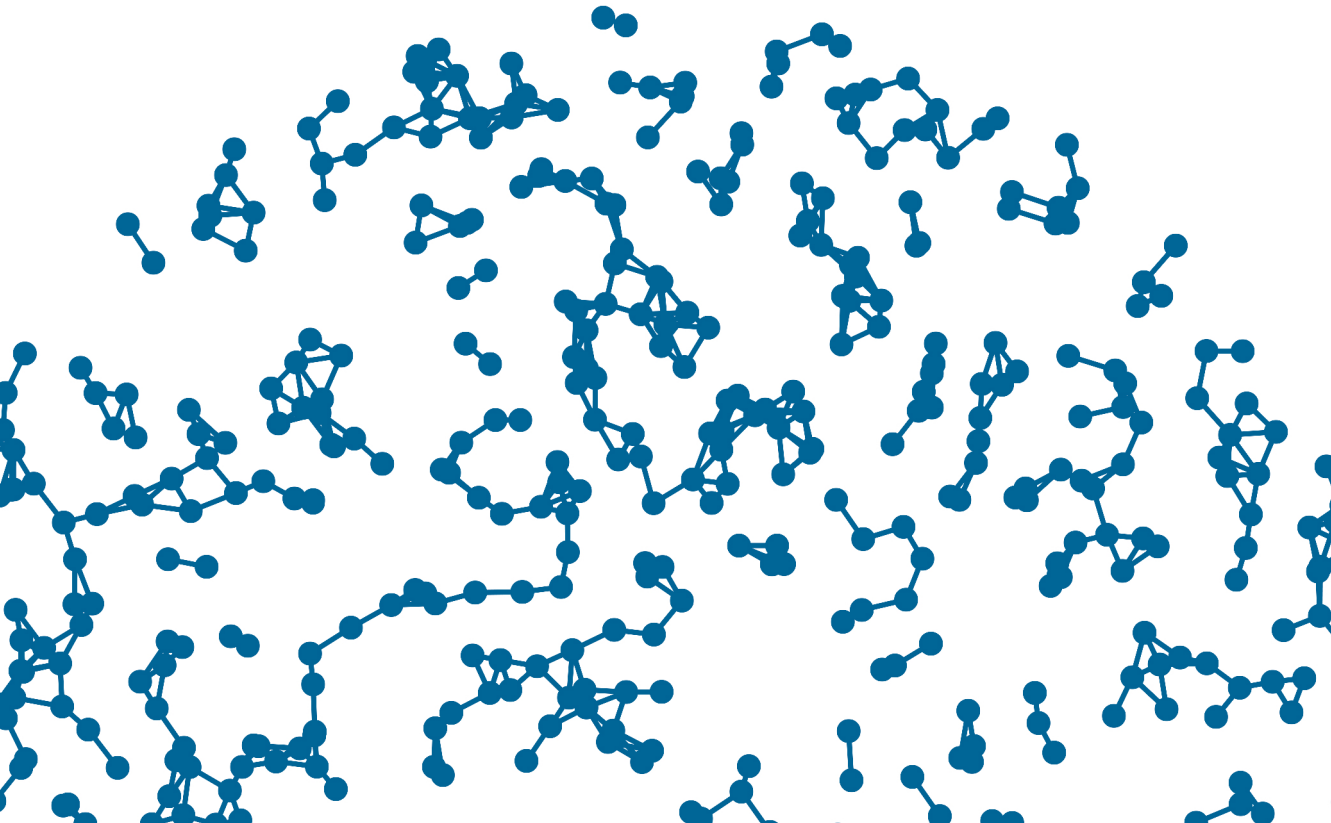
Gerrit Großmann

Stochastic Spreading on Complex Networks

Dissertation

zur Erlangung des Grades des Doktors der
Naturwissenschaften der Fakultät für
Mathematik und Informatik der
Universität des Saarlandes

Saarbrücken
2022



Tag des Kolloquiums: 12.12.2022

Dekan: Prof. Dr. Jürgen Steimle

Prüfungsausschuss: Prof. Dr. Verena Wolf
Prof. Dr. Luca Bortolussi
Prof. Dr. Tatjana Petrov
Prof. Dr. Holger Hermanns
Dr. Michael Backenköhler

An electronic version of this dissertation is available at:
`github.com/gerritgr/phd`

Abstract

Complex interacting systems are ubiquitous in nature and society. Computational modeling of these systems is, therefore, of great relevance for science and engineering. Complex networks are common representations of these systems (e.g., friendship networks or road networks). Dynamical processes (e.g., virus spreading, traffic jams) that evolve on these networks are shaped and constrained by the underlying connectivity.

This thesis provides numerical methods to study stochastic spreading processes on complex networks. We consider the processes as inherently probabilistic and analyze their behavior through the lens of probability theory. While powerful theoretical frameworks (like the SIS-epidemic model and continuous-time Markov chains) already exist, their analysis is computationally challenging. A key contribution of the thesis is to ease the computational burden of these methods.

Particularly, we provide novel methods for the efficient stochastic simulation of these processes. Based on different simulation studies, we investigate techniques for optimal vaccine distribution and critically address the usage of mathematical models during the Covid-19 pandemic. We also provide model-reduction techniques that translate complicated models into simpler ones that can be solved without resorting to simulations. Lastly, we show how to infer the underlying contact data from node-level observations.

Zusammenfassung

Komplexe, interagierende Systeme sind in Natur und Gesellschaft allgegenwärtig. Die computergestützte Modellierung dieser Systeme ist daher von immenser Bedeutung für Wissenschaft und Technik. Netzwerke sind eine gängige Art, diese Systeme zu repräsentieren (z. B. Freundschaftsnetzwerke, Straßennetze). Dynamische Prozesse (z. B. Epidemien, Staus), die sich auf diesen Netzwerken ausbreiten, werden durch die spezifische Konnektivität geformt.

In dieser Arbeit werden numerische Methoden zur Untersuchung stochastischer Ausbreitungsprozesse in komplexen Netzwerken entwickelt. Wir betrachten die Prozesse als inhärent probabilistisch und analysieren ihr Verhalten nach wahrscheinlichkeitstheoretischen Fragestellungen. Zwar gibt es bereits theoretische Grundlagen und Paradigmen (wie das SIS-Epidemiemodell und zeitkontinuierliche Markov-Ketten), aber ihre Analyse ist rechnerisch aufwändig. Ein wesentlicher Beitrag dieser Arbeit besteht darin, die Rechenlast dieser Methoden zu verringern.

Wir erforschen Methoden zur effizienten Simulation dieser Prozesse. Anhand von Simulationsstudien untersuchen wir außerdem Techniken für optimale Impfstoffverteilung und setzen uns kritisch mit der Verwendung mathematischer Modelle bei der Covid-19-Pandemie auseinander. Des Weiteren führen wir Modellreduktionen ein, mit denen komplizierte Modelle in einfachere umgewandelt werden können. Abschließend zeigen wir, wie man von Beobachtungen einzelner Knoten auf die zugrunde liegende Netzwerkstruktur schließt.

Acknowledgement

I wish to express my sincere gratitude to Verena Wolf, who has nurtured my scientific advancement since I was a Bachelor's student, and provided a wonderful and inspiring working atmosphere in her group. She always provided direction, expertise, and the freedom to discover my own path. I also wish to thank Jilles Vreeken and Luca Bortolussi for putting so many exciting ideas, topics, and challenges in my head and who greatly sharpened my scientific thinking. I am also immensely grateful to Tatjana Petrov, who said yes without hesitation when I asked her if she would be interested in reviewing my dissertation.

Special thanks go to my colleagues, collaborators, friends, and mentors, Charalampos Kyriakopoulos and Michael Backenköhler, for countless insightful discussions, passionate participation, and provision of expertise. Furthermore, I want to thank the whole *Modeling and Simulation* gang—namely, Thilo, Timo, Alexander, Joschka, Paula—and also my students Julian, Jonas, Yan Yan, and Lisa for all their valuable comments and interesting ideas.

I also want to express my very profound gratitude to my wonderful family for providing me with continuous encouragement and care throughout my studies. Last but not least, I want to thank all the people who made the last four years in Saarbrücken so vivid and joyful by filling the nearby bars, cafés, and parks with so many beautiful memories.

Contents

Part I

Preliminaries

1 Introduction

Imagine it is the beginning of 2020, you are a politician and you hear about a novel SARS-like disease emerging in China. What do you do? most likely, questions like *How bad is it going to be?*, *Should we try to prevent the virus outbreak?*, and, if so, *How could we possibly do that?* will come to mind. Fortunately, you have a compelling idea: *Can we use a computer simulation of the outbreak and see how it turns out?*

Three years into a global pandemic, and we know: We cannot. Part of this work is devoted to the challenges and caveats of epidemic forecasting. Therefore, we need to understand why a thesis on *stochastic processes on complex networks* cares about a global health crisis.

In short, epidemics are *the* quintessential example of a stochastic spreading process on complex networks. The spread of a pathogen results both from the properties of the pathogen itself (e.g., the number of days one is infectious) and from the characteristics of people's interactions. These interactions are commonly represented as a human-to-human contact network (*complex* refers to size and heterogeneity). This thesis adopts a *stochastic* modeling paradigm (instead of a *deterministic* one) to express the natural randomness and unpredictability of macroscopic systems (such as social environments) as well as the uncertainty in our knowledge.

To study stochastic spreading processes on complex networks, we need to understand their three main ingredients:

- A system of interacting agents (e.g., humans) forming the **nodes** of a network;
- A (more-or-less fixed) relationship among them (e.g., friendships, co-workers) forming the **edges**; and
- A process that propagates over the network (e.g., a pathogen) that changes the properties or **labels** of the nodes (e.g., *healthy* humans become *infected*).

Nodes, edges, and node-labels are the fundamental building blocks of our research. They allow us to write down formal specifications of a spreading process that we can study computationally. Typically, specific details of such a specification are encoded as *parameters*. A potential parameter in an epidemic model is the time it takes for an infected individual to recover. Parameters can be estimated from literature or calibrated using real-world observational data. Often, one is interested in how the emergent spreading dynamics changes as a function of the parameters.

It is also worth noting that networks naturally change over time (e.g., friendships are formed and broken). However, in this work, we consider the network structure to be static (only the labels change over time). A generalization to time-varying or adaptive networks is often (but not always) straightforward.

1.1 Networks Are Everywhere

Networks are ubiquitous. They can be found virtually everywhere in nature, society, science, and technology, as they naturally arise when entities or agents interact. This interaction is often characterized by sharing energy, mass, information, or other resources.

Networks we are all familiar with are typically networks that have a direct physical manifestation, like road networks, electrical transmission lines, the internet, or biological neural networks. Online social networks only exist in the virtual space but are still easily recognizable as networks.

On the contrary, some networks are more descriptive in nature, such as the international air traffic network, supply chain networks, the food web, or protein-protein interaction networks. Other networks are even more abstract. In the actor-network, actors are connected when they participated in the same movie, and the stock market network connects two companies when their stocks are highly correlated. Sometimes, describing something as a network might feel counter-intuitive, for instance, when considering bird flocks [Huepe et al., 2011; Young et al., 2013]. However, it can provide surprising insights, especially in these cases. For example, network analysis showed that the formation of a bird flock is a form of information processing [Moussaid et al., 2009].

1.1.1 Everything that Spreads Is an Epidemic

Networks—physical and abstract ones alike—form the substrate on which things can spread. Fake news and political polarization spread on (online and offline) social networks [Raponi et al., 2022; Campan et al., 2017], malware spreads on computer networks [Cheng et al., 2010; Hosseini and Azgomi, 2016], blackouts spread on power-grids [Koç et al., 2014], antibiotic resistance spreads among bacteria (via horizontal gene transfer) [Gehring et al., 2010], traffic jams spread on road networks [J. Lu and D. Lu, 2022], and contamination spreads in the water supply network [Davidson et al., 2005].

Other words for spreading, with sometimes slightly different connotations, are *diffusion*, *propagation*, *cascading*, and *contagion*.

One of the foundational concepts of network science was to discover that all these *spreadings* exhibit significant similarities (sometimes even referred to as a universality [Piet Van Mieghem, 2016]). In fact, for the sake of mathematical formalization, they are essentially the same. Epidemic spreading is not only the prime example but so much the default case that, in network science, anything that spreads is colloquially referred to as an epidemic. Even in mathematical notation, we often use the terms from epidemiology: An overloaded airport, a congested road, or a polarized Twitter influencer is considered to be *infected*.

1.2 There Are No Networks

With all these networks everywhere, it is essential to remember that networks are not natural entities. Like numbers, they are abstractions invented by humans to make life easier. It is not always apparent whether it is desirable to consider a system as a network. Generally speaking, a network is a neat abstraction if a particular behavior or property of a system is fundamentally characterized by the connectivity of individual components or agents—and not so much by the components individually. This property is commonly referred to as *emergent* behavior.

1.2.1 Modeler's Choices

When we want to consider a system as a network, there are still many design choices that the modeler is faced with, importantly: *What are the nodes?* and *How do we connect them?*

Brain Networks

The human brain is an excellent example of how different networks can be built from the same system. The most direct translation is to consider each neuron as a node and connect these nodes if there is an anatomical (axonal) connection between two neurons. This results in a *biological neural network* and is useful for tiny model organisms like the nematode Caenorhabditis elegans [Yan et al., 2017]. However, the size of the human brain makes it infeasible to observe and study the neural connections of the whole brain.

Alternatively, one can group all neurons of a specific brain region and consider the entire region as a single node in the brain network [Fornito, Zalesky, and Bullmore, 2016; Lynn and Bassett, 2019]. One possibility is the brain atlas proposed by Desikan et al. (2006) that identifies 68 large-scale cortical and subcortical regions of interest. Two nodes (corresponding to two regions) can then be connected if the amount of nerve fiber between them exceeds a threshold (to this end, so-called diffusion tractography techniques can be used). This results in what is called a *structural* brain network [Sporns, 2013]. Structural brain networks are useful to understand the anatomical organization of the human connectome and observe the effects of the aging process and neurodegenerative disorders such as Alzheimer's.

In addition, brain regions can be connected when they exhibit some form of statistical correlation in their activation patterns. This results in so-called *functional* brain networks [Sporns, 2013]. They are useful for investigating the brain's information processing, for instance, when studying intelligence or reactions to certain stimuli [Langer et al., 2012]. A myriad of methods and metrics exist to detect statistical associations, all resulting in (slightly) different networks.

Moreover, *effective* brain networks aim to describe causal relationships; we omit them here for brevity [Sporns, 2013]. The relationship

between different types of brain networks is also an exciting research area. However, the large number of design choices already poses a problem for replicating neuroscientific results [Buchanan et al., 2014; Welton et al., 2015; Liang et al., 2012; Fingelkurts et al., 2005].

Different types of networks are associated with their own spreading characteristics. For instance, epidemic-type spreading on the functional brain network can help to understand the emergence of epileptic seizures [Millán et al., 2022]. Spreading on the structural brain network helps to explain observed pattern in the information flow in human brains [Meier et al., 2017]. Spreading networked neurons provides potential explanations for peculiar synchronization patterns called *neuronal avalanches* [Benayoun et al., 2010].

Epidemics

Similar choices emerge in epidemiology, where a network can be constructed based on individuals and their connections, or by grouping geographical regions into nodes and connecting nodes based on the transfer between these regions. In this so-called *meta-population model*, labels do not indicate the state (infected, healthy) of an individual, but the fraction of infected individuals in a region [Murphy et al., 2021].

In summary, we note that networks are not natural objects but are subject to design choices. This construction is already part of the scientific process. Besides defining nodes and edges, there are many degrees of freedom (e.g., whether the directing or strength of the interaction has to be considered). However, in most cases, this thesis assumes that the network is already provided and focuses on their analysis.

We provide a brief overview of the historical influences of this work in Excursus 1.

Excursus 1: On the Shoulders of Giants

The study of spreading dynamics and the networks on which they unfold is a highly multidisciplinary research area. Here, we want to lay out some of the larger influences. A more detailed review can be found in [Barabási, 2013; Estrada and Knight, 2015; Fennell, 2015].

Epidemiology. Mathematical epidemiology originated in the seminal work of Kermack and McKendrick (1927) who were the first to describe the evolution of an epidemic with a mathematical equation system: the Susceptible-Infected-Recovered (SIR) model. Later, stochastic models were adopted, for instance by Bartlett (1960) and Bailey (1964) (cf. Held et al. (2019)). Similar models were developed in other branches of theoretical biology to study interacting (groups of) agents. For example, the Lotka-Volterra equations describe the population dynamics of two interacting species (predator and prey) [Lotka, 1925].

