agents, originated from AI, also appear as *players* in game theory, *actors* in actor theory, and *processes* in computer systems. While traditional reviews regarding agents focus on one of these fields, we will discuss all three together to highlight the advantages offered by each approach.

Game theory [134]. Agents can be considered to be players in a game, which abstracts away details such as the type of language to choose when two agents wants to communicate or how the representation of the world is stored within an agent's memory. This approach is mainly concerned with finding efficient strategies, of which the most well known are the *iterated elimination of dominated actions* (*i.e.* iteratively eliminate all actions for which there is a better action) and the search for a *Nash equilibrium*. Concepts such as communication (for example with *coordination graphs*) or learning can be applied to this model, but often will be defined in mathematical terms rather than through algorithms.

Software engineering [111, 63, 20]. This approach provides deep deep insight into the modules from which agent can be made, with their relations and characteristics, as illustrated with an example in Figure B.1 from [20]. The communication between agents is specified by protocols, whose complexity depends on the actions allowed. An interesting case is Richard Mayr's hierarchy of Process Rewrite Systems [94], which defines the relations between several formalisms in terms of the formal grammars that they allow. Although

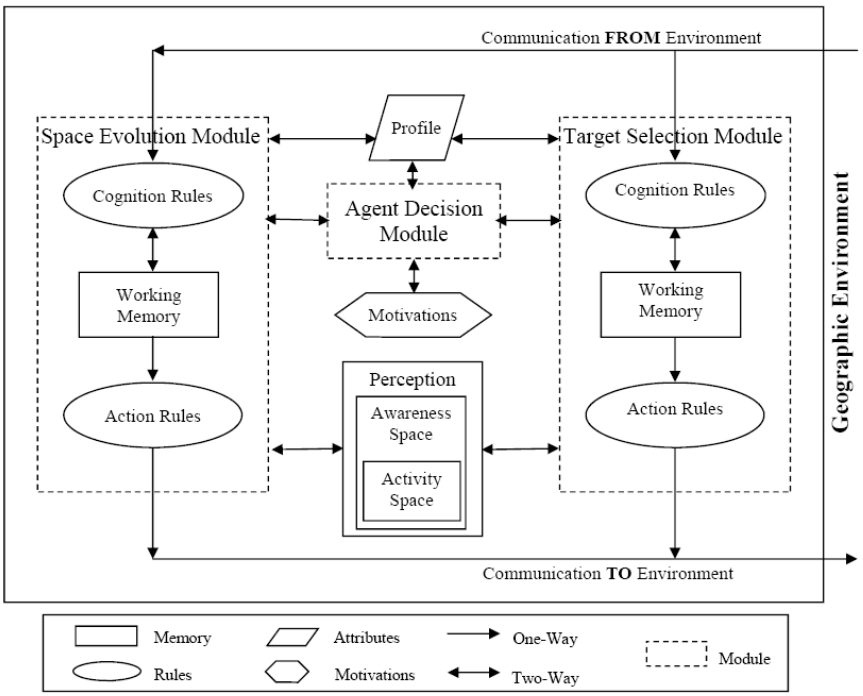

Figure B.1: Deeper insight into an agent: the modules and their relations.

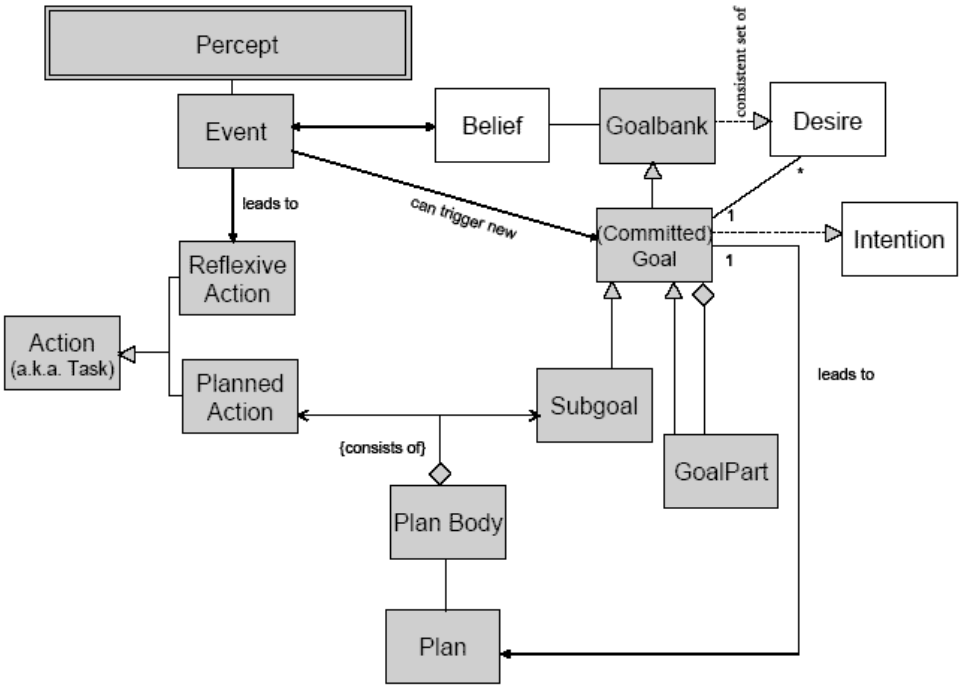

Figure B.2: An adaptation of the BDI architecture.

initially defined for processes, it can easily be used for agents[1]: the grammar $t \rightarrow_a t_1||t_2$ specifies that the event a turns an agent t into an agent t_1 and generates another agent t_2; similarly, $t \rightarrow_a t_1.t_2$ means that action a causes the agent t to wait for the result of the agent t_1 and then resumes as t_2. However, such formalisms are often based on a static topology: all the relations between the agents have to be known at the beginning and cannot evolve. One of the models allowing a dynamic topology is *actor theory* [3]: an *actor* has its own data and procedures, executes concurrently with other actors and communicate asynchronously by sending messages; thus, it corresponds to the definition of an agent. Another model is the *nomadic π-calculus* [140], similar in principles to actor theory with the main difference that an agent is associated with a host site and can migrate between sites during its execution.