Statistical Physics. Statistical physics studies the interactions of atoms and molecules through a statistical lens. Ludwig Boltzmann championed it in the 19th century [Boltzmann, 2012]. It describes a system (e.g., a gas) using a joint probability over the properties of the particles such that it favors low-energy states. Mean-field equations were developed to make the equations amenable to mathematical analysis by averaging the effects of neighboring molecules. The field put much effort into studying phase transitions (e.g., matter changes from the fluid state to a gas). The underlying framework was adopted in computational epidemiology (e.g., a pathogen becoming endemic can be seen as a phase transition) [Castellano et al., 2009].

Graph Theory. The roots of graph theory can be traced back to Leonhard Euler, who, in 1736, analyzed a peculiar infrastructure network: the bridges of Königsberg. He showed that it was impossible to walk across all seven bridges and never cross the same one twice. Thereby, he invented what we now call a *graph* by leaving out unnecessary details (cf. [Estrada and Knight, 2015]). Seminal work in modern graph theory emerged during the study of random graphs by Erdős, Rényi, et al. (1960). Many concepts of graph theory became relevant to network science. For instance, spectral graph theory studies the eigenvalues of (a matrix representation of) a graph and can be used to predict if a pathogen will become endemic or not [Prakash, Chakrabarti, et al., 2012]. Likewise, it carries information about the communities in a graph or network (i.e., densely connected substructures that are only loosely connected to other communities) [Piet Van Mieghem, 2010].

Stochastic Processes. Notable influences are drawn from the science of stochastic processes, specifically the study of Markov chains. Seminal work was done in the area of queuing theory by Jackson (1957) where customers in a network of waiting lines are modeled using random service times. Of similar importance was the interpretation of chemical reactions as a stochastic system by Gillespie (1977); related concepts were developed even earlier by Doob (1945). Kingman (1969) proposed a Markov chain semantics of population dynamics. A technique that became extremely useful for many application areas (see Goutsias and Jenkinson (2013) for an overview). Probability theory and stochastic process provide the underlying mathematical theory and make the description of such systems and the formalization of their properties rigorous.

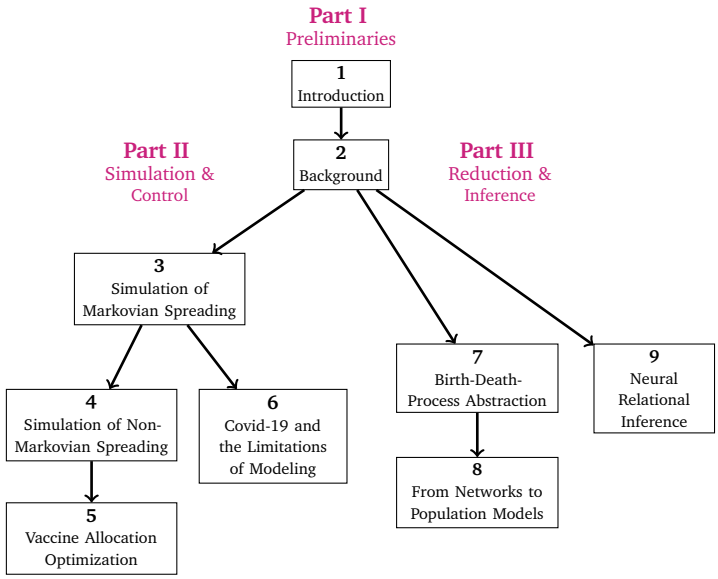


Fig. 1.1.: Thesis structure and recommended reading order.

1.3 About the Thesis

The thesis consists of various contributions that have been previously published as scientific articles or poster abstracts. Altogether, there are nine peer-reviewed publications (and one publication currently under review), grouped into seven chapters (3 to 9). The chapters are loosely grouped into two parts: **Part II: Simulation and Control** deals with the simulation of spreading processes and results from simulation studies, **Part III: Reduction and Inference** studies simulation-free model reduction techniques and methods for analyzing observational data. The remaining parts (**Part I: Preliminaries** and **Part IV: Concluding Remarks**) encapsulate the thesis for the sake of clarity and self-sufficiency. While each chapter identifies a unique idea, all are based on the notation and formalism introduced in Chapter 2 (*Background*). Figure 1.1 provides a suggested reading order.

Naturally, the thesis largely overlaps with formal publications. Some details are left out or added, and some parts are restructured to avoid redundancy and create consistency. Thus, a considerable amount of the thesis consists of self-citations (which are not explicitly mentioned each time). This is standard practice and is in compliance with current copyright law and university regulations.

A comprehensive list of the publications and the corresponding contributions is given below. In all cases, all authors provided critical feedback, reviewed the final manuscript, and provided at least minor edits. All publications were supervised by Verena Wolf (V.W.). These contributions are not repeated below for each manuscript individually.

Chapter 3
is based on the paper

- Rejection-Based Simulation of Stochastic Spreading Processes on Complex Networks [Großmann and Wolf, 2019];

published in the 2019 *Hybrid Systems and Biology* proceedings.

Contribution. Gerrit Großmann (G.G.) conceived the main conceptual ideas, carried out the implementation, performed the numerical experiments, and wrote the main part of the manuscript. V.W. contributed to the design and presentation of the case studies.

Chapter 4
is based on the two papers

- Rejection-Based Simulation of Non-Markovian Agents on Complex Networks [Großmann, Bortolussi, and Wolf, 2019];

- Efficient simulation of non-Markovian dynamics on complex networks [Großmann, Bortolussi, and Wolf, 2020];

published in the 2019 *International Conference on Complex Networks and Their Applications* conference proceedings and in PLOS ONE, respectively. The latter is a follow-up paper and extends the ideas of the former.

Contribution. G.G. conceived the main conceptual ideas, carried out the implementation, performed the numerical experiments, and wrote the main part of the manuscript. V.W. contributed to the correctness proofs. Luca Bortolussi (L.B.) wrote large parts of the abstract.

Chapter 5
is based on the paper

- Learning Vaccine Allocation from Simulations [Großmann, Backenköhler, Klesen, et al., 2020];

published in the 2020 *International Conference on Complex Networks and Their Applications* conference proceedings.

Contribution. G.G. conceived the main conceptual ideas, carried out the implementation, performed the main part of the numerical experiments, and wrote the main part of the manuscript. Michael Backenköhler (M.B.) provided insights into ranking methods and contributed to the presentation of the results. Jonas Klesen contributed to the review of the literature and the numerical evaluation and provided an independent examination of the results. V.W. contributed to the presentation of the results.

Chapter 6
is based on two papers

- Importance of interaction structure and stochasticity for epidemic spreading: A COVID-19 case study [Großmann, Backenköhler, and Wolf, 2020];

- Heterogeneity matters: Contact structure and individual variation shape epidemic dynamics [Großmann, Backenköhler, and Wolf, 2021b];

published in the 2020 *International Conference on Quantitative Evaluation of Systems* conference proceedings and in PLOS ONE, respectively.

Contribution. G.G. conceived the main conceptual ideas, carried out the implementation, performed the numerical experiments, and wrote the main part of the manuscript. All authors contributed to the concrete realization of the method and to the decision of what experiments to carry out. V.W. contributed to the literature review and to theoretical insights of branching processes.

Chapter 7
introduces a model-reduction technique using a very simple abstraction scheme. Parts of this chapter are published as

- Birth-Death Processes Reproduce the Infection Footprint of Complex Networks [Backenköhler and Großmann, 2020];

- Birth-Death Processes Reproduce the Infection Footprint [Großmann and Backenköhler, 2022].

The former contribution was submitted as a poster abstract to the 2020 *Hybrid Systems and Biology* conference. It was peer-reviewed and accepted but was never formally published. The latter contribution is published as an extended abstract in the 2022 *International Conference on Complex Networks and Their Applications* conference proceedings.

Contribution. G.G. wrote the larger part of the manuscript and provided the implementation and numerical evaluation. The main

method resulted from discussions between M.B. and G.G. Additionally, M.B. also wrote significant parts of the manuscript, contributed to the literature review and provided theoretical insights into graph relaxation. He also helped to improve the figures of the manuscript.

Chapter 8
explains how to analyze spreading models by

- Reducing Spreading Processes on Networks to Markov Population Models [Großmann and Bortolussi, 2019];

published in the 2018 *International Conference on Quantitative Evaluation of Systems* proceedings.

Contribution. G.G. conceived the main conceptual ideas, carried out the implementation, performed the numerical experiments, and wrote the main part of the manuscript. L.B. corrected technical details and contributed a paragraph explaining specific formulas.

Chapter 9
presents the inference part of the thesis:

- Unsupervised Relational Inference Using Masked Reconstruction [Großmann, Zimmerlin, et al., 2021];

where we assume that the network structure is not given, but we reconstruct it from observational data of a stochastic process. The work is currently under review at *Applied Network Science* (Springer).

Contribution. G.G. wrote the main manuscript text, devised the main idea, provided the implementation, and conducted the experiments. Julian Zimmerlin edited the manuscript text, implemented a previous version of a related model, supported the conceptualization, and contributed to the literature review. M.B. wrote parts of the manuscript, reviewed the literature, and created figures. V.W. wrote parts of the manuscript.

2 Background

This chapter introduces the notation we use in the course of the thesis and provides the necessary background knowledge. In particular, we define stochastic spreading processes, illustrate how they can be simulated, and explain their relationship to so-called *continuous-time Markov chains* (CTMCs). For a comprehensive overview of network science, we refer the reader to M. Newman (2018) and Barabási (2013); for a deeper dive into the mathematics and proofs, we refer the reader to István Z Kiss et al. (2017).

2.1 General Notation

We use $\mathbb{N}$ and $\mathbb{R}$ to denote the set of natural and real numbers, respectively. Functions are easily recognizable by a placeholder dot, e.g., $f(\cdot)$, $g(\cdot)$, $L(\cdot)$. Scalars are written in the default math-font, e.g., $x, y \in \mathbb{R}$. Vectors are lowercased and bold-faced, e.g., $\mathbf{x} \in \mathbb{R}^2$. Matrices and tensors are in uppercase ($\mathbf{A}$, $\mathbf{B}$, ...). To avoid confusion, we typically do not use subscripts as indices, instead we use $\mathbf{a}[i]$ or $\mathbf{A}[i, j]$ to refer to specific elements. Subscripts are, however, often used to specify time-steps.

Specific node-states are in typewriter font (`A`, `B`). The set of node-states is denoted $\mathcal{S}$, often $\mathcal{S} = \{\mathtt{S}, \mathtt{I}\}$ for infected and susceptible nodes. Finite sets and tuples are often written in calligraphy, such as the set of nodes $\mathcal{V}$ or the set of edges $\mathcal{E}$ of a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$

We use $\mathrm{Cat}(n)$ to denote the set of all probability vectors of size n:

$$\mathrm{Cat}(n) = \left\{ \mathbf{x} \in \mathbb{R}^n_{\geqslant 0} \mid \sum_i \mathbf{x}[i] = 1.0 \right\}.$$

Likewise, for a countable set $\mathcal{X}$, we use $\mathrm{Cat}(\mathcal{X})$ to denote the set of all probability distributions over $\mathcal{X}$ (typically assuming an implicit enumeration).

2.2 Graphs and Networks

The fundamental objects of network science are graphs. A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ consists of a finite[1] set of nodes (or vertices) $\mathcal{V}$ and a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. Two nodes $v_i, v_j \in \mathcal{V}$ are called *neighbors* (or adjacent/connected) if and only if there is an edge connecting them, that is, $(v_i, v_j) \in \mathcal{E}$. We use $n_{\mathcal{V}} = n = |\mathcal{V}|$ to denote the number of nodes. We use $n_{\mathcal{V}}$ only in special cases to avoid confusion. Moreover, we typically assume that:

- Graphs are undirected: when $(v_i, v_j) \in \mathcal{E}$, we (sometimes implicitly) assume that $(v_j, v_i) \in \mathcal{E}$;
- Graphs contain no self-loops: for all $v_i \in \mathcal{V}$, $(v_i, v_i) \notin \mathcal{E}$;
- Graphs are strongly connected: All nodes are reachable from all other nodes.

Degree and Degree Distribution. The *degree* of a node is the number of its neighbors. We use $k_i = |\{v_j \mid (v_i, v_j) \in \mathcal{E}\}|$ to denote the degree of v_i. The *degree distribution*, $P(\cdot)$, represents the frequency of each degree in a given graph. That is, $P(K = k)$ is the probability that a node (that is chosen uniformly at random) has degree k.

[1] The set of nodes and edges can be countably infinite, but this thesis only deals with finite graphs.

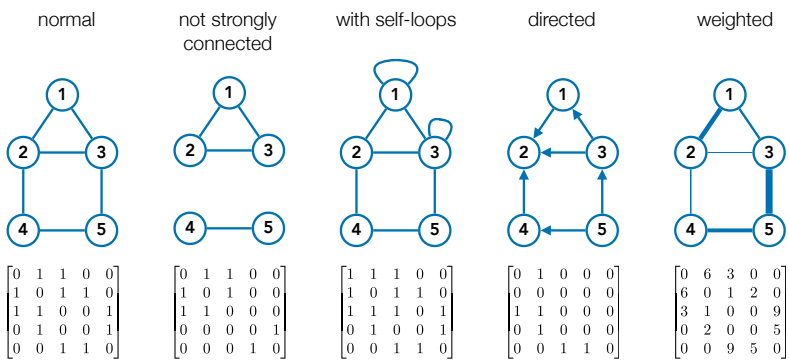


Fig. 2.1.: Different graph types with their corresponding adjacency matrix.

Adjacency matrix. Assume an enumeration of the (finite number of) nodes of a graph $v_1, v_2, \ldots, v_n$. We can represent $\mathcal{G}$ as a so-called *adjacency matrix* $\mathbf{A}$ of dimension $n_\mathcal{V} \times n_\mathcal{V}$. The entries in $\mathbf{A}$ are defined as:

$$\mathbf{A}[i,j] = \begin{cases} 1 & (v_i, v_j) \in \mathcal{E} \\ 0 & (v_i, v_j) \notin \mathcal{E} \end{cases}.$$

In an undirected graph, $\mathbf{A}$ will be symmetric and, when self-loops are absent, all diagonal entries will be zero. The degree of v_i, is the sum of the i-th column (or equivalent row) in $\mathbf{A}$. A graph is connected if $\mathbf{A}^n$ only has non-zero entries (a non-zero entry in $\mathbf{A}^k$ indicates that there is a path of length $\leqslant k$ connecting the two nodes). Adjacency metrics also make it easy to encode *weighted* graphs, where an entry a_{ij} encodes the strength of an interaction.

Example 2.1: Graphs and Matrices

Graph visualizations are shown in Figure 2.1 of a graph with $\mathcal{V} = \{1, \ldots, 6\}$ and different edge sets. We typically assume that the graph is of the leftmost type (normal).

2.2.1 Random Graph Models

Random graph models provide recipes for generating graphs. Typically, these mimic or highlight specific properties of some real-world networks. Famous examples are the *Erdős–Rényi* model, where each pair of nodes is connected by an edge with some probability p. This model is characterized by its simplicity and can serve as a baseline. In the *geometric* random graph model, nodes are randomly placed in a Euclidean space. Two nodes are connected if their distance is below a certain threshold r (called radius). Geometric graphs exhibit some properties similar to those of real-world contact networks. In the *Barabási–Albert* model, a graph is created node-by-node. Each new node is connected to m of the already existing nodes. Importantly, the m nodes are not chosen uniformly. The probability of choosing v_i is proportional to its current degree k_i. This mechanism is called *preferential attachment*. It results in a degree distribution that follows a power-law [Barabási, 2013].

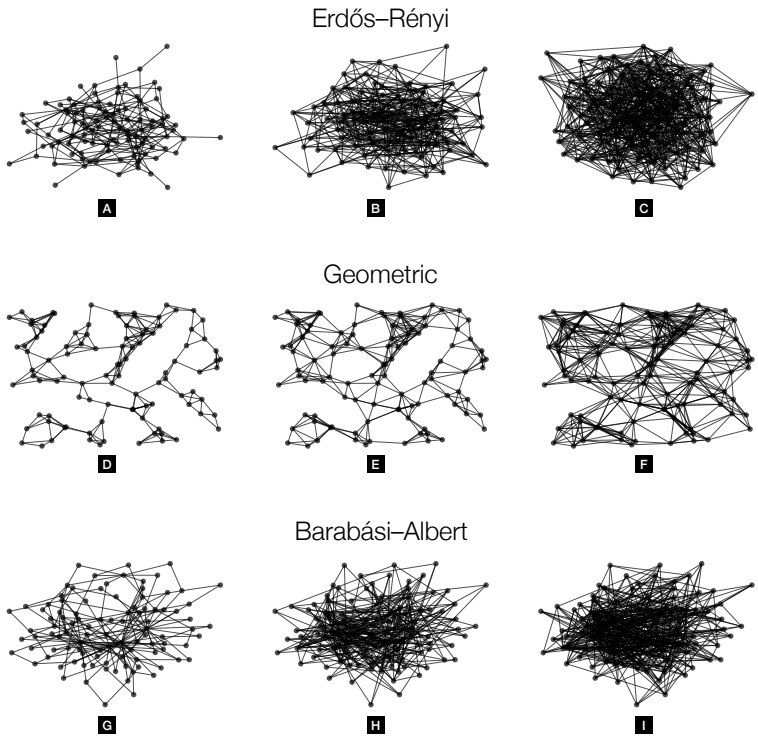


Fig. 2.2.: Samples of three different random graph models with varying parameters.

Example 2.2: Random Graph Models

We generate networks with 100 nodes. For the Erdős–Rényi graph, we use a connection probability of (A): 0.05 ($|\mathcal{E}| = 224$); (B): 0.1 ($|\mathcal{E}| = 474$), and (C): 0.15 ($|\mathcal{E}| = 730$). Geometric graphs are constructed by sampling 100 points (nodes) in $[0, 1]^2$ and using a radius of (D): 0.14 ($|\mathcal{E}| = 248$), (E): 0.17 ($|\mathcal{E}| = 338$), and (F): 0.23 ($|\mathcal{E}| = 627$). To generate Barabási–Albert graphs, we use a preferentially attachment parameter m of (G): 2 ($|\mathcal{E}| = 196$), (H): 3 ($|\mathcal{E}| = 291$), and (J): 4 ($|\mathcal{E}| = 475$).

Visualizations are provided in Figure 2.2.

2.3 Labeled Networks

We are mostly interested in a special form of graphs called *labeled graphs*, where each node is associated with a label (e.g., infected or susceptible). The label stems from a (typically finite) label set denoted $\mathcal{S}$ (e.g., $\mathcal{S} = \{\mathtt{S}, \mathtt{I}\}$ for infected and susceptible). A labeling-function $L : \mathcal{V} \to \mathcal{S}$ maps each node to its corresponding label. We say that $L(v_i)$ is the *node-state* of v_i. In the context of epidemic modeling, the node-state is also called *compartment*. We also call it the *internal* or *local* state of a node.

Given a node-enumeration, we can represent $L(\cdot)$ as a labeling-vector $\mathbf{x} \in \mathcal{S}^n$, where the entry corresponding to the i-th node is defined as $\mathbf{x}[i] = L(v_i)$. We call $\mathbf{x}$ a *network-state*, as it describes the superposition of all node-states.

2.3.1 Trajectories

When we talk about processes unfolding on networks, we usually mean that the graph is fixed and the node-states change over time and influence each other. That is, for a static graph G with n nodes, we look at a sequence of time-annotated network-states. Such a sequence is also called *trajectory* and is denoted τ. We start at time $t = 0$ and assume a real-valued time-horizon, h, is specified upfront.

Continuous-Time Trajectory. In this thesis, we typically assume that time is continuous, allowing us to keep track of time in a refined way and providing flexibility. The process can stay in a specific network-state for an arbitrary amount of time called *residence time*. Thus, we annotate a network-state with the timepoint when it is entered. Let $\mathbf{x}_j$ denote the j-th network-state. Then, $(\mathbf{x}_j, t_j)$ means that at time

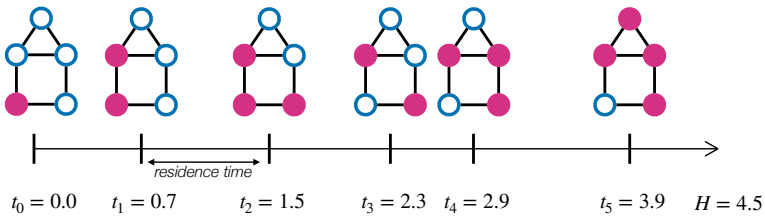


Fig. 2.3.: Example of continuous-time trajectory. The two node-states are indicated by color and filling. Network-states are illustrated together with the contact network.

t_j the system jumps from $\mathbf{x}_{j-1}$ to $\mathbf{x}_j$. The residence time in $\mathbf{x}_j$ is $t_{j+1} - t_j$.

Intuitively, we start with $t_0 = 0$ and increase t until the predefined time-horizon h is reached. The network-state changes at timepoints $t_1 < t_2 < \ldots t_m < h < t_{m+1}$ (with $t_i \in \mathbb{R}_{>0}$). We define a *trajectory* as a sequence

$$\tau = [(0, \mathbf{x}_0),\ (t_1, \mathbf{x}_1),\ \ldots,\ (t_m, \mathbf{x}_m)]\ .$$

Example 2.3: Trajectory

An example continuous-time trajectory with two node-states is provided in Figure 2.3. The trajectory reaches six different network-states until a time horizon ($h = 4.5$) is reached.

In an abstract perspective, we can assume that a *process* is a black box that generates a trajectory when given a graph and an initial network-state. A *stochastic* process uses randomness to generate a variety of different trajectories from the same initial state. In the next section, we will explicitly explain how to generate such trajectories for typical spreading models. Thereby, we define a family of useful stochastic processes on networks.

2.4 Stochastic Dynamics on Networks

The quintessential example of spreading on networks is the *Susceptible-Infected-Susceptible* (SIS) model. Next, we introduce the SIS model as a concrete example of a stochastic spreading process. Then we generalize the possible dynamical laws to a more expressive class called *multi-state* models.

2.4.1 SIS Model

In the SIS model, nodes are either infected (I) or susceptible (S) (healthy but not immune). Thus, the set of node labels is $\mathbb{S} = \{\mathtt{I}, \mathtt{S}\}$. Two laws govern the stochastic behavior of the system:

- **Recovery:** Each I-node can spontaneously recover (i.e., become susceptible again) with recovery rate $\alpha \in \mathbb{R}_{>0}$.
- **Infection:** Each SI-edge transmits an infecting (turns into an II-edge) with the infection rate $\beta \in \mathbb{R}_{>0}$.

The model is parametrized by two variables α and β called (reaction) rates (or *rate constants* to be more precise). Colloquially, they specify the speed of the reactions. If β is large compared to α then infections happen faster (i.e., more often) and most nodes tend to be infected. When recoveries happen faster, most nodes are susceptible.

The first rule specifies that each I-node turns into an S-node at a certain speed. We call this a *spontaneous* (or *independent*) rule because the behavior does not depend on interactions with neighboring nodes. Conversely, the second rule depends on an interaction between two nodes and is called a *contact* or *interaction* rule.

Model Assumptions. The SIS model is extremely simple. Notable model simplifications are that nodes are either infected or susceptible, and do not receive immunity after an infection. They can also not be vaccinated or die. Moreover, all connections have the same strength, and the time a node is infected/susceptible follows an exponential distribution (more on this in the next section). However—even with all these simplifications—solving the model mathematically is infeasible for realistic networks (e.g., more than 20 nodes). As we shall see, this simple model is already quite powerful and is now arguably the de facto standard to model information diffusion on networks.

Generating SIS Trajectories. We generate trajectories in a sequential way. That is, for a given trajectory:

$$\tau = [(0, \mathbf{x}_0),\ (t_1, \mathbf{x}_1),\ \ldots,\ (t_j, \mathbf{x}_j)]$$

we generate the next entry $(t_{j+1}, \mathbf{x}_{j+1})$. Note that we assume that the initial (first) network-state at $t = 0$ is given. Furthermore, we assume that both the residence time in $\mathbf{x}_j$ (that is, $\mathbf{t}_{j+1} - \mathbf{t}_j$) and the successor state $\mathbf{x}_{j+1}$ depend only on the current network-state $\mathbf{x}_j$, not on previous network states and not on the absolute time that has passed already. This is called a *Markov property* and we investigate it in more detail in the simulation chapters.

2.4.2 Naïve SIS Simulation

Given a network-state and time $(t_j, \mathbf{x}_j)$, we generate the next network-state with steps described in Algorithm 1.

Algorithm 1: Naïve SIS Simulation

Input: Rates α, β; Graph $\mathcal{G}$; Network-state $\mathbf{x}_j$; Time t_j.
Output: Successor network-state $\mathbf{x}_{j+1}$; t_{j+1}.

1. Initialize an empty list L and $\mathbf{x}_{j+1} \coloneqq \mathbf{x}_j$.
2. For each I-node v_i: Generate an exponentially distributed random variate with rate α, i.e., $t \sim \text{Exp}(\alpha)$, representing the node's recovery time (candidate).
   $\Rightarrow$ Store $(t, v_i, \mathtt{S})$ in L.
3. For each SI-edge (v_i, v_j): Generate an exponentially distributed random variate with rate β, i.e., $t \sim \text{Exp}(\beta)$, representing the timepoint (candidate) when v_j infects v_i (each edge is considered only once).
   $\Rightarrow$ Store $(t, v_i, \mathtt{I})$ in L.
4. Identify the triplet $(\hat{t}, \hat{v_i}, \hat{s})$ with the shortest jump time candidate t from L and:
   $\Rightarrow$ Set $\mathbf{x}_{j+1}[\hat{i}] \coloneqq \hat{s}$ and $t_{j+1} \coloneqq t_j + \hat{t}$.

The method is *naïve* in the sense that it is statistically correct but computationally expensive. Other methods, discussed in Chapter 3 (*Simulation of Markovian Spreading*), minimize the number of generated random variates and do not clear the event list L in each step.

Exponential Distribution. We use $\mathrm{Exp}(\lambda)$ to denote the exponential probability distribution with rate $\lambda \in \mathbb{R}_{>0}$ (not to be confused with the exponential function $e^x = \exp(x)$). To sample a random variate $y \sim \mathrm{Exp}(\lambda)$. One can use $y = -\ln(u)/\lambda$, where u is a random sample from the uniform distribution over $[0, 1]$ and $\ln(\cdot)$ is the natural logarithm. The probability density of $\mathrm{Exp}(\lambda)$ is:

$$f_\lambda(x) = \begin{cases} \lambda e^{-\lambda x} & x \geqslant 0 \\ 0 & \text{otherwise.} \end{cases}$$

Race Condition. We generate jump-time candidates for each possible reaction (infection or recovery) and only care about the smallest one. All other variates are discarded. This is called a *race condition*.

2.4.3 Family of Multi-State Models

Following Fennell and Gleeson (2019), we now introduce a general class of spreading models called *multi-state* processes. A similar model class is studied under the name of *Stochastic Automata Networks* (SANs) [Plateau and W. J. Stewart, 2000].

Consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, a finite set of node-states $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$, and a network-state $\mathbf{x}$. For a given node $v_i \in \mathcal{V}$, we use $N(i, s_j)$ to denote the number of neighbors in state $s_j \in \mathcal{S}$. Moreover, we define the *neighborhood-counting vector*

$$\mathbf{m}_i \in \mathbb{Z}_{\geqslant 0}^{|\mathcal{S}|} = [N(i, s_1), N(i, s_2), \ldots, N(i, s_{|\mathcal{S}|})] \,.$$

The sum over the neighborhood-counting vector is always the node's degree: $\sum_j \mathbf{m}_i[j] = k_i$. In an abuse of notation, for an arbitrary node-state s_j, we often use s_j directly as an index, that is, we write $\mathbf{m}[s_j]$ instead of $\mathbf{m}[j]$.

Example 2.4: Neighborhood Vectors

Assume that $\mathcal{S} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}\}$ and that v_1 has three neighbors, one in node-state A and two in C. Then, $\mathbf{m}_1 = (1, 0, 2)$ (assuming alphabetical ordering). We might write $\mathbf{m}_1[\mathtt{C}]$, not $\mathbf{m}_1[3]$, to denote the number of neighbors of v_1 in node-state C.

Rules. In the multi-state paradigm a *reaction rule* is a triplet $s_{in} \xrightarrow{f(\cdot)} s_{out}$ (or $(s_{in}, f(\cdot), s_{out})$) where:

- $s_{in} \in \mathcal{S}$ is the initial node-state of a node (before the rule is applied).
- $f : \mathbb{Z}_{\geqslant 0}^{|\mathcal{S}|} \to \mathbb{R}_{\geqslant 0}$ is a rate function operating on neighborhood-counting vectors.
- $s_{out} \in \mathcal{S}$ is the output node-state of a node (after the rule is applied).

For each node v_i in state s_{in}, the rate $f(\mathbf{m}_i)$ denotes how quickly this node turns into s_{out}. Typically, we only allow rules with $s_{in} \neq s_{out}$.

Example 2.5: Rules

Assume that $\mathcal{S} = \{\mathtt{A}, \mathtt{B}\}$. A rule

$$\mathtt{A} \xrightarrow{f(\mathbf{m})=\mathbf{m}[\mathtt{B}]} \mathtt{B}$$

turns A-nodes into B-nodes. The higher the number of B-neighbors, the faster an A-node will flip. If a node has zero B-neighbors, its rate will be zero, thus, the node will stay in A (until at least one of its neighbors changes to B).
Notationally, we typically omit the left side of the rate-equation and write $\mathtt{A} \xrightarrow{\mathbf{m}[\mathtt{B}]} \mathtt{B}$.

Naïve Multi-State Simulation. Again, we define the stochastic semantics of a multi-state model in a generative manner. A multi-state model consists of an ordered set of node-states, an ordered set of rules, and the underlying graph. Starting with an initial network-state, we generate trajectories step-by-step. Assume the current network-state is $\mathbf{x}_j$. The successor network-state ($\mathbf{x}_{j+1}$) and the jump-time of the current network-state (t_{j+1}) are generated following Algorithm 2.

Algorithm 2: Naïve Multi-State Simulation

Input: Rules; Graph $\mathcal{G}$; Network-state $\mathbf{x}_j$; Time t_j.
Output: Successor network-state $\mathbf{x}_{j+1}$; t_{j+1}.

1. Initialize an empty list L and $\mathbf{x}_{j+1} \coloneqq \mathbf{x}_j$.
2. For each node v_i and each applicable rule $s_{in} \xrightarrow{f(\cdot)} s_{out}$ (applicable means $s_{in} = \mathbf{x}_j[i]$):
   $\Rightarrow$ Sample a candidate jump-time $t \sim \text{Exp}(f(\mathbf{m}_i))$.
   $\Rightarrow$ Store (t, v_i, s_{out}) in L.
3. Among all triples in L, identify $(\hat{t}, \hat{v}_i, \hat{s}_{out})$ with the smallest time and:
   $\Rightarrow$ Set $\mathbf{x}_{j+1}[\hat{i}] \coloneqq \hat{s}_{out}$ and $t_{j+1} \coloneqq t_j + \hat{t}$.

We can generate successor states until the time horizon is reached. Similar to the `SIS` simulation, all pairs of nodes and rules compete in a race condition.

Model 1: Susceptible-Infected-Susceptible (SIS)

In the SIS model, infected (I) nodes infect their susceptible (S) neighbors. Infected nodes can also recover, that is, they become susceptible again (no immunity is gained).

States: $\mathcal{S} = \{\mathtt{S}, \mathtt{I}\}$

Parameters: Recovery rate: $\alpha \in \mathbb{R}_{>0}$
Infection rate: $\beta \in \mathbb{R}_{>0}$

Rules: Recovery: $\mathtt{I} \xrightarrow{\alpha} \mathtt{S}$
Infection: $\mathtt{S} \xrightarrow{\beta \mathbf{m}[\mathtt{S}]} \mathtt{I}$

Examples: Figure 2.3 (p. 23) shows a trajectory of an SIS-like process (infected nodes are shown to be filled and susceptible nodes are empty). Figure 2.5 (p. 34) shows the *Markov graph* (all network-states and their relation) on a toy model.

From SIS to Multi-State. The SIS model can be defined as a multi-state process as seen in Model 1. The equivalence to the rules in Section 2.4.1 is not completely obvious, because in the SIS simulation (cf. Algorithm 1), we compute an individual jump-time candidate for each SI-edge (with rate β). Now we compute for each S-node v_i a single infection time with rate $N(i, \mathtt{I})\beta$. The reason for the equivalence lies in the properties of the exponential distribution. Specifically, the minimum of $N(i, \mathtt{I})$ random variates with rate β follows the same distribution as a single random variate with rate $N(i, \mathtt{I})\beta$ [A. O. Allen, 1990]. This property is important and we will exploit it in the following chapters.

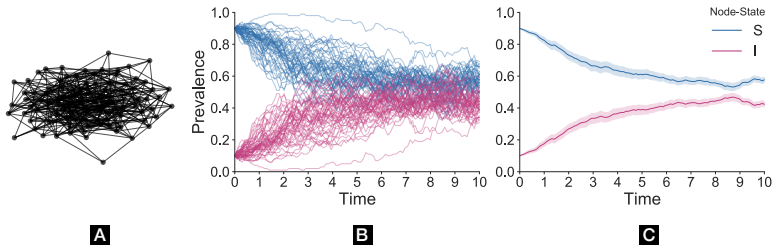


Fig. 2.4.: Summary statistics on an ER-graph: **(a)**: ER-graph. **(b)**: all individual trajectories. **(c)**: the aggregated mean (with bootstrapped 95% confidence intervals).