Artificial intelligence. Advances in computational intelligence of systems traditionally come from AI, and thus this area provides thorough studies on reasoning agents, *i.e.* agents have a knowledge model that is used by a reasoning engine. The knowledge of an agent cannot be perfect, as it only perceives a fraction of the world and this perception can be further limited by the agent's representation: for example, an agent cannot store all information as it has a limited memory size; thus, the knowledge of an agent is referred to as its *beliefs*. Furthermore, an agent takes a set of actions in order to reach its goal and this can be thought of as turning the initial state of the world into several possible desirable ones; it cannot be guaranteed that the actions of the agent will result in a desirable state and thus we say that an agent expresses *intentions* regarding its *desires*. This led to the BDI architecture, for *Beliefs, Desires, Intentions* of which several adaptations have been proposed, one being showed in Figure B.2 from [7].

Early studies on reasoning for real-world cases have shown the limitation of monotonic logics, in which the number of conclusions can only grow with the addition of new information (hence a monotonic growth). For example, an agent can observe a thousand birds flying and, having to form a general rule, will conclude that all birds fly. However, if it later observes a bird such as a penguin or a bird with a broken wing, that does not fly, then the conclusion should change, either in a probabilistic way (98% of birds fly) or by making more assumptions (all birds that are not abnormal fly). In other words, additional knowledge can restrict the number of conclusions and thus non-monotonic logic is necessary

[1] A process can be considered to be the equivalent of an agent within the context of computers: it is independent, communicates with other processes, and can also benefit from advanced methods such as learning.

to model the reasoning of agents. Another way to consider the problem is by stating that the addition of new beliefs might contradict previous beliefs, and thus the set of beliefs held by the agent should be revised. This approach is called *belief revision* (see [53] for the relationship between belief revision and non-monotonic logic) and it deals with the fundamental problem of keeping an agent's set of beliefs consistent as new observations are made. The early logic framework AGM [6] due to Carlos **A**lchourron, Peter **G**ardenfors and David **M**akinson offers a logic characterization of the properties that a revision operator should obey. The problem of inconsistent beliefs can be extended to the overall MAS: agents have different beliefs and those beliefs may have to be merged in order to provide a more general picture of a phenomenon; the process of solving this problem is called *belief merging*. Recent frameworks [40] have been developed to include both belief merging and belief revision.

Appendix C

Concepts of data-streams

As shown in section 4.4 an immunization system can have the same requirements as the fundamental ones of a data-stream. In this appendix, we discuss three aspects directly related to our system. Firstly, an agent cannot store all records, and thus it needs a data-structure, called a *synopsys data-structure* [56], that is smaller than the base data set. Secondly, data-structures only provide explanation about the low-level part of data streams, as they deal with *representation*: a higher-level part, consisting of an abstract view of streams, it provided by considering the *denotation* [92]. Finally, the application to agents is to learn from their environment, thus we present different methods to learn from data streams.

Formally, an *f(n)-synopsis data structure* for a class Q of queries is a data structure that provides exact or approximate answers to queries from Q, using $O(f(n))$ space for a data set of size n, where $f(n) = n^{\varepsilon}$ for some constant $\varepsilon < 1$. Among the benefits of a sublinear data structure are that we can have enough space left in an agent's memory for other purposes and we can get remote transmissions at minimal cost, as previously mentioned for the advantage of transmiting a decision tree over raw data. The main challenges of a good synopsis are twofold: we have to minimize the loss of accuracy (further specified by the confidance of the approximate response), and we require fast computations so that the structure can be obtained and maintained in minimal time. Note however that the goal is not always to reduce the data to *one* synopsis: an approximate answer engine may maintain several synopses to overcome the difficulties of accuracy for various classes of primitive operations.

A synopsis construction algorithm must satisfy the *one pass constraint*, that the content of the stream can be examined at most once [2]. Further constraints depend on the nature

of the data: either a static but massive data set, or a data stream evolving over time. We will focus on the latter. There are five main construction techniques:

- **Sampling**. This is an unbiased estimate of the data with provable error guarantees, and it is often the best method for high dimensional applications[1]. Typically, we want each point of the stream to have the same probability n/N to be in the sample, where N is the number of total points (not known in advance) and n the desired size of the sample. This is achieved by the *reservoir sampling* method. Alternatives may favour more recent elements, because they give a better picture of the situation with respect to changes: a bias function is used to regulate the sampling and we have a *biased reservoir sampling* method.

- **Histograms**. The buckets of a histogram do not embed any information about the distribution of the data points within them, *i.e.* the distribution is assumed to be uniform. However, the buckets can be of different sizes and the main challenge lies in choosing the sizes. $V-optimal$ histograms can be constructed *offline* using dynamic programming with quadratic time complexity, but they cannot be used in a data stream setting. Instead the $1+\varepsilon$ approximation in [55] can be applied in linear time and polylogarithmic size in the number of items.

- **Wavelets**. One of the challenges is to maintain the coefficients dynamically. This can be done simply through Haar wavelets, in which "higher order coefficients of the decomposition illustrate the broad trends in the data, whereas the more localized trends are captured by the lower order coefficients". The other challenge is the choice of coefficients to keep, as their number is equal to the length of the data stream [**?**, 76].

- **Sketches**. A data point is considered to be a vector v with dimensionality n, and the *random projection* method reduces its dimensionality to k by picking a set of k appropriately chosen random vectors of dimensionality d and calculating the dot product of v with each of these vectors. For example, a random vector might be generated as

[1]Selecting a sample does not depend on the dimensionality whereas other methods that do, such as histograms, face more challenges in those situations. Note however that the dimensionality is not the only criterion for the choice of a method and that the property that one wishes to capture is the primary objective. For example, if one is interested in a property such as the cardinality of the data stream, then a single counter would be more efficient than sampling, regardless of the dimension.

follows [69]: each component is "an independent random variable with normal distribution N(0,1)" (*i.e.* zero mean and unit variance) and the resulting vector is normalized to one unit in magnitude. For example, to get a sketch vector with dimensionality 2 from $t = (2, 1, 3, 1)$, we generate two random vectors $v_1 = (-0.45, -0.09, 0.10, 0.97)$ and $v_2 = (-0.19, 0.73, -0.61, 0.21)$, and calculate the dot products $(0.18, -1.28)$. A more general characterization of sketches, proposed in [36], considers a sketch to be a two-dimensional array for which the content is constructed using hash functions. Sketches require logarithmic space "in the number of distinct items in the stream" and can be used to approximate quantities such as the L_1 and L_2 norms of vectors, to find the most frequent items (called *heavy hitters*), or significant differences between streams. They are also used as a primitive for $V - optimal$ histograms or wavelets.

To evaluate the quality of a synopsis, a metric is chosen according to the technique used. For example, the mean square error is easy to obtain for wavelets by retaining the largest coefficients, but we can have very large errors for some points and thus the maximum error metric is more appropriate. Sampling techniques benefit from statistically proven properties: for example, we have provable bounds for the error and there is a probability δ that the answer is $\varepsilon-$approximate.

An input stream can be seen as the description of a signal A that arrives as a sequence of items $a_1, ..., a_n$, where n can be an arbitrarily large number, *i.e.* the stream is "potentially unbounded in size" [8]. The ordering of the sequence is fixed by the source, thus the receiver has no control over it. There are two main models for a stream. Firstly, we can can consider each item a to be a snapshot of a vector that evolves with time, *i.e.* $a(t) = [a_1(t), ..., a_n(t)]$, with a being the zero vector defined by $\forall i \in \{1, n\}, a_i(0) = 0$. The stream is then a serie of updates for each element, and the update (i_t, c_t) at time t will add the value c_t to the element i_t, which results in the vector:

$$a_j(t) = \begin{cases} a_j(t-1) + c_t & \text{if } j = i_t \\ a_j(t-1) & \text{otherwise} \end{cases}$$

If c_t can only be positive, the model is a *cash register*; if negative values are also allowed, we have a *turnstile*. The latter is subdvidided into a *general* case, in which the values $a_i(t)$ can become negative, and a *non-negative* case if guaranteed by the application[2]. Secondly,

[2]For example, the difference between two cash register streams may yield negative values, whereas in a database "you can only delete a record you inserted" [103].

we can use the time series approach, which models situations in which the data is generated continuously as in network traffic. However, these are only models for the *encoding* of the stream, which do not carry any of the *meaning*. For example, the meaning of a temperature probe is a discrete signal with regular sampling rate, while the encoding can be all the samples (time series) or the differences between two samples (general turnstile). The meaning of the structure provides useful extra information: for example, the ordering in a stream of URLs produced by a web crawl [92] depends on the way the crawler works, and treating such a stream as an unordered collection is not adequate. *Reconstitution functions* [92] were proposed as a transition from representation (*encoding*) to denotation (*semantic*). Concretely, if the items in the stream are of type T and the desired domain of interpretation is D, then the reconstitution consists of successive approximations of D by using sequences of T of increasing sizes. In other words, the stream is not available as a whole, so only successive finite prefixes of the stream can be used and each such prefixe correspond to the sequence of items received so far. We can also consider that the change from T to D consists of changing the structure of items[3], and an example is seen in figure C.1 from [92].

As we now understand what data streams are, we can clarify the main assumptions of this setting. Firstly, we said that data streams are too massive to be stored and linear data structures are prohibited. However, if the signal describing the stream has a small range, such as the age of people, then a linear data structure is not an issue: there would be at most 150 values, regardless of how long the data stream. Thus, we are implicitly considering a signal with a *large* range, such as IP addresses (2^{32} possible values) or http addresses (considered as potentially infinite as there is no limit on an address itself and we can furthermore embed queries into it). Furthermore, items generally have several attributes, hence an overall domain that is even larger. Secondly, we stated that the amount of computation time per item must be low. The rationale is that we cannot control the rate at which data arrives, and we need to avoid a situation in which data is arriving faster than the algorithm can process it. This does not mean that the processing time is *strictly* bounded by the data stream rate: for example, the *load shedding* approach studies how unprocessed data can be dropped while minimizing the loss of accuracy. This has been implemented in systems such as Aurora, in which a *drop* operator randomly drops items when the input rate is too

[3]In reconstitution functions, the structure is fully specified, whereas in data mining it is to be found and is somewhat limited to simpler types (although recent improvements in [80] seem to enable more complicated structures).

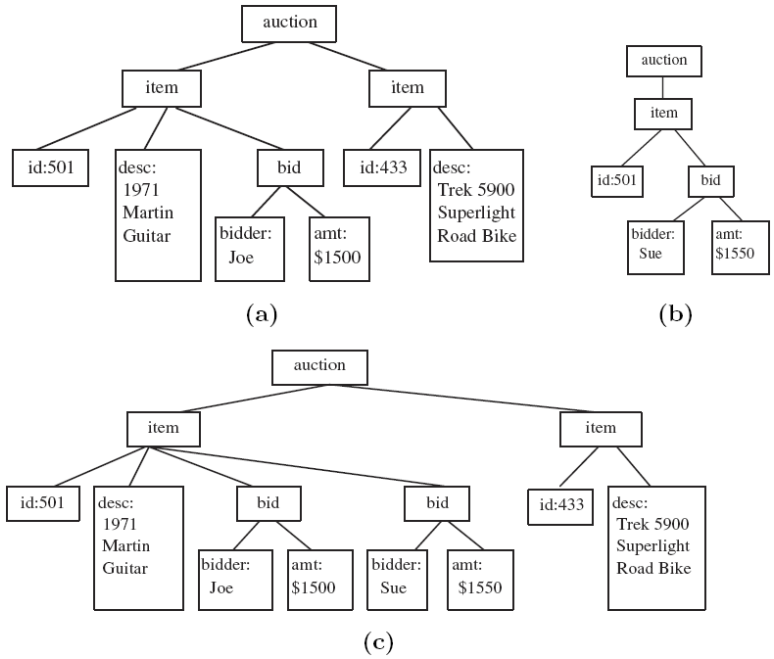

Figure C.1: (a) is the reconstitution of the mathematical structure. When a new item (b) arrives, it is integrated as part of the structure (c).

high [25]. Thus, to be precise, the data stream rate is an upper bound on the time for *exact* computations but may be relaxed by an approximation factor ε; furthermore, amortized analysis should be used when computing the complexity.