Global Summary Statistics. Often, we are only interested in the high-level behavior of a system, for instance, the overall number of infected nodes over time. Therefore, for a given trajectory τ, we define $S_\tau(t, s)$ as the fraction of nodes in node-state $s \in \mathcal{S}$ at timepoint t; $S_\tau(t, s)$ is often referred to as *prevalence* (of state s at t). Note that, timepoint t corresponds to the network-state $\mathbf{x}_j$ such that $t_{j+1} > t > t_j$. For a set of trajectories, we can consider the empirical mean of S (or other useful statistical quantities).

Example 2.6: Global Summary Statistics

We simulate `SIS` dynamics on an ER graph with 100 nodes and generate 100 trajectories $\{\tau\}_i$. Starting with 10 infected nodes, one can clearly see how the infection spreads, so that eventually half of the nodes are infected. The function $S_\tau(t, s)$ is visualized in Figure 2.4.

2.4.4 Properties of Trajectories

Now we know how to generate different trajectories. Here, we want to recap some of their properties. The description is colloquial and will be elucidated in the course of the thesis.

1. A trajectory always starts in a fixed initial state; after that, the behavior is governed by randomness.
2. Jump times happen consecutively in time. Only a finite number of jumps are possible in a finite amount of time.
3. Exactly one node changes its node-state at each step.
4. At any given timepoint, the future behavior only depends on the current network-state (following from the Markov-property in time and time-homogeneity).
5. The behavior of a node only depends on the node itself and its immediate neighborhood (Markov-property in space).
6. All nodes are equivalent and can, when considered in isolation, only be distinguished by their node-state (network homogeneity).
7. The jump-time of each network-state follows an exponential distribution.
8. The probability of a future network-state does not depend on the jump-time of the current state.

The properties (1) to (6) follow directly from the generation method (cf. Algorithm 2) and from our definition of rules. We further discuss these properties in Section 2.5 in the context of Markov chains. Property (7) follows from the fact that the minimum of n independent exponentially distributed random variables also follows an exponential distribution. Property (8) follows from the fact that the probability that a specific node "wins" the race condition is constant over time. In the course of the thesis, we relax some of these properties. For instance, in Chapter 4 (*Simulation of non-Markovian Spreading*), we consider cases where the properties (7) and (8) are not satisfied. In Chapter 9 (*Neural Relational Inference*), we also study trajectories of deterministic systems.

2.5 CTMC Semantics of Spreading Dynamics

Continuous-time Markov chains (CTMCs) are a well-known formalism for describing stochastic processes [W. J. Anderson, 2012]. We can specify the semantics of a multi-state process by translating it to a CTMC. This is a viable alternative to specifying the semantics in terms of a generative simulation algorithm, as seen in the previous sections. The equivalence between CTMCs and multi-state processes was first noticed by Piet Van Mieghem et al. (2008).

A CTMC is typically specified by state space $\mathcal{X}$ and state transitions $\mathcal{T}$ (and an initial CTMC-state). State transitions are directed edges between elements of the state space. Each transition is annotated with a rate. That is $\mathcal{T} \subseteq \mathcal{X} \times \mathbb{R}_{>0} \times \mathcal{X}$. We use $\lambda_{x,y} \in \mathbb{R}_{>0}$ to denote the corresponding transition-rate between two CTMC-states.

CTMC Construction. We transform a multi-state process into a (time-homogeneous) CTMC in the following way:

- The **state space** is the set of network-states. That is $\mathcal{X} = \mathcal{S}^n$ (n is the number of nodes, $\mathcal{S}$ is the set of node-states).
- A **transition** $(\mathbf{x}, \lambda_{\mathbf{x},\mathbf{y}}, \mathbf{y}) \in \mathcal{T}$ exists iff at least one[2] rule $r = s_{in} \xrightarrow{f(\cdot)} s_{out}$ exists such that applying r to any node in $\mathbf{x}$ leads to $\mathbf{y}$. Let v_z denote this node, then $\lambda_{\mathbf{x},\mathbf{y}} = f(\mathbf{m}_z)$.

We call this graph-based CTMC-specification *Markov graph*.

[2] In the somewhat pathological case that two (or more) rules have the same s_{in} and s_{out}, we need to sum up the corresponding rates to obtain $\lambda_{\mathbf{x},\mathbf{y}}$.

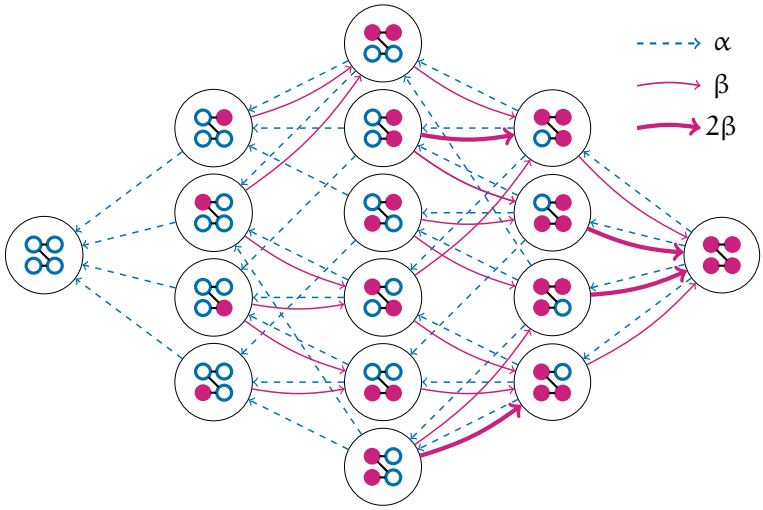


Fig. 2.5.: Markov graph induced by the SIS model (S : *blue*, I : *pink, filled*) for a 4-node contact network.

Example 2.7: Markov Graph

The complete Markov graph of a simple contact network and SIS dynamics (Model 1) is given in Figure 2.5.

Building a CTMC is useful in many ways:

- It rigorously specifies a probability density over the set of trajectories (formally, we can deduce a probability space $(\Omega, \mathcal{F}, \mathrm{P})$ over the trajectories [Vardi, 1985; Baier, Haverkort, Hermanns, and Katoen, 2003]).
- It provides us with a (Monte-Carlo) sampling algorithm (via a random walk on the state space).
- There is a rich pool of tools and theorems for their theoretical and numerical analysis, to name a few: [Bortolussi and Hillston, 2012; Baier, Haverkort, Hermanns, and Joost-Pieter Katoen, 2000; Derisavi et al., 2003; Joost-Pieter Katoen et al., 2001].

Curse of Dimensionality. The size of the CTMC state space increases exponentially with the number of nodes. For two node-states and 280 nodes, this already means that the cardinality of $\mathcal{X}$ is higher than the number of atoms in the observable universe. Realistic networks can have millions of nodes. This means that methods that rely on numerically enumerating all CTMC-states (like the ones in Excursus 2) are only feasible for small toy examples).

CTMC Properties. The semantics of the CTMC itself is a *stochastic process*, that is, a family of random variables $\{X_t\}_{t\in[0,h]}$ where $X_t(\cdot)$ maps a trajectory to the CTMC-state at time t. We consider *time-homogeneous* CTMCs, that is:

$$P(X_t = \mathbf{x}_1 \mid X_0 = \mathbf{x}_0) = P(X_{t+t'} = \mathbf{x}_1 \mid X_{t'} = \mathbf{x}_0)\ ,$$

where $\mathbf{x}_i$ is a CTMC-state (network-state) and $t, t' \geqslant 0$, $t + t' < h$. Intuitively, this means that the absolute timepoint is irrelevant (the probabilities are invariant to time translations).

The *Markov property* of a CTMC is given by the constraint:

$$\begin{aligned} &P(X_{t_n} = \mathbf{x}_n \mid X_{t_{n-1}} = \mathbf{x}_{t-1}) \\ =&P(X_{t_n} = \mathbf{x}_n \mid X_{t_{n-1}} = \mathbf{x}_{t-1}, \ldots, X_{t_0} = \mathbf{x}_0) \end{aligned}$$

for $0 \leqslant t_0 \leqslant t_1 \leqslant \cdots \leqslant t_n \leqslant h \in \mathbb{R}$ and where each $\mathbf{x}_i$ denotes a CTMC-state. Intuitively, this means that the history of a trajectory does not matter, only the current system state is relevant for the future evolution.

2.5.1 CTMC Simulation

Simulating a multi-state process using Algorithm 2 and simulating the corresponding CTMC using Algorithm 3 is *statistically equivalent*. This means that all trajectories have the same likelihood of being generated. This is not surprising as the algorithms are very similar, except that now all outgoing transitions compete in a race condition and not all node-rule pairs.

Algorithm 3: Naïve CTMC Simulation

Input: CTMC-state $x \in \mathcal{X}$; Time t_i.
Output: Successor CTMC-state $y \in \mathcal{X}$; t_{i+1}.

1. Initialize an empty list L.
2. For each outgoing transition from x (i.e., $(x, \lambda_{x,y'}, y') \in \mathcal{T}$ for some $y' \in \mathcal{X}$):
   $\Rightarrow$ Sample a candidate jump-time $t \sim \text{Exp}(\lambda_{x,y'})$.
   $\Rightarrow$ Store (t, y') in L.
3. Among all tuples in L, identify $(\hat{t}, \hat{y})$ with the shortest time and:
   $\Rightarrow$ Return $y \coloneqq \hat{y}$ and $t_{i+1} \coloneqq t_i + \hat{t}$.

Excursus 2: Solving CTMCs

Apart from stochastic simulations, one can numerically solve the CTMC, either in terms of its *transient* or its *steady-state* behavior.

Transient Analysis. In the transient case, we start with an initial CTMC-state and evaluate how the probability mass moves through the CTMC over time. The solution provides us with a probability distribution over all CTMC-states for each point in time. The default way to do this is to solve a system of ordinary differential equations (ODEs); one ODE for each CTMC-state. Alternatives are *uniformization* (where the CTMC is turned into a discrete-time process) and solving the matrix exponential (which contains the transition rates). We refer to [Trivedi, 2008] for an overview. We use transient analysis in Chapter 8 (*From Networks to Population Models*).

Steady-State Analysis. Steady-state analysis aims at finding the distribution over CTMC-states in the long-term limit $t \to \infty$. This *equilibrium* distribution is not necessarily unique (it could depend on the initial CTMC-state). In this thesis, we do not have this problem by construction of the CTMC. Yet, some spreading models yield a single *absorbing* (or *trap*) state that cannot be left (e.g., all nodes are susceptible); more on this in Chapter 7 (*Birth-Death Process Abstraction*). In equilibrium, the inflow of probability mass of each CTMC-state equals its outflow (*global balance* condition). This gives rise to an equation system (number of equations is the number of CTMC states), which can be solved using Gaussian elimination or alternative methods (with more numerical stability and resource efficiency) [Trivedi, 2008].

Part II

Simulation and Control

3 Simulation of Markovian Spreading

Sampling stochastic spreading processes (i.e., generating trajectories via simulation) is the simplest and most obvious way to study their behavior and identify global properties.

In this chapter, we present CORAL ([Co]ntagion [r]ejection simulation [al]gorithm). CORAL combines the advantages of event-based simulation and rejection sampling. The method outperforms state-of-the-art methods in terms of absolute runtime and scales significantly better with the network size while being statistically equivalent to Algorithm 1. CORAL is implemented in Rust and publicly available[1].

Our **key idea** is to over-approximate the instantaneous rate at which infected nodes infect their neighbors: We assume that all neighbors of all infected nodes are susceptible and correct for this with rejection events. The over-approximations drastically reduce the number of generated events during the simulation, allowing us to generate successor network-states in near-constant time (w.r.t. the network size).

[1] `github.com/gerritgr/Rejection-Based-Epidemic-Simulation`

3.1 Introduction

Until now, we have only discussed simple algorithms (Algorithm 2 and Algorithm 3) that are not suitable for large networks. Specifically, the following problems occur as the networks become larger:

1. The number of reactions in a fixed time interval increases (the residence time in each network-state decreases).
2. The computational cost of each single reaction increases.

This chapter is only concerned with the second problem. Moreover, we focus on the `SIS` epidemic model (Model 1), not on general multi-state models. We discuss possible generalizations at the end of the chapter.

The reason for the increased number of reactions is the race condition. We generate jump time candidates for all nodes and the one with the smallest jump time "wins" (cf. item (3) in Algorithm 2). Naturally, this minimum tends to be smaller as the number of nodes increases. Thus, to generate a trajectory until a fixed time horizon is reached, we need more reactions.

The increased costs of each reaction are due to the fact that we have to iterate over all the nodes. Moreover, for each node, we have to scan the whole neighborhood of the node (e.g., identify the number of infected neighbors). The costs for this depend on the concrete graph representation (e.g., adjacency matrix or edge list) but increase either way. Reducing the number of reactions in a trajectory (i.e., approaching the first problem) is not possible when we aim for statistical equivalence.

Next, we present methods from the literature that can be used to simulate `SIS` processes. After that, we present our method and demonstrate its effectiveness in three case studies.

3.2 Methods in Literature

In this section, we briefly review techniques that have been previously suggested for the simulation of SIS-type processes. For a more comprehensive description, we refer the reader to [István Z Kiss et al., 2017; Cota and Ferreira, 2017].

A common way to reduce computational complexity when simulation Markovian systems is the so-called *Gillespie algorithm* (GA) or *Gillespie's direct method*. It was proposed by Gillespie (1977) (and partially before that by Doob (1945)) in order to simulate stochastic chemical reaction networks.

Algorithm 4: Gillespie-Based SIS Simulation (GA)

Input: Rates α, β; List of infected nodes $\mathcal{L}_I$; List of SI-edges $\mathcal{L}_{SI}$, Time t_j.
Output: Time t_{j+1}; Updated $\mathcal{L}_I$, $\mathcal{L}_{SI}$.

1. Compute $\lambda_{all} \coloneqq \alpha|\mathcal{L}_I| + \beta|\mathcal{L}_{SI}|$.
2. Sample $t \sim \mathrm{Exp}(\lambda_{all})$, set $t_{i+1} \coloneqq t_i + t$.
3. Sample $u \sim \mathrm{U}(0,1)$.
4. **If** $u < \frac{\alpha|\mathcal{L}_I|}{\lambda_{all}}$: ▷ Recovery
   - ⇒ Sample a recovering node v_i from $\mathcal{L}_I$ (uniformly).
   - ⇒ Update $\mathcal{L}_I$ and $\mathcal{L}_{SI}$ to account for v_i being susceptible.

   Else: ▷ Infection
   - ⇒ Sample an edge that transmits an infection (v_i, v_j) from $\mathcal{L}_{SI}$ (uniformly).
   - ⇒ Update $\mathcal{L}_I$, $\mathcal{L}_{SI}$ to account for v_j being infected.

3.2.1 Standard Gillespie Algorithm (GA)

Idea. The key idea is to reduce the number of random samples necessary to generate a reaction. Instead of generating a jump time candidate for each I-node and SI-edge (as in Algorithm 1), we directly sample the residence time (i.e., the minimum of all jump time candidates). This is only possible when we know the exact number of I-nodes and SI-edges in the network. We adjust the data structures accordingly. Consequently, this allows us to update the network-state without iterating over the whole contact network.

Method. We use as key data structures two lists that are constantly updated: a list of all infected nodes ($\mathcal{L}_{\mathtt{I}}$) and a list of all SI-edges ($\mathcal{L}_{\mathtt{SI}}$).

In each simulation step, we first draw an exponentially distributed delay for the time until the next rule is invoked ("fires"). For this, we compute an aggregated rate $\lambda_{\text{all}} = \alpha|\mathcal{L}_{\mathtt{I}}| + \beta|\mathcal{L}_{\mathtt{SI}}|$ (where α and β denote the recovery and infection rate, respectively). Then we randomly decide if an infection or a recovery event is happening. The probability of the latter is proportional to its rate, i.e. $\frac{1}{\lambda_{\text{all}}}\alpha|\mathcal{L}_{\mathtt{I}}|$, and thus, the probability of an infection is $\frac{1}{\lambda_{\text{all}}}\beta|\mathcal{L}_{\mathtt{SI}}|$. After that, we pick an infected node (in case of a recovery) or an SI-edge (in case of an infection) uniformly at random. We update the two lists accordingly (cf. Algorithm 4).