Our main application is to learn from data streams. In classical learning, we start by learning a model from a finite dataset, called the *training set*, and we use this model for certain tasks o,vpmvo,g new instances. For example, a hundred instances can be used to generate a classifier which will assign a class label to unlabelled instances. Such models are often static as they are not updated through new instances. The assumption is that instances are generated by a *stationary distribution, as well as being independent and identically distributed* [51]: if this assumption holds, a big enough sample would provide a satisfactory model of the distribution, and this distribution does not change so there is little need to update the model. This key assumption does not hold in the data stream setting: applications detecting money laundering look for *correlations* between transactions (thus it is not independent and identically distributed), while sensor networks work in highly dynamic environments (hence non-stationary). Furthermore, standard techniques of data mining

have a superlinear cost which is prohibited here. Therefore, learning from data streams is drastically different from classical learning: we have a non-stationary distribution, we might have correlations in the data, we are upper-bounded by an amortized linear cost, and the model has to be built as the data arrives. Thus, one is typically looking at an *approximate and incremental learning algorithm*, where 'incremental' is used as a synonymous of *online* or *successive* and means that the model is revised by incorporating the new data[4].

Blind methods are a simple way to achieve incremental learning, by updating the model automatically at regular intervals. However, this is not efficient: either the time frame is small to ensure that most changes are captured, but then there is a cost overhead when there are no changes, or the time frame is large and some changes are missed. Thus, a trade-off is desired between the cost of update and the gain in accuracy and the focus is on change-detection mechanisms [51]. A similar trade-off is needed to take into account that concepts become outdated [135]: if we automatically forget old examples at regular intervals, then either the time frame is small and the model is not trained with enough data (*i.e.* is inaccurate) or the time frame is large and the model is trained on outdated concepts. Finally, note that we are not interested in *all* learning algorithms that are incremental, as "some are reasonably efficient but do not guarantee that the model learned will be similar to the one obtained by [a learning algorithm that does not have to be incremental; furthermore,] they are highly sensitive to example ordering, potentially never recovering from an unfavorable set of early examples" [41]. Thus, we focus on the following techniques that produce results similar to the ones obtained by a traditional algorithm[5]:

- **Support Vector Machine (SVM)** [51]. A data point is an n-dimensional vector, and all data points belong to two possible classes. A SVM classifies the data points by separating them with an $(n-1)$-dimensional hyperplane that leaves maximum margin between the two classes (although this classifier is linear and SVMs can also perform non-linear classification with more complex shapes as separators). The nearest data points to the hyperplane are the *support vectors*, and as they account for a fraction of

[4] *Decremental unlearning* is another central concept and is not opposed to incremental algorithms, but rather means that concepts that have been learnt by the model have to be forgotten when the trend in the data stream changes. The model can be simplified, or the confidence decreased.

[5] For other incremental learning algorithms, see the systems COBWEB or WINNOW produced in the 1980s. Motivation is found in *Incremental learning from noisy data* (Schlimmer and Ganger,1986), in which such algorithms are evaluated on the "cost of updating memory", the "quality of learned concept descriptions" (*i.e.* accuracy) and "the number of observations needed to obtain a stable concept description" (not applicable here because of the non-stationary distribution).

the data points they can be used as a summary. The incremental algorithm in [129] used this observation for massive datasets: the dataset is partitioned into blocks, an SVM classifier trained on each block, and the support vectors are added to the next block. A comparison with a training on the whole dataset showed a difference in accuracy of only 1.6% on average and 7.65% at most. These results were improved in [26], in which decremental unlearning was used through a standard procedure called Leave-one-out (LOO). However, there are still too many support vectors to store them explicitly: for example, the Ripley data set contains 250 data points and a linear classifier yields 89 data points, which is far from the requirements of polylogarithmic space. Finally, this approach satisfies the needs of incremental algorithms, but was designed for massive datasets and the extension to data streams is still being developed.

- **Decision trees**. To compare two decision tree learners, one can estimate the probability that the learners choose a different split at a node. Hoeffding trees were proposed in [41] as an incremental tree learner, and the probability that they choose a different split at a node from a non-incremental tree learner such as C4.5 decreases exponentially with the number of samples. The idea is that in a classical setting with a stationary distribution and independence of the examples, it "may be sufficient to consider only a small subset of the training examples that pass through a node" to find a split. The idea was implemented by the Very Fast Decision Tree algorithm and the method was improved in [73] by reducing the execution time for numerical attributes with fewer samples while maintaining the same probabilistic bound. Concept-Adapting VFDT system (CVFDT) [67] extends VFDT to data streams: the counter of a node to which new data corresponds is incremented, and decremented for old data. In a stationary distribution, the operations would counter-balance each other on average, but in a non-stationary distribution, the gain of information obtained by splitting on an attribute could change. In this case, CVFDT grows an 'alternative' subtree with the new best split. Each node s with at least one alternative subtree is tested periodically: alternate subtrees for which the accuracy does not increase with time are eliminated, and if a subtree has better accuracy than the original subtree then it replaces it. CVFDT is four time slower than VFDT according to experiments, but provides better models: they have smaller average error rate and increase more smoothly with a change of concept (see figure C.2 from [67]). A very different approach is the Time Stamp

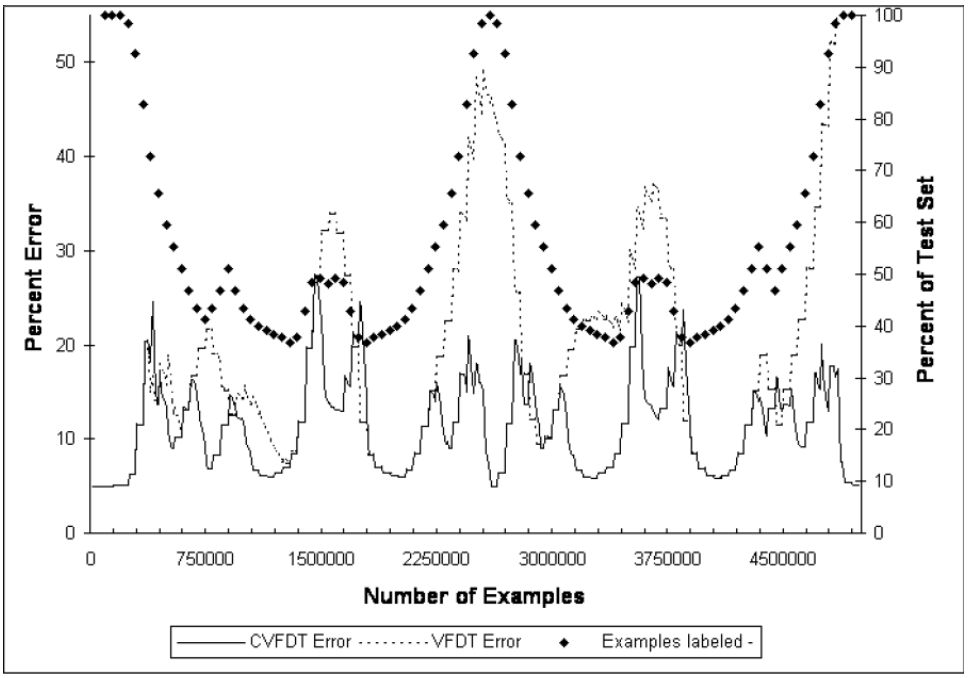

Figure C.2: Error rates of CVFDT and VFDT on a non-stationary distribution.

Attribute Relevance (TSAR), consisting of having the time stamp as an additional attribute used in the data mining process [66].

- **Ensemble**. To approach the problem of outdated data in the model, it was proposed to use the class distribution: "historical data whose class distributions are similar to that of current data can reduce the variance of the current model and increase classification accuracy" [135]. To achieve this goal, the authors trained a set (or *ensemble*) of models instead of only one: while a single classifier outputs $f_c(y)$ as the probability of the instance y belonging to class c, an ensemble of classifiers outputs an average over the $\alpha_i \times f_c^i(y)$, where i is the *i-th* classifier and α_i its weight in the average. It was proven that the ensemble can yield lower classification error than a single classifier, and the efficiency of this approach was confirmed experimentally.

Bibliography

[1] Charu C. Aggarwal. An introduction to data streams. In *Data Streams*, pages 1–8. Springer US, 2007.

[2] Charu C. Aggarwal and Philip S. Yu. A survey of synopsis construction in data streams. In *Data Streams*, pages 169–207. Springer US, 2007.

[3] Gul A. Agha, Prasannaa Thati, and Reza Ziaei. Actors: a model for reasoning about open distributed systems. *Formal methods for distributed processing: a survey of object-oriented approaches*, pages 155–176, 2001.

[4] Reka Albert and Albert-Laszlo Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[5] Reka Albert, Hawoong Jeong, and Albert-Laszlo Barabasi. Error and attack tolerance of complex networks. *Nature*, 406:378–382, 2000.

[6] C. E. Alchourron, P. Gardenfors, and D. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of symbolic logic*, 50(2):510–530, 1985.

[7] J. Debenham B. Henderson-Sellers, Q.N. Numi Tran. An etymological and metamodel-based evaluation of the terms 'goals and tasks' in agent-oriented methodologies. *Journal of Object Technology*, 4(2):131–150, 2005.

[8] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In Lucian Popa, editor, *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, pages 1–16. ACM, 2002.

[9] James P. Bagrow, Erik M. Bollt, Joseph D. Skufca, and Daniel Ben-Avraham. Portraits of complex networks. *Europhysics letters*, 81(6), 2008.

[10] Amir Bar-Or, Daniel Keren, Assaf Schuster, and Ran Wolff. Hierarchical decision tree induction in distributed genomic databases. *IEEE Trans. Knowl. Data Eng.*, 17(8):1138–1151, 2005.

[11] Albert-Laszlo Barabasi. *Linked: How everything is connected to everything else and what it means.* Plume, 2003.

[12] Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science*, 286, 1999.

[13] Lali Barriere, Francesc Comellas, and Cristina Dalfo. Deterministic hierarchical networks. *Preprint Universitat Politecnica de Catalunya*, 2006.

[14] Springer Berlin, editor. *On Reshaping of Clustering Coefficients in Degree-Based Topology Generators*, 2004.

[15] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424:175–308, 2006.

[16] M. C. Boily, Z. Asghar, T. Garske, A. C. Ghani, and R. Poulin. Influence of selected formation rules for finite population networks with fixed macrostructures: Implications for individual-based model of infectious diseases. *Mathematical Population Studies*, 14(4):237–267, 2007.

[17] Bela Bollobas and Oliver Riordan. *Percolation.* Cambridge University Press, 2006.

[18] Ulrik Brandes and Thomas Erlebach, editors. *Network analysis.* Springer Verlag, 2005.

[19] Ulrik Brandes and Dorothea Wagner. visone - analysis and visualization of social networks. pages 321–340. Springer-Verlag, 2004.

[20] P. L. Brantingham, Uwe Glässer, B. Kinney, K. Singh, and Mona Vajihollahi. Modeling urban crime patterns: Viewing multi-agent systems as abstract state machines. In *Abstract State Machines*, pages 101–118, 2005.

[21] Fred Brauer and Carlos Castillo-Chavez. *Mathematical Models in Population Biology and Epidemiology.* Springer, 2001.

[22] S. R. Broadbent and J. M. Hammersley. Percolation processes. i. crystals and mazes. *Proc. Cambridge. Philos. Soc.*, 53:629–641, 1957.

[23] Guido Caldarelli and Alessandro Vespignani, editors. *Large Scale Structure and Dynamics of Complex Networks: From Information Technology to Finance and Natural Science.* World Scientific Publishing Company, 2007.

[24] D. S. Callaway, M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Network robustness and fragility: Percolation on random graphs. *Physical Review Letters*, 85:5468–5471, 2000.

[25] Donald Carney, Ugur Çetintemel, Mitch Cherniack, Christian Convey, Sangdon Lee, Greg Seidman, Michael Stonebraker, Nesime Tatbul, and Stanley B. Zdonik. Monitoring streams - a new class of data management applications. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*, pages 215–226. Morgan Kaufmann, 2002.

[26] Gert Cauwenberghs and Tomaso Poggio. Incremental and decremental support vector machine learning. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS), Denver, CO, USA*, pages 409–415. MIT Press, 2001.

[27] Phillip K. Chan and Salvatore J. Stolfo. Toward parallel and distributed learning by meta-learning. In *Working Notes AAAI Work. Knowledge Discovery in Databases*, pages 227–240, 1993.