Runtime. The expensive part of each step is keeping $\mathcal{L}_{\mathtt{SI}}$ updated. For this, we iterate over the whole neighborhood of the changing node, and for each susceptible neighbor, we remove (after a curing) or add (after an infection) the corresponding edge to/from the edge list. Thus, we need one add/remove operation on the list for each susceptible neighbor.

Correctness. The correctness follows from the properties of the exponential distribution. Consider m random variables; each following an exponential distribution with rate $\lambda_1, \ldots, \lambda_m$, respectively. Their minimum also follows an exponential distribution with rate $\sum_i \lambda_i$ [A. O. Allen, 1990]. Thus, we can use λ_{all} to sample the residence time. The probability that the random variable j wins the race is proportional to λ_j (i.e., $\lambda_j / \sum_i \lambda_i$). Likewise, the cumulative probability that a recovery (resp. infection) events "fires" first is $\frac{1}{\lambda_{all}} \alpha |\mathcal{L}_I|$ (resp. $\frac{1}{\lambda_{all}} \beta |\mathcal{L}_{SI}|$). We can then pick the specific node (resp. edge) uniformly at random because all recovery (resp. infection) events have the same rate and, therefore, equal probability of "winning" the race.

3.2.2 Optimized Gillespie Algorithm (OGA)

The *Optimized Gillespie Algorithm* (OGA) was introduced by Cota and Ferreira (2017) as a rejection-based extension of GA. Here, we only discuss the basic version of the algorithms.

Idea. The idea is to forget about $\mathcal{L}_{SI}$ and only store a list of $\mathcal{L}_I$. To sample infection events, one samples the starting point of the infection directly from $\mathcal{L}_I$. Thus, to generate an infection event, we first sample an infected node from $\mathcal{L}_I$ and then we (uniformly) sample a susceptible neighbor, which becomes infected. Since infected nodes with many susceptible neighbors have a higher probability of being the starting point of infection (i.e., they have more SI-edges associated with them), we cannot sample uniformly. Instead, we sample from $\mathcal{L}_I$ so that the probability of picking an infected node is proportional to its number of susceptible neighbors.

This is a problem because non-uniform sampling from $\mathcal{L}_I$ is expensive. Particularly, because after each reaction, the number of susceptible

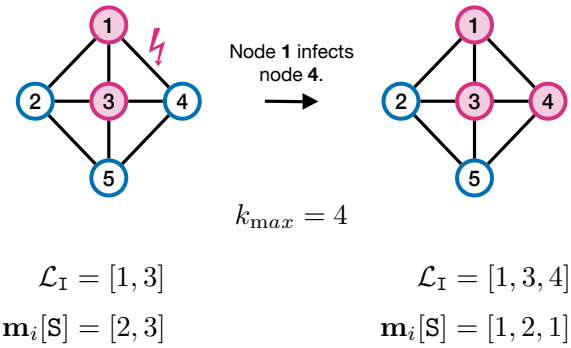


Fig. 3.1.: OGA: Example of an infection event starting from node 1. We sample from $\mathcal{L}_I$ proportional to $\mathbf{m}_i[S]$. To this end, we first sample uniformly at random from $\mathcal{L}_I$. Then we correct for the uniformity assumption using a rejection step. The rejection probability in the depicted step for node 1 is $\frac{4-2}{4} = 50\%$. The rejection probability for node 3 would be $\frac{4-3}{4} = 25\%$.

neighbors (determining the probability to be picked) changes. Thus, annotating $\mathcal{L}_I$ with non-uniform weights and keeping this annotation updated is impractical.

Method. OGA samples nodes from $\mathcal{L}_I$ uniformly at random. Therefore, it uses the maximum degree, k_{max}, of the network and assumes each infected node has k_{max} susceptible neighbors. The maximal degree is a trivial upper bound for the maximal possible number of susceptible neighbors. After sampling an infected node v_i, OGA rejects the node with probability $\frac{k_{max}-k_i}{k_{max}}$. If the node is not rejected, OGA uniformly chooses a neighbor of that node that becomes infected (if possible). If this neighbor is already infected, a rejection event is triggered: Nothing happens except that the global clock is updated. This yields an overall rejection probability of $\frac{k_{max}-\mathbf{m}_i[S]}{k_{max}}$ where $\mathbf{m}_i[S]$ is the number of susceptible neighbors of v_i (following notation from Section 2.4.3). Note that the rejection probability exactly corrects for the over-approximation of using k_{max} instead of $\mathbf{m}_i[S]$. This is illustrated in Figure 3.1 and in Algorithm 5.

Algorithm 5: Optimized Gillespie Algorithm (OGA)

Input: Rates α, β; Lists $\mathcal{L}_I$; Time t_j.
Output: Time t_{j+1}; Updated $\mathcal{L}_I$.

1. Compute $\lambda_{all} \coloneqq \alpha|\mathcal{L}_I| + \beta k_{max}|\mathcal{L}_I|$.
2. Sample $t \sim Exp(\lambda_{all})$, set $t_{i+1} \coloneqq t_i + t$.
3. Sample $u \sim U(0,1)$.
4. **If** $u < \frac{\alpha|\mathcal{L}_I|}{\lambda_{all}}$: ▷ Recovery
   - ⇒ Sample a recovering node v_i from $\mathcal{L}_I$ (uniformly).
   - ⇒ Remove v_i from $\mathcal{L}_I$.

   Else: ▷ Infection
   - ⇒ Sample a node from $\mathcal{L}_I$ (uniformly).
   - ⇒ Return (reject) with probability $\frac{k_{max}-k_i}{k_{max}}$.
   - ⇒ Sample a neighbor v_j from v_i (uniformly at random).
   - ⇒ If v_j is susceptible: add v_j to $\mathcal{L}_I$.

Runtime. Compared to the GA, updating the list of infected nodes becomes cheaper because only the node that actually changes its state is added to (or removed from) $\mathcal{L}_I$. Naturally, the speed-up in each step comes at the cost (of a potentially enormous amount) of rejection events. Even a single high-degree node will continuously lead to rejected events. Cota and Ferreira (2017) propose the algorithm for simulations close to the epidemic threshold, where the number of infected nodes is typically small.

In OGA, we have two rejection steps, first based on the degree of a node and second based on the number of susceptible neighbors. St-Onge et al. (2018) point out that in the case of heterogeneous

networks, a binary tree can be used to optimize the first rejection step and speed up the sampling significantly. This allows them to derive an upper bound for the rejection probability. However, this does not help when most neighbors of infected nodes are also infected. In this case, most of the events will still be rejection events, which slows down the simulation.

Correctness. Cota and Ferreira show the correctness of their method based on so-called *phantom processes*. We utilize a similar technique later in this chapter.

3.2.3 Event-Based Simulation

In the event-driven approach, the primary data structure is an event queue, in which events are sorted and executed according to the timepoints at which they will occur. This eliminates the costly process of randomly selecting a node for each step (popping the first element from the queue has constant time complexity). Events refer to either the recovery of a specific node or an infection through a specific edge. Moreover, it is easy to adapt the event-driven approach to rules with non-Markovian waiting times or to a network where each node has individual recovery and infection rates [István Z Kiss et al., 2017]. A variant of this algorithm can also be found in [Sahneh et al., 2017].

Method. Event-based simulation of an `SIS` process is done as follows: For the initialization, we draw for each infected node an exponentially distributed time until recovery with rate α and add the respective curing event to the queue. Likewise, for each susceptible node v_i (with at least one infected neighbor), we generate an infection event with rate $\beta \mathbf{m}_i[\mathtt{I}]$. During the simulation, we always

take the earliest event from the queue, change the network state accordingly, and update the global clock t. After a node v_i becomes infected, the infection rates of its susceptible neighbors increase. Thus, it is necessary to iterate over all (susceptible) neighbors of v_i, draw renewed waiting times for their infection events, and update the event queue accordingly. Although improvements have been suggested [István Z Kiss et al., 2017], these queue updates are rather costly.

Runtime. Since each step requires an iteration over all neighbors of the current node, the worst-case runtime depends on the maximal degree of the network. Moreover, for each neighbor, it might be necessary to reorder the event queue. The time complexity of reordering the queue depends (typically logarithmically) on the number of elements in the queue and adds high additional costs to each step.

Correctness. Trajectories generated using the event-driven approach are statistically equivalent to those generated with $O(GA)$. Here, we give a short intuition and refer to [István Z Kiss et al., 2017]. We can think of this method as a version of Algorithm 2 where we do not clear the event list after every single step. We only generate new jump time candidates for nodes whose neighborhood has changed. This is possible because—due to the memoryless property of the exponential distribution—re-sampling the waiting times (jump-time candidates) yields the same probability density as using the old ones.

3.3 Our Method: CORAL

In this section, we propose our method, CORAL, for the simulation of SIS-type processes. The key idea is to combine an event-driven approach with rejection sampling while keeping the number of rejections to a minimum. First, we introduce the main data structures:

Event Queue

It stores all future (infection and curing) events generated so far. Each event is associated with a timepoint and the affected node(s). Curing events contain a reference to the recovering node. Infection events refer to an infected (source) node and a susceptible (target) node.

Graph

In the graph data structure, each node is associated with a list of neighbors, its current node-state, a degree, and, if infected, a prospective recovery time.

We also keep track of the time in a global clock. We assume that an initial network-state, a time horizon (or any other stopping criterion), and the rate parameters (α, β) are given as input. In Alg. 1-4 (p. 53), we provide pseudocode for the detailed steps of the method.

Initialization

Initially, we iterate over the network and sample a recovery time (exponentially distributed with rate α) for each infected node (Line 2, Alg. 1). We push the recovery event to the queue and annotate each infected node with its recovery time (Line 5, Alg. 2). Next, we iterate over the network a second time and generate an infection event for each infected node (Line 5, Alg. 1). The procedure for generating infection events is explained later.

We need two iterations because each infected node's recovery time must be available for the infection events. These events identify the earliest infection attempt of each node.

Iteration

The main procedure of the simulation is illustrated in Alg. 4. We schedule events until the global clock reaches the specified time horizon (Line 9). In each step, we take the earliest event from the queue (Line 7) and set the global clock to the event time (Line 8). Then we apply the event (Line 11-20).

In case of a recovery event, we simply change the state of the corresponding node from `I` to `S` and are done (Line 12). Note that we always generate (exactly) one recovery event for each infected node. Thus, each recovery event is always consistent with the current network-state. Note that the queue always contains precisely one recovery event for each infected node.

If the event is an infection event, we apply it if possible (Line 14-18) and reject it otherwise (Line 19-20). We update the global clock either way. Each infection event is associated with a source node and a target node. The infection event is applicable if the current state of the target node is `S` (which might not be the case anymore) and the current state of the source node is `I` (which will always be the case). After a successful infection event, we generate a new recovery event for the target node (Line 16) and two novel infection events: one for the source node (Line 17) and one for the target node that is now also infected (Line 18). If the infection attempt was rejected, we only generate a new infection event for the source node (Line 20). Thus, we always have precisely one infection event in the queue for each infected node.

Generating Infection Events

The generation of infection events and the distinction between *unsuccessful* and *potentially successful* infection attempts are an essential part of the algorithm.

In Alg. 3, for each infected node, we only generate the earliest infection attempt and add it to the queue. Therefore, we first sample the exponentially distributed waiting time for node v_i with rate $k_i \beta$ (k_i is the degree of v_i) and compute the timepoint of infection (Line 5). If the timepoint of the infection attempt is after its recovery event, we stop, and no infection event is added to the queue (Lines 6-7). Note that in the graph structure, each node is annotated with its recovery time (`node.recovery_time`) to have it immediately available.

Next, we uniformly select a random neighbor that will be attacked (Line 8). If the neighbor is currently susceptible, we add the event to the event queue and the current iteration step ends (Lines 9-12).

If the neighbor is currently infected, we check the recovery time of the neighbor (Line 9). If the infection attempt occurs before the recovery timepoint, we already know that the infection attempt will fail (already infected nodes cannot become infected). Thus, we perform an *early reject* (Lines 10-12 are not executed). That is, instead of pushing the surely unsuccessful infection event to the queue, we directly generate another infection attempt, i.e., we re-enter the while-loop in Lines 4-12. We repeat the above procedure until the recovery time of the current node is reached or the infection can be added to the queue (i.e., no early rejection is happening).

Figure 3.3 (p. 54) provides a basic example of a possible execution of our method.

Algorithm 1 Graph Initialization

```
1: procedure InitGraph(G, α, β, Q)
2:     for each node in G do
3:         if node.state = I then
4:             GenerateRecoveryEvent(node, μ, 0, Q)
5:     for each node in G do                          ▷ recovery times are available now
6:         if node.state = I then
7:             GenerateInfectionEvent(node, β, 0, Q)
```

Algorithm 2 Generation of a Recovery Event

```
1: procedure GenerateRecoveryEvent(node, α, t_global, Q)
2:     t_event = t_global + draw_exp(μ)
3:     e = Event(src_node = node, t=t_event, type=recovery)
4:     node.recovery_time = t_event
5:     Q.push(e)
```

Algorithm 3 Generation of an Infection Event

```
1: procedure GenerateInfectionEvent(node, β, t_global, Q)
2:     t_event = t_global
3:     rate = β*node.degree
4:     while true do
5:         t_event += draw_exp(rate)
6:         if node.recovery_time < t_event then                ▷ no event is generated
7:             break
8:         attacked_node = draw_uniform(node.neighbor_list)
9:         if attacked_node.state = S
               or attacked_node.recovery_time < t_event then   ▷ check for early reject
10:            e = Event(src_node=node, target=attacked_node,
                         time=t_event, type=infection)
11:            Q.push(e)                                       ▷ was successful
12:            break
```

Algorithm 4 SIS Simulation

```
    Input: Graph (G) with initial states, time horizon (h), recovery rate (α), infection rate (β)
    Output: Graph at time h                                    ▷ or any other measure of interest
1:  Q = emptyQueue()                                           ▷ sorted w.r.t. time
2:  InitGraph(G, α, β, Q)
3:  t_global = 0
4:  while true do
5:      if Q.is_empty() then
6:          break
7:      e = Q.pop()
8:      t_global = e.time
9:      if t_global > h then
10:         break
11:     if e.type = recovery then
12:         G[e.src_node].state = S
13:     else
14:         if G[e.target_node].state = S then
15:             G[e.target_node].state = I
16:             GenerateRecoveryEvent(e.target_node, α, t_global, Q)
17:             GenerateInfectionEvent(e.src_node, β, t_global, Q)
18:             GenerateInfectionEvent(e.target_node, β, t_global, Q)
19:         else                                               ▷ late reject
20:             GenerateInfectionEvent(e.src_node, β, t_global, Q)
```

Fig. 3.2.: Pseudocode for our event-based rejection sampling method CORAL.

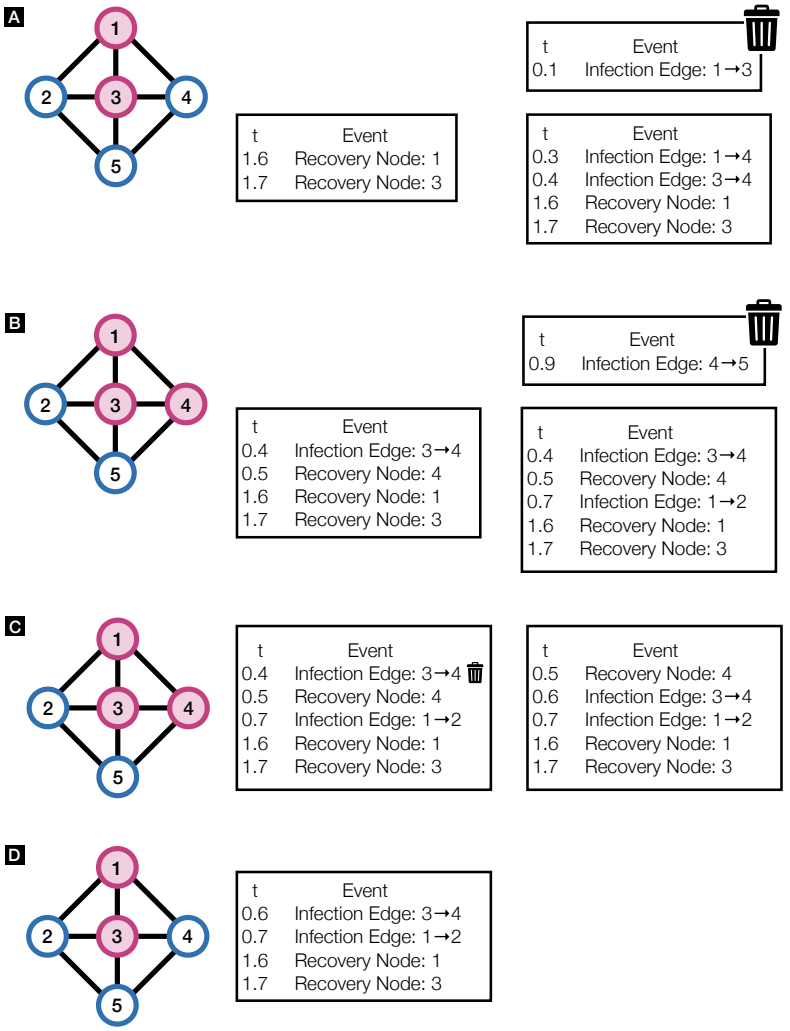


Fig. 3.3.: First four steps of our method for a toy example (I: magenta, S: blue). **(a)**: Initialization, generate recovery events (left queue), and infection event for each infected node (right queue). The first infection attempt from node 1 is an early reject. **(b)**: The infection from 1 to 4 was successful, we generate a recovery event for 4 and two new infection events for 1 and 4. The infection event of node 4 is rejected directly because it occurs after recovery. **(c)**: (Late) Reject of the infection attempt from 3 to 4 as 4 is already infected. A new infection event starting from 3 is inserted into the queue. **(d)**: Node 4 recovers, the remaining queue is shown.