[28] Li-Chiou Chen and Kathleen M. Carley. The impact of countermeasure propagation on the prevalence of computer viruses. *IEEE Transactions on Systems, Man, and Cybnertics-Part B: Cybernetics*, 34(2), 2004.

[29] Yiping Chen, Gerald Paul, Shlomo Havlin, Fredrik Liljeros, and H. Eugene Stanley. Finding a better immunization strategy. *Physical Review Letters*, 101, 2008.

[30] Fan R. K. Chung. *Spectral Graph Theory*, volume 92. American Mathematical Society, 1997.

[31] Reuven Cohen, Shlomo Havlin, and Daniel ben Avraham. Efficient immunization strategies for computer networks and populations. *Physic Review Letters*, 91(24), 2003.

[32] Francesc Comellas. Complex networks: Deterministic models. *Physics and Theoretical Computer Science*, 7:275–293, 2007.

[33] Francesc Comellas, Guillaume Fertin, and Andre Raspaud. Recursive graphs with small-world scale-free properties. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 69, 2004.

[34] Francesc Comellas, Javier Ozon, , and Joseph G. Peters. Deterministic small-world communication networks. *Inf. Process. Lett.*, 76(1-2):83–90, 2005.

[35] Francesc Comellas, Javier Ozon, and Joseph G. Peters. Deterministic small-world communication networks. *Information Processing Letters*, 76:83–90, 2000.

[36] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[37] Pascal Crepey, Fabian Alvarez, and Marc Barthelemey. Epidemic variability in complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 73(4), 2006.

[38] Razvan Cristescu, Baltasar Beferull-Lozano, and Martin Vetterli. On network correlated data gathering. *IEEE Infocom*, 2004.

[39] P. Davidsson. Agent based social simulation: A computer science view. *Journal of Artificial Societies and Social Simulation*, 5(1), 2002.

[40] James P. Delgrande, Jérôme Lang, and Torsten Schaub. Belief change based on global minimisation. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2468–2473, 2007.

[41] P. Domingos and G. Hulten. Mining high-speed data streams. In *Knowledge Discovery and Data Mining*, pages 71–80, 2000.

[42] Philippe Duchon, Nicolas Hanusse, Emmanuelle Lebhar, and Nicolas Schabanel. Could any graph be turned into a small world? *Theoretical Computer Science, special issue on Complex Networks*, 355(1):96–103, 2006.

[43] Victor M. Eguiluz and Konstantin Klemm. Epidemic threshold in structured scale-free networks. *Physical Review Letters*, 89, 2002.

[44] Paul Erdos and Alfred Renyi. On random graphs. *Publicationes Mathematicae*, 6, 1959.

[45] Usama M. Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. Knowledge discovery and data mining: Towards a unifying framework. In *Knowledge Discovery and Data Mining*, pages 82–88, 1996.

[46] Francisco J. Ferrer-Troyano, Jesús S. Aguilar-Ruiz, and José Cristóbal Riquelme Santos. Incremental rule learning and border examples selection from numerical data streams. *J. UCS*, 11(8):1426–1439, 2005.

[47] Peter Ferrie, Frederic Perriot, and Peter Szor. Worm wars. *Virus Bulletin*, pages 5–8, 2003.

[48] Johannes Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[49] Silvia Gago. Eigenvalue distribution in scale free graphs. *Preprint Universitat Politcnica de Catalunya*, 2007.

[50] Andrea Galeotti, Sanjeev Goyal, Matthew O. Jackson, Fernando Vega-Redondo, and Leeat yariv. Network games. *Review of Economic Studies (forthcoming)*, 2008.

[51] Joao Gama and Rasmus Ulslev Pedersen. Predictive learning in sensor networks. In *Learning from Data Streams: Processing Techniques in Sensor Networks*, pages 143–164. Springer, 2007.

[52] Benoit Garbinato, Denis Rochat, and Marco Tomassini. Impact of scale-free topologies on gossiping in ad hoc networks. In *Sixth IEEE International Symposium on Network Computing and Applications (NCA)*, 2007.

[53] Peter Gardenfors. Belief revision: A vade-mecum. *Lecture Notes in Computer Science*, 649:1–10, 1992.

[54] Leszek Gasieniec, Ralf Klasing, Russell Martin, Alfredo Navarra, and Xiaohui Zhang. Fast periodic graph exploration with constant memory. *SIROCCO, LNCS*, 4474:26–40, 2007.

[55] Johannes Gehrke, Flip Korn, and Divesh Srivastava. On computing correlated aggregates over continual data streams. In *SIGMOD Conference*, pages 13–24, 2001.

[56] Phillip B. Gibbons and Yossi Matias. Synopsis data structures for massive data sets. In *SODA*, pages 909–910, 1999.

[57] Michelle Girvan and Mark E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[58] Mark Granovetter. The strength of weak ties. *American Journal of Sociology*, 78(6), 1973.

[59] Geoffrey Grimmet. *Percolation, 2nd edition.* Springer, 1999.

[60] Peter Gvozdjak and Joseph G. Peters. Gossiping in inclined leo satellite networks. In Cyril Gavoille, Jean-Claude Bermond, and André Raspaud, editors, *SIROCCO'99, 6th International Colloquium on Structural Information & Communication Complexity, Lacanau-Ocean, France, 1-3 July, 1999*, pages 166–180. Carleton Scientific, 1999.

[61] L. O. Hall, N. Chawla, and K. W. Bowyer. Decision tree learning on very large data sets. In *IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 2579–2584, 1998.

[62] Nicolas Hanusse, Evangelos Kranakis, and Danny Krizanc. Searching with mobile agents in networks with liars. *Lecture Notes in Computer Science*, 1900:583–590, 2000.

[63] Aaron Hector and V. Lakshmi Narasimhan. A new classification scheme for software agents. In *3rd International Conference on Information Technology and Applications (ICITA 2005)*, pages 191–196, 2005.

[64] Herbert W. Hethcote. The mathematics of infectious diseases. *SIAM Review*, 42(4):599–653, 2000.

[65] Herbert W. Hethcote, Harlan W. Stech, and Pauline Van Den Driessche. Nonlinear oscillations in epidemic models. *SIAM J. Appl. Math.*, 40(1), 1981.