3.3.1 Analysis

Our approach combines the advantages of an event-based simulation with the advantages of rejection sampling. In contrast to the *Optimized Gillespie Algorithm*, finding the node for the next event can be done in constant time. More importantly, the number of rejection events is dramatically minimized because the queue only contains realistically possible events. Therefore, it is crucial that each node "knows" its own recovery time and that recovery events are always generated before infection events. In contrast to traditional event-based simulation, we do not have to iterate over all neighbors of a newly infected node followed by a potentially costly queue reordering.

Runtime

For the runtime analysis, we assume that a binary heap is used to implement the event queue and that the graph structure is implemented using a hashmap. Each simulation step starts by taking an element from the queue (cf. Line 7, Alg. 4), which can be done in constant time. Applying the change of state to a particular node has constant time complexity on average and linear time complexity (in the number of nodes) in the worst case, as it is based on lookups in the hashmap.

Generating a waiting time (Line 3, Alg. 3) can be done in constant time because we know the degree (and therefore the rate) of each node. Likewise, sampling a random neighbor (Line 8) is constant in time (assuming the number of neighbors fits in an integer). Checking for an *early reject* (Line 9) can also be done in constant time because each neighbor is sampled with the same (uniform) probability and is annotated with its recovery time. Although each early rejection can be computed in constant time, the number of early rejections

can increase with the mean (and maximum) degree of the network. Inserting the newly generated infection event(s) into the event queue (Line 11) has a worst-case time complexity of $\mathcal{O}(\log n_H)$, where n_H is the number of elements in the heap. In our case, n_H is bounded by twice the number of infected nodes. However, we can expect constant insertion costs on average [Hayward and McDiarmid, 1991; Porter and I. Simon, 1975].

Correctness

Here, we argue that our method generates correct sample trajectories of the underlying Markov model. To see this, we assume some hypothetical changes to our method that do not change the sampled trajectories but make it easier to reason about the correctness. First, assume that we abandon *early rejects* and insert all events into the event queue regardless of their possibility of success. Second, assume that we change the generation of infection events such that we do not only generate the earliest attempt but all infection attempts until recovery of the node. Note that we do not do this in practice, as this would lead to more rejections (less early rejections).

Similar to Cota and Ferreira (2017), we find that our algorithm is equivalent to the direct event-based implementation of the following spreading process:

$$\mathtt{I} \xrightarrow{\mu} \mathtt{S} \qquad \mathtt{S} + \mathtt{I} \xrightarrow{\beta} \mathtt{I} + \mathtt{I} \qquad \mathtt{I} + \mathtt{I} \xrightarrow{\beta} \mathtt{I} + \mathtt{I}\,.$$

In [Cota and Ferreira, 2017], $\mathtt{I} + \mathtt{I} \xrightarrow{\beta} \mathtt{I} + \mathtt{I}$ is called a *shadow process*, because the application of this rule does not change the network-state. Hence, rejections of infections in the SIS model can be interpreted as applications of the shadow process. Note that the rate at which this rule is applied to the network is the rate of the rejection events. Hence, the rate at which an infected node v_i

attacks its neighbors (no matter whether it is in I or S) is exactly βk_i. Our method includes the shadow process in our simulation in the following way: For each SI-edge and II-edge, an infection event is generated with rate β and inserted into the queue. The decision if this event will be a real or a "shadow infection" is postponed until the event is applied. This is possible because both rules have the same rate.

Model 2: SIRS

In the SIRS model, infected (I) nodes infect their susceptible (S) neighbors. Infected nodes can also recover and gain temporal immunity. That is, they become recovered (R) before they become susceptible again.

States:	$\mathcal{S} = \{S, I, R\}$	
Parameters:	Recovery rate:	$\alpha_1 \in \mathbb{R}_{\geqslant 0}$
	Infection rate:	$\beta \in \mathbb{R}_{\geqslant 0}$
	Immunity loss rate:	$\alpha_2 \in \mathbb{R}_{\geqslant 0}$
Rules:	Recovery:	$I \xrightarrow{\alpha_1} R$
	Infection:	$S \xrightarrow{\beta \mathbf{m}[S]} I$
	Immunity loss:	$R \xrightarrow{\alpha_2} S$

3.3.2 Generalizations

So far, we have only considered SIS processes on static and unweighted networks. This section discusses how to generalize our simulation method to SIS-type processes on temporal and weighted networks.

General Epidemic Models

A key ingredient to our algorithm is the early rejection of infection events. This is possible because we can compute a node's curing time already when the node gets infected. In particular, we exploit that there is only one way to leave state I, that is, by applying a node-based rule. This gives us a guarantee about the remaining time in state I. Other epidemic models have a similar structure. For instance, consider the Susceptible-Infected-Recovered-Susceptible (SIRS) model (Model 2), where infected nodes first become recovered (immune), before entering state I again.

We also consider the competing pathogens model of Masuda and Konno (2006) (Model 3), where two infectious diseases compete over the susceptible nodes.

Model 3: Competing Pathogens

Two pathogens I and J compete over the susceptible nodes. A similar model can be found in [Masuda and Konno, 2006].

States: $\mathcal{S} = \{\mathtt{S}, \mathtt{I}, \mathtt{J}\}$

Parameters:

Recovery rate I:	$\alpha_{\mathtt{I}} \in \mathbb{R}_{\geqslant 0}$
Recovery rate J:	$\alpha_{\mathtt{J}} \in \mathbb{R}_{\geqslant 0}$
Infection rate I:	$\beta_{\mathtt{I}} \in \mathbb{R}_{\geqslant 0}$
Infection rate J:	$\beta_{\mathtt{J}} \in \mathbb{R}_{\geqslant 0}$

Rules:

Recovery I:	$\mathtt{I} \xrightarrow{\alpha_{\mathtt{I}}} \mathtt{S}$
Recovery J:	$\mathtt{J} \xrightarrow{\alpha_{\mathtt{J}}} \mathtt{S}$
Infection I:	$\mathtt{S} \xrightarrow{\beta_{\mathtt{I}} \mathbf{m}[\mathtt{I}]} \mathtt{I}$
Infection J:	$\mathtt{S} \xrightarrow{\beta_{\mathtt{J}} \mathbf{m}[\mathtt{J}]} \mathtt{J}$

In both cases, we can exploit that certain states (`I`, `J`, `R`) can only be left under node-based rules, and thus their residence time is independent of their neighborhood. This makes it simple to annotate each node in any of these states with their exact residence time and perform early rejections accordingly. Early rejections cannot be applied if we do not have these guarantees.

Weighted Networks

In weighted networks, each edge $e \in \mathcal{E}$ is associated with a positive real-valued weight $w(e) \in \mathbb{R}_{>0}$. Each edge-based rule of the form

$$\mathtt{A} + \mathtt{C} \xrightarrow{\lambda} \mathtt{B} + \mathtt{C}$$

fires on this particular edge with rate $w(e) \cdot \lambda$. Applying our method to weighted networks is simple: During the generation of infection events, instead of sampling the waiting time with rate λk_i, we now use $\lambda \sum_{v_j \in N(v_i)} w(v_i, v_j)$ as the rate ($N(v_i)$ is the set of neighbors of v_i). Moreover, instead of choosing a neighbor that will be attacked with uniform probability, we choose them with a probability proportional to their edge weight. This can be done by rejection sampling, or in $\mathcal{O}(\log(k_i))$ time complexity using a binary tree, or in constant time by pre-computing an inverse-transform for all nodes.

Temporal Networks

Temporal (time-varying, adaptive, dynamic) networks are an intriguing generalization of static networks, which generally complicates the analysis of their spreading behavior [Vestergaard and Génois, 2015; Masuda and Holme, 2017; Holme and Saramäki, 2012; Holme, 2015a]. Generalizing the Gillespie algorithm for Markovian epidemic-type processes is not always trivial [Vestergaard and Génois, 2015].

In order to keep our model as general as possible, we assume here that an external process governs the temporal changes in the network. This process runs simultaneously with our simulation and might or might not depend on the current network-state. It changes the current graph by adding or removing edges, one edge at a time. For instance, after processing one event, the external process could add or remove an arbitrary number of edges at specific timepoints until the time of the next event is reached. It is simple to integrate this into our simulation.

Given that the external process removes an edge, we can update the neighbor list and the degrees. For each infection event that reaches the top of the queue, we first check if the corresponding edge is still present and reject the event otherwise. This is possible because removing edges decreases infection rates which we can correct by using rejections. When an edge is added to the graph, the infection rate might increase. Thus, it is not sufficient to only update the graph. We also generate an infection event which accounts for the new edge. In order to minimize the number of generated events, we change the algorithm such that each infected node is annotated with the timepoint of its subsequent infection attempt. Now consider an infected node. When it obtains a new edge, we generate an exponentially distributed waiting time with rate β modeling the infection attempt through this specific link. We only generate a new event (and remove the old one) if this timepoint lies before the timepoint of the subsequent infection attempt.

Since most changes in the graph do not require changes to the event queue (and those that do only cause two operations at maximum), we expect our method to handle temporal networks very efficiently. In the case that a vast number of edges in the graph change at once, we can always decide to iterate over the whole network and newly initialize the event queue.

3.4 Case Studies

We demonstrate the effectiveness of our approach on three classical epidemic-type processes.

Setup. We use synthetically generated networks following the configuration model [Fosdick et al., 2018] with a truncated power-law degree distribution, that is, $P(k) \propto k^{-\gamma}$ for $3 \leqslant k \leqslant 1000$. We compare the performance using degree distributions with $\gamma \in \{2, 3\}$. This yields a mean degree around 30 ($\gamma = 2$) and 10 ($\gamma = 3$). We use models from the literature but adapt rate parameters freely to generate interesting dynamics. Nevertheless, we find that our observations generalize to a wide range of parameters that yield networks with realistic degree distributions and spreading dynamics.

We also report how the number of nodes in a network is related to the CPU time of a single step. This is more informative than using the total runtime of a simulation because the number of steps obviously increases with the number of nodes when the time horizon is fixed. The CPU time per step is defined as the total runtime of the simulation divided by the number of steps; only counting the steps that actually change the network-state (i.e., excluding rejections). We do not count rejection events because that would give unfair advantages to rejection-based approaches. The evaluation was performed on a 2017 MacBook Pro with a 3.1 GHz Intel Core i5 CPU and 16 GB of RAM.

Baseline. We compared the performance of our method with the *Standard Gillespie Algorithm* (GA) and the *Optimized Gillespie Algorithm* (OGA) for different network sizes. Note that an implementation of the OGA was only available for the `SIS` model, and the comparison is therefore not available for other models. Due to the high number

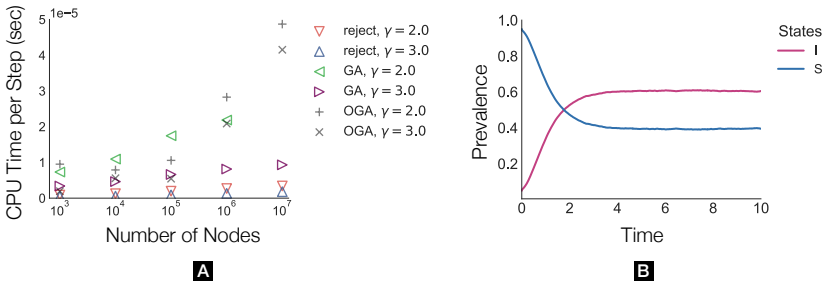


Fig. 3.4.: SIS model **(a)**: [Lower is better.] Average CPU time for a single step (i.e., change of network-state) for different networks. *reject* refers to CORAL. The GA method run out of memory for $\gamma = 2.0$, $n = 10^7$. **(b)**: Example dynamics for a network with $\gamma = 3.0$ and 10^5 nodes.

of rejection steps in all models, we expect a similar difference in the performance between our approach and OGA for other models.

Experiment 1: SIS Model

For the SIS model (Model 1) we used rate parameters of $(\alpha, \beta) = (1.0, 0.6)$ and an initial distribution of 95% susceptible nodes and 5% infected nodes. CPU times are reported in Figure 3.4a. For a sample trajectory, we plot the fraction of nodes in each state w.r.t. time (Figure 3.4b). To compare with OGA, we used the official Fortran implementation from [Cota and Ferreira, 2017] and estimated the average CPU time per step based on the absolute runtime. Note that the comparison is not perfectly fair due to implementation differences and additional input/output of the OGA code. It is not surprising that OGA performs comparably poorly, as the method is suited for simulations close to the epidemic threshold. Moreover, our maximum degree is huge, which negatively affects the performance of OGA.

We also carried out experiments on models closer to the epidemic threshold (i.e., where the number of infection events is very small, e.g., $\beta = 0.1$) and with smaller maximal degree (e.g., $k_{max} = 100$).

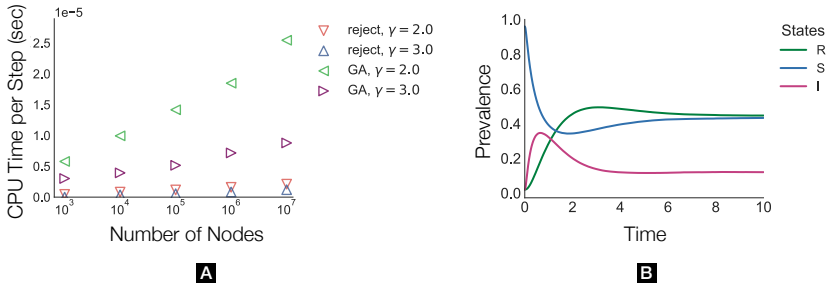


Fig. 3.5.: SIR model **(a)**: [Lower is better.] Average CPU time for a single step (i.e., change of network-state) for different networks. *reject* refers to CORAL. **(b)**: Example dynamics for a network with $\gamma = 2.0$ and 10^5 nodes.

The relative speed-up to GA increased slightly compared to the results in Figure 3.4 (p. 62). The performance of OGA improved significantly, leading to similar performance as our method (results not shown).

Experiment 2: SIRS Model

Next, we considered the SIRS model (Model 2), which admits more complex dynamics. We used rate parameters of $(\alpha, \alpha_2, \beta) = (1.1, 0.3, 0.6)$ and an initial distribution of 96% susceptible nodes and 2% infected and recovered nodes, respectively. As above, CPU times and example dynamics are reported in Figure 3.5. We see that the run-time behavior is almost the same as in the SIS model.

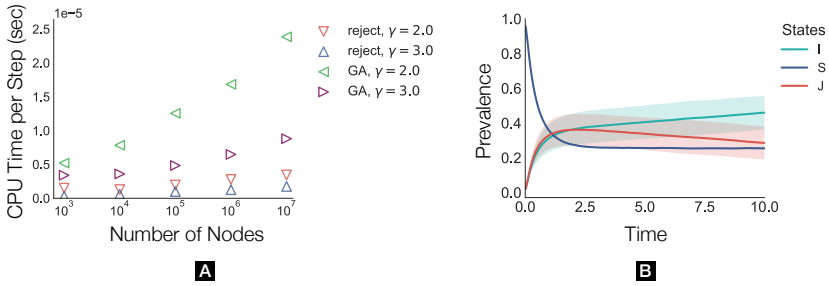


Fig. 3.6.: Competing pathogens model **(a)**: [Lower is better.] Average CPU time for a single step (i.e., change of network-state) for different networks. *reject* refers to CORAL. **(b)**: Mean fractions and standard deviations of a network with $\gamma = 2.0$ and $n = 10^4$.