[66] Ray J. Hickey and Michaela M. Black. Refined time stamps for concept drift detection during mining for classification rules. In John F. Roddick and Kathleen Hornsby, editors, *Temporal, Spatial, and Spatio-Temporal Data Mining, First International Workshop TSDM 2000 Lyon, France, September 12, 2000, Revised Papers*, volume 2007 of *Lecture Notes in Computer Science*, pages 20–30. Springer, 2001.

[67] Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining time-changing data streams. In *KDD*, pages 97–106, 2001.

[68] Earl B. Hunt, Janet Marin, and Philip J. Stone. *Experiments in Induction*. Academic Press, 1966.

[69] Piotr Indyk, Nick Koudas, and S. Muthukrishnan. Identifying representative trends in massive time series data sets using sketches. In Amr El Abbadi, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter, and Kyu-Young Whang, editors, *VLDB 2000, Proceedings of 26th International Conference on Very Large Data Bases, September 10-14, Cairo, Egypt*, pages 363–372. Morgan Kaufmann, 2000.

[70] Raj Jain. *The art of computer systems performance analysis*. John Wiley & Sons, Inc., New York, 2002.

[71] M. Jamali and H. Abolhassani. Different aspects of social network analysis. In *Web Intelligence, 2006. WI 2006. IEEE/WIC/ACM International Conference on*, pages 66–72, 2006.

[72] B. K. Jim, A. Trusina, P. Minnhagen, and K. Sneppen. Self organized scale-free networks from merging and regeneration. *The European Physical Journal B - Condensed Matter and Complex Systems*, 43(3):369–372, 2005.

[73] Ruoming Jin and Gagan Agrawal. Efficient decision tree construction on streaming data. In Lise Getoor, Ted E. Senator, Pedro Domingos, and Christos Faloutsos, editors, *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27*, pages 571–576. ACM, 2003.

[74] UNAIDS:the joint United Nations programme on HIV/AIDS. Revised hiv estimates. *http://data.unaids.org/pub/EPISlides/2007*, 2007.

[75] Jonathan Jordan. The degree sequence and spectra of scale-free random graphs. *Random structures and Algorithms*, 29(2):226–242, 2006.

[76] Panagiotis Karras and Nikos Mamoulis. One-pass wavelet synopses for maximum-error metrics. In Klemens Böhm, Christian S. Jensen, Laura M. Haas, Martin L. Kersten, Per-Åke Larson, and Beng Chin Ooi, editors, *Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, August 30 - September 2*, pages 421–432. ACM, 2005.

[77] Michael Kearns and Siddharth Suri. Networks preserving evolutionary equilibria and the power of randomization. *Proceedings of the 7th ACM conference on Electronic commerce*, pages 200–207, 2006.

[78] Matt Keeling. Course notes on mathematics of epidemiology. *MSc Maths, University of Warwick*, 2003.

[79] Evelyn Fox Keller. Revisiting 'scale-free' networks. *BioEssays*, 27(10):1060–1068, 2005.

[80] Charles Kemp and Joshua B. Tenenbaum. The discovery of structural form. *Proceedings of the National Academy of Sciences*, 105(31), 2008.

[81] A. G. Mc Kendrick. Applications of mathematics to medical problems. *Proceedings of the Edinburgh Mathematical Society*, 14:98 – 130, 1926.

[82] Jeffrey O. Kephart and Steve R. White. Directed-graph epidemiological models of computer viruses. *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, pages 343–359, 1991.

[83] Harry Kesten. What is percolation? *Notices of the Americal Mathematical Society*, 53(5), 2006.

[84] Judith Kleinfeld. Six degrees: Urban myth? *Psychology Today*, 2002.

[85] Jan Kostka, Yvonne Anne Oswald, and Roger Wattenhofer. Word of mouth: Rumor dissemination in social networks. In *Lecture Notes in Computer Science*, volume 5058, pages 185–196, 2008.

[86] Mirjam Kretzschmar and Jacco Wallinga. Networks in epidemiology. *Mathematical Population Studies*, 14(4):203–209, 2007.

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

[88] D. T. Lee. Maximum clique problem of rectangle graphs. *Advances in Computing Research*, 1:91–107, 1983.

[89] Deyi Li and Yi Du. *Artificial Intelligence with Uncertainty.* Chapman and Hall, 2008.

[90] Lun Li, David Alderson, Reiko Tanaka, John C. Doyle, and Walter Willinger. Towards a theory of scale-free graphs: Definition, properties, and implications (extended version). *CoRR*, abs/cond-mat/0501169, 2005.

[91] Ulrike Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, 2007.

[92] David Maier, Jin Li, Peter A. Tucker, Kristin Tufte, and Vassilis Papadimos. Semantics of data streams and operators. In Thomas Eiter and Leonid Libkin, editors, *Database Theory - ICDT 2005, 10th International Conference, Edinburgh, UK, January 5-7, Proceedings*, volume 3363 of *Lecture Notes in Computer Science*, pages 37–52. Springer, 2005.

[93] Sergei Maslov. Role model for modules. *Nature Physics*, 3:18–19, 2007.

[94] R. Mayr. Process rewrite systems. *Information and Computation*, 156:264–286, 1999.

[95] Andreas Meier and Volker Sorge. Exploring properties of residue classes. *Symbolic computation and automated reasoning*, pages 175–190, 2001.

[96] Stanley Milgram. The small world problem. *Psychology Today*, pages 60–67, 1967.

[97] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: Simple building blocks of complex networks. *Science*, 298:824–827, 2002.

[98] Ron Milo, Shalev Itzkovitz, Nadav Kashtan, Reuven Levitt, Shai Shen-Orr, Inbal Ayzenshtat, Michal Sheffer, and Uri Alon. Superfamilies of evolved and designed networks. *Science*, 303:1538–1542, 2004.

[99] Joseph S. B. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM Journal on Computing*, 28(4):1298–1309, 1999.

[100] Michael Molloy and Bruce A. Reed. A critical point for random graphs with a given degree sequence. *Random Struct. Algorithms*, 6(2/3):161–180, 1995.

[101] James Moody. Race, school integration, and friendship segregation in america. *The American Journal of Sociology*, 107(3):679–716, 2001.

[102] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet quarantine: Requirements for containing self-propagating code. *Proceedings of the IEEE Infocom*, 2003.

[103] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2), 2005.

[104] M. E. J. Newman. Power laws, pareto distributions and zipf's law. *Contemporary Physics*, 46:323–351, 2005.