Experiment 3: Competing Pathogens Model

Finally, we considered the Competing Pathogens model (Model 3). We used rate parameters of $(\beta_I, \beta_J, \alpha_1, \alpha_2) = (0.6, 0.63, 0.6, 0.7)$ and an initial distribution of 96% susceptible nodes and 2% infected nodes for both pathogens `I` and `J`, respectively. CPU times and network dynamics are reported in Figure 3.6. The model is interesting because we see that in the beginning `J` dominates `I` due to its higher infection rate. However, nodes infected with pathogen `J` recover faster than those infected with `I`. This gives the `I` pathogen the advantage that infected nodes have more time to attack their neighbors. In the limit, `I` takes over and `J` dies out. For this model, stochastic noise has a significant influence on the macroscopic dynamics. Therefore, we also reported the standard deviation of the fractions. Note that the fraction of susceptible nodes is almost deterministic. Performance-wise, our rejection method performs slightly worse than in the previous models (w.r.t. the baseline). We believe this is due to the even greater number of infection events and rejections.

3.5 Conclusions

This chapter presented a novel rejection algorithm for the simulation of epidemic-type processes. We combined the advantages of rejection sampling and event-driven simulation. In particular, we exploited that nodes can only leave certain states using node-based rules, which made it possible to pre-compute their residence times, allowing us to perform early rejection of certain events.

Our numerical results show that our method outperforms previous approaches, especially for network dynamics that is not close to the epidemic threshold. In particular, the speed-up increases with the maximum degree of the network.

In future work, we plan to extend the method to compartment models with arbitrary rules, including an automated decision for which states early rejections can be computed and are useful.

4 Simulation of non-Markovian Spreading

The last chapter discussed how to simulate Markovian spreading processes on networks. Here, we propose a rejection-based algorithm for non-Markovian spreading called RED[1] ([R]ejection-based [E]vent-[D]riven simulation algorithm). Non-Markovian means that we do not restrict ourselves to exponentially distributed residence times.

Our **key idea** is to encode the probability density between node events as a rate function. The rate function gives us a memoryless-like interpretation and makes rejection sampling possible. In turn, rejection sampling allows over-approximating the rate function. We can exploit this to drastically reduce the number of required updating steps similar to the previous chapter.

RED scales exceptionally well with the size and connectivity of the underlying contact network and produces statistically correct samples. Moreover, we develop a powerful formalism to model non-Markovian interactions between nodes. Therefore, we equip each node with an internal clock and allow rules to take the local time into account. This is the only chapter that covers non-Markovian processes. Conceptually, we also adjust our perspective on nodes and consider them as *agents* that actively change their node-state instead of being passively changed by a set of rules.

[1] `github.com/gerritgr/non-markovian-simulation`

4.1 Introduction

In the Markovian models from the last chapters, the following properties hold (giving rise to the CTMC semantics, cf. Section 2.5, p. 33):

- Nodes have exponentially distributed residence times.
- The instantaneous rate at which a node changes ($f(\mathbf{m}_i)$ in Algorithm 2) is constant over time (as long as the neighborhood does not change).
- The successor node-state is independent of the jump time (cf. Section 3.2.1).

The three properties are related to the specifics of the exponential distribution.

Instantaneous Rates and Memorylessness

Consider an exponentially distributed random variable with rate λ. This rate is also called *instantaneous* (reaction) rate and is directly related to the *memorylessness* of each node. This makes sense when we consider the x axis as time and a variate as the specific firing time (or jump time). Then, the instantaneous rate λ is proportional to the probability that the process fires in the next infinitesimal time unit. Particularly, the instantaneous rate is constant (independent of the time that has already passed). Hence, nodes or agents in Markovian models are called memoryless, because they do not "remember" how much time they have already spent in their internal state. Likewise, when events compete in a race condition, the probability that a particular event "wins" is also independent of the time that has already passed when the event happens.

Non-Markovian Dynamics

Consider the case that recovered nodes (which we call agents from now on) become susceptible after a uniformly distributed residence time; or the case that infections happen after waiting times that follow a Weibull distribution. This chapter deals with the simulation of stochastic spreading processes where inter-event times are specified by arbitrary probability distributions.

It is long known that it is unrealistic to assume exponentially distributed inter-event times in many real-world scenarios. As empirical results show, this holds, for instance for the spread of epidemics [Lloyd, 2001; Yang, 1972; Blythe and R. Anderson, 1988; Hollingsworth et al., 2008; Feng and Thieme, 2000], opinions in online social networks [Barabasi, 2005; Vázquez et al., 2006], and neural spike trains [Softky and Koch, 1993]. Unfortunately, inter-event times that can follow arbitrary distributions complicate the analysis of such processes.

Often Monte-Carlo simulations are the only feasible way to investigate the emerging dynamics, but even these suffer from high computational costs. Specifically, they often scale poorly with the size of the contact networks. Recently, Masuda and Rocha (2018) introduced the Laplace-Gillespie algorithm (LGA) for the simulation of non-Markovian dynamics on networks. Their work is based on an earlier approach, the non-Markovian Gillespie algorithm (NMGA) developed by Boguná et al. (2014). We will explain both methods in more detail later.

4.1.1 Contribution

This work extends the idea of rejection-based simulation to networked systems that admit non-Markovian behavior.

We propose RED based on three main ideas:

1. We express the distributions of inter-event times as time-varying instantaneous rates (referred to as *intensity* or *rate functions*).

2. We sample inter-event times based on an over-approximation of the intensity function, which we counter-balance by using a rejection step.

3. We utilize a priority (resp. event) queue to decide which agent fires next.

Combining these ideas reduces the computational costs of each simulation step. More precisely, if an agent transitions from one local state to another, no update of neighboring agents is required, even though their instantaneous rates might change due to the event. In short, the reason for that is that (by using the rate over-approximation), we always assume the "worst-case" behavior of an agent. If a neighboring agent is updated, the (actual) instantaneous rate of an agent might change, but it will never exceed the rate over-approximation, which was used to sample the firing time. Hence, the sampled firing time is always an under-approximation of the true one, regardless of what happens to adjacent agents.

Naturally, this comes with a cost, in our case rejection (or null) events. Rejection events counter-balance the over-approximation of the instantaneous rate. The larger the difference between the actual rate and the over-approximated rate, the more rejection events will occur. Hence, in combination, rejections and over-approximations yield a statistically correct algorithm.

Differences to CORAL

For clarity, we briefly point out the differences and similarities between RED and CORAL from the previous chapter. Both methods

- use a priority queue to store future events;
- use rejection events to counter-balance over-approximations;
- minimize the computational costs of updating neighboring nodes after each event.

However, RED

- is not limited to `SIS`-type models;
- does not use early rejections (they are `SIS`-specific);
- generates events only for the currently active agent.

Most importantly, RED allows arbitrary inter-event time distributions.

4.2 Multi-Agent Model

Here, we introduce our formalism for agent-based dynamics on complex networks. Our goal is to have a framework that is as expressive as possible while remaining intuitive. In particular, we want to allow interactions as complex as in the multi-state formalism from Section 2.4.3, but inject it with the possibility for arbitrary inter-event time distributions.

Next, we specify the *network-state*. After that, we explain how the dynamics can be formalized.

Network State

At any given timepoint, the current network-state is described by two functions:

Local State

$L : \mathcal{V} \to \mathcal{S}$ assigns to each agent v_i a local state (node-state) $L(v_i) \in \mathcal{S}$.

Local Clock

$R : \mathcal{V} \to \mathbb{R}_{\geqslant 0}$ equips each agent with a local clock. $R(v_i)$ describes the current residence time of each agent (the time elapsed since the agent changed its local state the last time).

The definition is analogous to Section 2.3 but adds the local clock.

Neighborhood State. Analog to the neighborhood-counting vector in the Markovian case, we define the *neighborhood state*. At any point in time, the neighborhood state $M(v_i)$ of an agent $v_i \in \mathcal{V}$ is a multiset (denoted by double brackets) containing the local states and residence times of all neighboring agents:

$$M(v_i) = \left\{\!\!\left\{ \big(L(v_j), R(v_j)\big) \;\middle|\; (v_i, v_j) \in \mathcal{E} \right\}\!\!\right\} .$$

We use $\mathcal{M}$ to denote the set of all possible neighborhood-states in a given model.

Network Dynamics

We say an agent *fires* when it transitions from one local state to another. The time between two firings of an agent is called *inter-event* time. Moreover, we refer to the remaining time until it fires as its *time delay*. The firing time of an agent v_i depends on its direct neighborhood $M(v_i)$.

Next, we specify how the network-state evolves over time. Therefore, we assign to each agent $v_i \in \mathcal{V}$ two functions $\phi_i(\cdot)$ and $\psi_i(\cdot)$. The intensity function $\phi_i(\cdot)$ governs when v_i fires and the selection function $\psi_i(\cdot)$ defines its successor state. Both functions depend on the local state of v_i (first parameter), the local clock of v_i (second parameter), and the neighborhood $M(v_i)$ (third parameter).

Intensity Function

$\phi_i : \mathcal{S} \times \mathbb{R}_{\geqslant 0} \times \mathcal{M} \to \mathbb{R}_{\geqslant 0}$ defines the *instantaneous rate* of v_i. If $\lambda = \phi_i\big(L(v_i), R(v_i), M(v_i)\big)$, then the probability that v_i fires in the next infinitesimal time interval t_Δ is λt_Δ (assuming $t_\Delta \to 0$).

Selection Function

$\psi_i : \mathcal{S} \times \mathbb{R}_{\geqslant 0} \times \mathcal{M} \to \mathrm{Cat}(\mathcal{S})$ determines which successor state to choose. Here, $\mathrm{Cat}(\mathcal{S})$ denotes the set of all probability distributions over $\mathcal{S}$. If $\mathbf{p} = \psi_i\big(L(v_i), R(v_i), M(v_i)\big)$, then the next local state of v_i is $s \in \mathcal{S}$ with probability $\mathbf{p}[s]$. We evaluate $\psi_i(\cdot)$ at the timepoint when the agent fires.

Typically, $\psi_i(\cdot)$ and $\psi_i(\cdot)$ are the same for all agents, and we can omit their index.

Note that we assume that these functions have no pathological behavior. That is, we exclude the cases in which $\phi_i(\cdot)$ is defined such that it is not integrable or where some intensity function $\phi_i(\cdot)$ would cause an infinite number of simulation steps in finite time (cf. Section 4.2.1).

Putting Everything Together. A multi-agent network model is fully specified by a tuple $(\mathcal{G}, \mathcal{S}, \{\phi_i\}, \{\psi_i\}, L_0(\cdot))$, where $\mathcal{S}$ is the set of local states and $L_0(\cdot)$ specifies the initial state of each agent.

4.2.1 Examples

We provide three examples, from simple to pathological.

Example 4.1: SIS Model in Multi-Agent Framework

Consider the classical (Markovian) SIS model. $\phi(\cdot)$ and $\psi(\cdot)$ are the same for all agents:

$$\phi(s,t,m) = \begin{cases} \alpha & \text{if } s = \mathtt{I} \\ \beta \sum\limits_{(s',t')\in m} \mathbb{1}(s' = \mathtt{I}) & \text{if } s = \mathtt{S}\,. \end{cases}$$

Note that we use an indicator function $\mathbb{1}(\cdot)$ to count the number of infected neighbors. Moreover,

$$\psi(s,t,m) = \begin{cases} \mathtt{I} \to 0,\ \mathtt{S} \to 1 & \text{if } s = \mathtt{I} \\ \mathtt{I} \to 1,\ \mathtt{S} \to 0 & \text{if } s = \mathtt{S}\,. \end{cases}$$

The model is Markovian as neither $\phi(\cdot)$ nor $\psi(\cdot)$ depend on the local clock of any agent. Moreover, $\psi(\cdot)$ is deterministic in the sense that an agent in state I always transitions to S with probability one and vice versa.

Example 4.2: Complex Cascade Model

Consider a modification of the independent cascade model [D'Angelo et al., 2016] where agents are susceptible (S), infected (I), or immune/removed (R). Infected nodes try to infect their susceptible neighbors. The infection attempts can be successful (turning the neighbor from S to I) or unsuccessful (turning the neighbor from S to R). Agents that are in I or R remain there.

The model can, for instance, describe fake news propagation. Infected agents are the ones who re-tweeted misleading information. Someone exposed to the tweed decides if they want to share it. Seeing it multiple times or from multiple sources does not increase the chances of re-tweeting. However, multiple infected friends decrease the time until the tweet is seen.

$$\phi(s, t, m) = \begin{cases} e^{-t_{\text{dist}}} & \text{if } s = \texttt{S} \text{ and } \sum_{(s',t') \in m} \mathbb{1}(s' = \texttt{I}) > 0 \\ 0 & \text{otherwise .} \end{cases}$$

Here, t_{dist} denotes the time elapsed since the latest infected neighbor became infected. Thus, the intensity at which agents "attack" their neighbors decreases exponentially and only the most recently infected neighbor counts. We use $p_i \in [0, 1]$ to denote the probability of a successful infection ($\psi(\cdot)$ is only relevant for susceptible nodes):

$$\psi(s, t, m) = \{\texttt{I} \to p_i,\ \texttt{R} \to 1 - p_i,\ \texttt{S} \to 0\} .$$

This example is both: non-Markovian, because the residence times of the neighbors influence the rate, and non-linear, because the individual effects from neighboring agents do not simply add up.

Example 4.3: Pathological Behavior

Consider two connected agents. Agent v_1 always stays in state I. Agent v_2 switches between states R and S. The frequency at which v_2 alternates increases with the local clock of v_1 (denoted by $R(v_1)$). Let the rate to jump from S to R (and vice versa) be $\frac{1}{1-R(v_1)}$ for $R(v_1) < 1$. Assume that we want to

perform a simulation for the time interval $[0, 1]$. It is easy to see that the instantaneous rate of ν_2 approaches infinity and that the number of simulation steps (state transitions) does not converge.
Generally speaking, pathological behavior may occur if $\phi_i(s, t, m)$ approaches infinity with growing $R(\nu_j)$ (within the simulation time), where ν_j is a neighbor of ν_i. However, it is allowed that $\phi_i(s, t, m)$ approaches infinity with increasing t ($t = R(\nu_i)$) because the agent will eventually fire, and the local clock will be set to zero again.

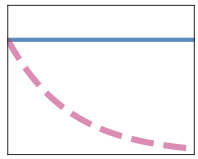
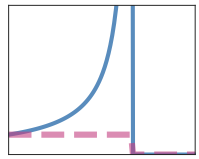
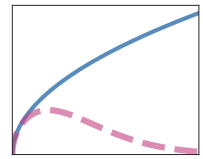
4.3 Semantics of the Multi-Agent Model

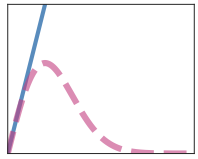
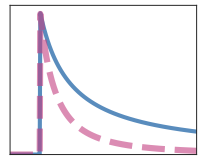
This section specifies the semantics of a multi-agent model in a generative matter (using a simulation algorithm). We start by exploring the intensity function $\psi(\cdot)$.

4.3.1 Intensities and Densities

We used the exponential distribution in the last chapter to compute firing times (with a fixed rate). In the multi-agent framework, we specify our system using time-varying rates. This is advantageous for the combination with rejection sampling. Before we define a first simulation algorithm, we want to establish the relationship between probability densities (here, denoted $\nu(\cdot)$) and intensity functions (here, denoted $\lambda(\cdot)$). Therefore, we leverage the theory of renewal processes [D. R. Cox, 1962; Daley and D. V. Jones, 2003; D. Ma, 2011] and refer the reader to Excursus 3. Some examples of intensity functions $\lambda(\cdot)$ for their corresponding PDFs $\nu(\cdot)$ are shown in Table 4.1 (p. 77).

Tab. 4.1.: Schematic illustration of intensity functions and inter-event time densities.