[105] M. E. J. Newman, D. J. Watts, and S. H. Strogatz. Random graph models of social networks. *Proceedings of the National Academy of Sciences*, 99(1):2566–2572, 2002.

[106] Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[107] Mark E. J. Newman, Duncan J. Watts, and Steven H. Strogatz. Random graphs with arbitrary degree distributions and their applications. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 64, 2001.

[108] BBC News. Clock ticking on worm attack code. 2009.

[109] David M. Nicol and Michael Liljenstam. Models of active worm defenses. *Proceedings of the Measurement, Modeling and Analysis of the Internet (IMA Workshop '04)*, 2004.

[110] D. Nojiri, J. Rowe, and K. Levitt. Cooperative response strategies for large scale attack mitigation. *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition*, 2003.

[111] H. S. Nwana. Software agents: An overview. *Knowledge Engineering Review*, 11:205–224, 1996.

[112] American Institute of Physics, editor. *Chaos, Focus issue: Synchronization in complex networks*, volume 18. 2008.

[113] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 2005.

[114] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 65, 2002.

[115] Brian D. Peer and Eric K. Bollinger. Common grackle (quiscalus quiscula). *The Birds of North America Online*, 1997.

[116] Carlo Piccardi and Renata Casagrandi. Inefficient epidemic spreading in scale-free networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 77, 2008.

[117] J. Quinlan. Improved use of continuous attributes in c4.5. *Journal of Artificial Intelligence Research*, 4:77–90, 1996.

[118] Anatol Rapoport. Contribution to the theory of random and biased nets. *Bulletin on Mathematical Biology*, 19(4), 1957.

[119] Erzsebet Ravasz and Albert-Laszlo Barabasi. Hierarchical organization in complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 67, 2003.

[120] Arnold L. Rosenberg. *Graph Separators, with Applications*. Kluwer Academic, 2001.

[121] Hernan Rozenfeld, Joseph Kirk, Erik Bollt, and Daniel ben Avraham. Statistics of cycles: How loopy is your network? *Journal of Physics A: Mathematical and Theoretical*, 38:4589–4595, 2005.

[122] S. J. Russell and P. Norvig. *Artificial Intelligence: a Modern Approach (2nd Edition).* Prentice Hall, 2003.

[123] Jari Samaraki, Mikkko Kivela, Jukka-Pekka Onnela, Kimmo Kaski, and Janos Kertesz. Generalizations of the clustering coefficient to weighted complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 75(2), 2006.

[124] Herbert A. Simon. On a class of skew distribution functions. *Biometrika*, 42:425–440, 1955.

[125] Chaoming Song, Shlomo Havlin, and Hernan A. Makse. Self-similarity of complex networks. *Nature*, 433:392–395, 2005.

[126] Alexandre O. Stauffer and Valmir C. Barbosa. A dissemination strategy for immunizing scale-free networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 76, 2006.

[127] Steven H. Strogatz. *SYNC: The Emerging Science of Spontaneous Order.* Hyperion, 2003.

[128] Toshiharu Sugawara and Satoshi Kurihara. Learning message-related coordination control in multiagent systems. In Chengqi Zhang and Dickson Lukose, editors, *Multi-Agent Systems: Theories, Languages, and Applications, 4th Australian Workshop on Distributed Artificial Intelligence, Brisbane, Queensland, Australia, July 13, 1998, Selected Papers*, volume 1544 of *Lecture Notes in Computer Science*, pages 29–44. Springer, 1998.

[129] Nadeem Ahmed Syed, Huan Liu, and Kah Kay Sung. Incremental learning with support vector machines. In *Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99), Stockholm, Sweden*, 1999.

[130] Andreas L. Symeonidis and Pericles A. Mitkas. *Agent intelligence through data mining*, volume 14. Springer Science, 1995.

[131] Douglas B. Terry, David Goldberg, David A. Nichols, and Brian M. Oki. Continuous queries over append-only databases. In Michael Stonebraker, editor, *Proceedings of the 1992 ACM SIGMOD International Conference on Management of Data, San Diego, California, June 2-5*, pages 321–330. ACM Press, 1992.

[132] Hiroshi Toyoizumi and Atsuhi Kara. Predators: Good will mobile codes combat against computer viruses. *New Security Paradigms Workshop (ACM)*, 2002.

[133] Alexei Vazquez, Balazs Racz, Andras Lukacs, and Albert-Laszlo Barabasi. Impact of nonpoissonian activity patterns on spreading processes. *Physics Review Letters*, 98, 2007.

[134] Nikos Vlassis. *A Concise Introduction to Multiagent Systems and Distributed Artificial Intelligence*. Morgan and Claypool, 2007.

[135] Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 226–235. ACM, 2003.

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

[137] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham. A taxonomy of computer worms. *WORM'03 (ACM)*, 2003.

[138] Songjie Wei, Jelena Mirkovic, and Martin Swany. Distributed worm simulation with a realistic internet model. *Proceedings of the 19th Workshop on Principles of Advanced and Distributed Simulation*, pages 71–79, 2005.

[139] E. J. Wittmann and M. Baylis. Climate change: Effects on culicoides-transmitted viruses and implications for the uk. *The Veterinary Journal*, 160:107 – 117, 2000.

[140] Pawel Wojciechowski and Peter Sewell. Nomadic pict: Language and infrastructure design for mobile agents. In *First International Symposium on Agent Systems and Applications (ASA'99)/Third International Symposium on Mobile Agents (MA'99)*, Palm Springs, CA, USA, 1999.

[141] Thomas Wolle and Hans L. Bodlaender. A note on edge contraction. *Technical Report UU-CS-2004-028*, 2004.

[142] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[143] R. Xulvi-Brunet and I. M. Sokolov. Construction and properties of assortative random networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 70, 2004.

[144] Haijun Zhou and Reinhard Lipowsky. Network brownian motion: A new method to measure vertex-vertex proximity and to identify communities and subcommunities. *Computational Science - ICCS 2004 (LNCS 3038)*, pages 1062–1069, 2004.

[145] Cliff Changchun Zou, Weibo Gong, and Don Towsley. Code red worm propagation modeling and analysis. *Proceedings of the 9th ACM conference on Computer and communications security*, pages 138–147, 2002.