	Exponential	Uniform	Weibull
y-axis: $\nu(t)$ (dashed), $\lambda(t)$ (solid) **x-axis**: time			
Parameters	$\lambda \in \mathbb{R}_{>0}$	$a, b \in \mathbb{R}_{>0}$ $a < b$	$c, u \in \mathbb{R}_{>0}$
Intensity $\lambda(t)$	λ	$\frac{\mathbb{1}_{t\in[a,b]}}{1-\frac{t}{b-a}}$	$cu(tu)^{c-1}$
PDF $\nu(t)$	$\lambda e^{-\lambda t}$	$\mathbb{1}_{t\in[a,b]}$	$cu(tu)^{c-1}e^{-(tu)^c}$

Rayleigh	Power law
	
$\sigma \in \mathbb{R}_{>0}$	$\alpha, t_{min} \in \mathbb{R}_{>0}$ $\alpha > 1$
$\frac{t}{\sigma^2}$	$\mathbb{1}_{t\geqslant t_{min}} \frac{\alpha-1}{t}$
$\frac{t}{\sigma^2} e^{-\frac{t^2}{2\sigma^2}}$	$\mathbb{1}_{t\geqslant t_{min}} \frac{\alpha-1}{t_{min}} (\frac{t}{t_{min}})^{-\alpha}$

Excursus 3: From Intensities to Densities to Intensities

Consider the *survival function* $S(t)$ of a probability. For a timepoint t, it describes the likelihood that the process has not fired within $[0, t]$. For a probability density $\nu(\cdot)$ (defined over non-negative values), we find that:

$$S(t) = P(T > t) = 1 - \int_0^t \nu(t')dt' .$$

Recall that the intensity function ($\lambda(\cdot)$) determines the instantaneous rate at each point in time. In other words, it is the density, conditioned on the fact that the process has not fired yet. Hence:

$$\lambda(t) = \frac{\nu(t)}{S(t)} .$$

Consequently, we find:

$$\nu(t) = \lambda(t)S(t) .$$

To translate intensities to densities, we define $S(t)$ in terms of $\lambda(t)$ using the following intuition: The intensity function determines the firing probability in an infinitesimal time interval. Hence, the integral over the intensity function determines the probability of firing within a time interval.

If we now consider an exponential distribution with rate 1, the probability that the process does not fire in $[0, t]$ is $S(t) = \exp(-t)$, where t is exactly the integral of the intensity function ($\lambda = 1$) from $[0, t]$. When we now consider a time-varying $\lambda(t)$, we can also express the survival function using an exponential by plugging in the corresponding integral $\lambda(t)$ for t, yielding:

$$S(t) = P(T > t) = \exp\left(-\int_0^t \lambda(t')dt'\right) .$$

Putting everything together, we find the relationship:

$$\lambda(t) = \frac{\nu(t)}{S(t)} = \frac{\nu(t)}{1 - \int_0^t \nu(t')dt'} ,$$

and (4.1)

$$\nu(t) = \lambda(t)S(t) = \lambda(t)\exp\left(-\int_0^t \lambda(t')dt'\right) .$$

Using this equation, we can derive intensity functions from any inter-event time distribution (uniform, log-normal, gamma, power-law, ...). In cases where it is impossible to derive $\lambda(\cdot)$ analytically, we can still compute it numerically. All density functions of time delays can be expressed as time-varying rates (i.e., intensities). However, only intensity functions with an infinite integral can be expressed as a PDF. If $\int_0^\infty \lambda(t)dt$ is finite, the process might not fire at all.

4.3.2 Naïve Simulation Algorithm

Now we specify a naïve simulation algorithm to define the semantics. Recall that the network-state is specified by the mappings $L(\cdot)$ and $R(\cdot)$. Let t_{global} denote the global simulation time (initialized with zero).

The simulation is based on a race condition among all agents: each agent picks a random firing time candidate, but only the one with the shortest time wins and fires (i.e., changes its local state). The global clock only increases by that amount of time. The algorithm initializes a trajectory with $L_0(\cdot)$ and performs steps until a stopping criterion is reached. The computation of the next step is provided in Algorithm 7 and is very similar to Algorithm 2.

This simulation approach is, while being intuitive, very inefficient. Our approach, RED, will be statistically equivalent while being much faster.

Algorithm 7: Naïve non-Markovian Multi-Agent Simulation

Input: Graph $\mathcal{G}$; Network-state $(L(\cdot), R(\cdot))$; Dynamics $\{\phi(\cdot)\}_i$, $\{\psi(\cdot)\}_i$; Global clock t_{global}.
Output: Successor network-state $(L'(\cdot), R'(\cdot))$; Time t'_{global}.

1. Generate a random jump time candidate t_i for each agent v_i. ▷ Cf. Section 4.3.3.
2. Identify the agent v_j with the shortest time t_j.
3. Select a successor state s' for v_j based on $\mathbf{p} = \psi_j\big(L(v_j), R(v_j) + t_j, M(v_j)\big)$.
4. Update
   - $\Rightarrow$ $L'(v_j) \coloneqq s'$ (and $L'(v_i) = L(v_i)$ for all $i \neq j$),
   - $\Rightarrow$ $R'(v_j) \coloneqq 0$, ▷ Reset local clock.
   - $\Rightarrow$ $R'(v_i) \coloneqq R(v_i) + t_j$ (for all $i \neq j$),
   - $\Rightarrow$ $t'_{\text{global}} \coloneqq t_{\text{global}} + t_j$.

4.3.3 Generating Jump Times

This section describes how to generate a jump time candidate t_i for an agent v_i (as used in Line 1 of Algorithm 7). Recall that we encode inter-event time distributions using intensity functions. Thus, we use the intensity function of agent v_i (provided by $\phi_i(\cdot)$) to generate the jump time candidate. There are several ways to do this. For an overview, we refer to [Keeler, 2019; Pasupathy, 2011; Gerhard and Gerstner, 2010; Daley and D. V. Jones, 2003]. Here, we explain three

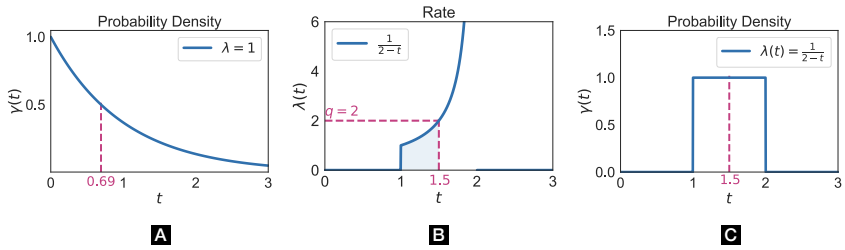


Fig. 4.1.: Sampling event times with an intensity function $\lambda(t) = \frac{\mathbb{1}_{t\in[1,2]}}{2-t}$. **(a)**: Generate a random variate from the exponential distribution with rate $\lambda(t) = 1$ and PDF $\nu(t) = \exp(-t)$, the sample here is 0.69. **(b)**: We integrate the intensity function until the area of 0.69 is reached, here $t_n = 1.5$. **(c)**: This is the intensity function corresponding to the uniform distribution in $\nu(t) = \mathbb{1}_{t\in[1,2]}$.

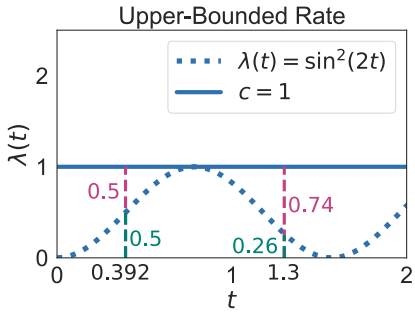


Fig. 4.2.: Rejection sampling example: Sampling t_i from a time-varying intensity function $\lambda(t) = \sin^2(2t)$ using an upper bound of $c = 1$. Two iterations are shown with rejection probabilities shown in red. After one rejection step, the method accepts in the second iteration and returns $t_i = 1.3$.

efficient and straightforward ways. We always assume that the local states of neighboring agents remain the same. This is valid because we only consider the shortest jump time candidate.

Possibility 1: PDFs. An obvious way is to turn the intensity function induced by $\phi_i(\cdot)$ into a PDF (cf. Section 4.3.1) and sample from it using inverse transform sampling.

Possibility 2: Numerical Integration. A more direct way is to perform numerical integration on $\phi_i(\cdot)$. Let us, therefore, define for each v_i the *effective rate* $\lambda_i(\cdot)$ which is the evolution of the intensity $\phi_i(\cdot)$ starting from the current timepoint, assuming no changes in the local states of the neighboring agents:

$$\lambda_i(t_\Delta) = \phi_i\Big(L(v_i), R(v_i) + t_\Delta, M_{t_\Delta}(v_i)\Big),$$
$$\text{where } M_{t_\Delta}(v_i) = \Big\{\Big\{\big(L(v_j), R(v_j) + t_\Delta\big) \mid (v_i, v_j) \in \mathcal{E}\Big\}\Big\}.$$

Here, t_Δ denotes the time evolution "from now on".

The effective rate makes it possible to sample the time delay t_i after which agent v_i fires (if it wins the race), using the inversion transform method. First, we sample an exponentially distributed random variate $x \sim \text{Exp}(1)$, then we integrate $\lambda_i(\cdot)$ to find t_i. Formally, t_i is chosen such that the equation

$$\int_0^{t_i} \lambda_i(t_\Delta) dt_\Delta = x \tag{4.2}$$

is satisfied. The idea is the following: We first sample a random variate x assuming a fixed rate (intensity function) of 1. The corresponding density is $\exp(-x) = S(t)$. Next, we consider the "real" time-varying intensity function $\lambda_i(\cdot)$ and choose $[0, t_i]$ such that

the area under the time-varying intensity function is equal to x (cf. Eq. (4.2)). We know that

$$P(X > x) = \exp(-x) = \exp\left(-\int_0^{t_i} \lambda_i(t_\Delta)\,dt_\Delta\right) = P(Y > t_i)\,.$$

Hence, if x follows an exponential distribution, then t_i is distributed according to the time-varying rate $\lambda_i(\cdot)$ (the corresponding random variable is denoted Y here). In other words, the area under the curve of $\lambda(t_\Delta) = 1$ (from 0 to t_i) follows the same distribution as the area under the curve of $\lambda_i(t_\Delta)$.

Intuitively, by sampling the integral, we a priori define the number of infinitesimal time-steps we take until the agent eventually fires. This number naturally depends on the rate function. If the rate decreases, more steps will be taken. We refer the reader to Pasupathy (2011) for a proof.

Possibility 3: Rejection Sampling. An alternative approach to sample time delays is to use rejection sampling (this is not the rejection sampling, which is the key of the RED method, though) which is illustrated in Figure 4.2 (p. 81). Assume that we have $c \in \mathbb{R}_{\geqslant 0}$ with $\lambda_i(t_\Delta) \leqslant c$ for all t_Δ. We start with $t_i = 0$. Next, we sample a random variate t_i', which is exponentially distributed with rate c. Next, we set $t_i = t_i + t_i'$ and accept t_i with probability $\frac{\lambda_i(t_i)}{c}$. Otherwise, we reject t_i' and repeat the process. Rejection sampling is much faster than numerical integration if a reasonably tight over-approximation can be found. The correctness can be shown similarly to the correctness of RED (Section 4.5.4). That is, one creates a *complementing*- (or *shadow*-process) which accounts for the difference between the upper bound c and $\lambda(t)$.

4.4 Related Work

Most recent work on non-Markovian dynamics studies formal models of such processes and their analysis [I. Z Kiss et al., 2015; Pellis et al., 2015; Jo, Perotti, et al., 2014; Sherborne et al., 2016; Starnini et al., 2017]. Research has focused primarily on how specific distributions (e.g., uniformly distributed curing times) alter the behavior of the epidemic spreading, for instance, the epidemic threshold (see [Pastor-Satorras, Castellano, et al., 2015; István Z Kiss et al., 2017] for an overview). Most of this work is, however, rather limited in scope: Only certain distributions or networks with infinite nodes or specific properties are considered, not the emerging dynamics. Even though substantial effort was dedicated to using rejection sampling in the context of Markovian stochastic processes on networks (cf. Chapter 3), only a few approaches are known to us that focus on non-Markovian dynamics. We present an adaptation of the classical Gillespie method called non-Markovian Gillespie algorithm (NMGA) and its adaptation, the Laplace-Gillespie algorithm (LGA).

Non-Markovian Gillespie Algorithm (NMGA)

Boguná et al. (2014) propose a modification of the Gillespie algorithm for non-Markovian systems, NMGA. Their method is statistically exact but computationally expensive. Conceptually, NMGA is similar to the baseline in Section 4.3.2 but computes the firing times using the survival function (cf. Section 4.3.1). Specifically, they utilize the *joint* survival function of all agents.

Unfortunately, in NMGA, it is necessary to iterate over all agents in each simulation step to construct the joint survival function. The authors also propose a fast approximation where only the current instantaneous rate (at the beginning of each step) is used. They assume that all instantaneous rates remain constant until the next

event. This is reasonable when the number of agents is very high and the time delay of the fastest agent becomes very small.

Laplace-Gillespie Algorithm (LGA)

Masuda and Rocha (2018) introduced the LGA. The method reduces the computational costs of finding the next event time compared to NMGA. They only consider inter-event time densities that can be expressed as a continuous mixture of exponentials:

$$\gamma_i(t) = \int_0^\infty p_i(\lambda)\lambda e^{-\lambda t} d\lambda \,. \tag{4.3}$$

Here, $p_i(\cdot)$ is a PDF over the rate $\lambda \in \mathbb{R}_{\geqslant 0}$, encoding a continuous mixture of exponentials. The restriction of inter-event times limits the scope of the method to survival functions which are *completely monotone* [Masuda and Rocha, 2018]. The advantage is that we can sample the time delay t_i of an agent v_i by first sampling λ_i according to $p_i(\cdot)$ and then sampling from an exponential distribution with rate λ_i.

4.5 Our Method: RED

We propose the RED algorithm for generating statistically correct trajectories of non-Markovian spreading models on networks.

4.5.1 Rate Over-Approximation

Recall that we use the effective rate $\lambda_i(\cdot)$ to express how the instantaneous rate of v_i changes over time, assuming that no neighboring agent changes its state (colloquially, we extrapolate the rate into the future). A key ingredient of our approach is the construction of $\widehat{\lambda}_i(\cdot)$ which upper-bounds the instantaneous rate of v_i, taking into consideration all possible state changes of v_i's neighboring agents. That is, at all times $\widehat{\lambda}_i(t)$ is larger than (or equal to) $\lambda_i(t)$ while we allow that arbitrary state changes of neighbors occur at arbitrary times in the future.

Formally, the upper bound always satisfies:

$$\widehat{\lambda}_i(t_\Delta) \geqslant \sup_{M' \in \mathcal{M}_{v,t_\Delta}} \phi_i\Big(S(n), R(n) + t_\Delta, M'\Big), \tag{4.4}$$

where $\mathcal{M}_{i,t_\Delta}$ denotes the set of reachable neighborhoods of agent v_i after t_Δ time units. Sometimes $\widehat{\lambda}_i(\cdot)$ is referred to as *dominator* of $\lambda_i(\cdot)$ [Gerhard and Gerstner, 2010].

Note that it is not feasible to compute the over-approximation algorithmically, so we have to derive it analytically. Upper-bounds can be constant or dependent on time. For multi-agent models (with a finite number of local states) time-dependent upper bound exists for all practically relevant intensity functions since we can derive the maximal instantaneous rate w.r.t. all reachable neighborhood states. It does not work in some pathological cases (cf. Section 4.5.5).

Example 4.4: Upper Bounds of an Intensity

Consider again the Markovian SIS example from earlier (Example 4.1). The recovery of an infected agent does not depend on its neighborhood. Hence, the rate is always α, which is also a trivial upper bound. The rate at which a susceptible agent, v_i, becomes infected is $\beta \sum_{(s',t')\in M(v_i)} \mathbb{1}(s' = I)$. This means that the instantaneous infection rate can be bounded by $\widehat{\lambda}_i(t_\Delta) = \beta k_i$. Note that this upper bound does not depend on t_Δ. When we use this upper bound to sample the time delay candidate of an agent, this timepoint will always be an under-approximation, even when more neighbors become infected.

However, consider, for instance, a recovery time that is uniformly distributed on $[1, 2]$. In this case, $\lambda_i(\cdot)$ approaches infinity (cf. Figure 4.1b, p. 81) making a constant upper bound impossible (even without considering any changes in the neighborhood).

4.5.2 The RED Algorithm

As input, our algorithm takes a multi-agent model specification $(\mathcal{G}, \mathcal{S}, \{\phi\}_i, \{\psi\}_i, L_0(\cdot))$ and corresponding upper-bounds $\{\widehat{\lambda}\}_i$. As output, the method produces statistically exact trajectories (samples) following the earlier semantics. RED is based on two main data structures:

Labeled graph

A graph represents the contact network. Each agent v_i is annotated with its current state $L(v_i)$ and the timepoint of its last state transition $T(v_i)$ (similar to $R(v_i)$, more on this later).

Event queue

The event queue stores all (potential) future events (i.e. firings). An event is encoded as a tuple $(v_i, \widehat{\mu}, \widehat{t}_i)$, where v_i is the agent that wants to fire, $\widehat{t}_i$ the prospective absolute timepoint of firing, and $\widehat{\mu} \in \mathbb{R}_{\geqslant 0}$ is an over-approximation of the true effective rate (at the future timepoint $\widehat{t}_i$). The queue is sorted according to $\widehat{t}_i$.

A global clock, t_{global}, keeps track of the elapsed time since the simulation started. We initialize the simulation by setting $t_{\text{global}} = 0$ and generating one event per agent. Using $T(v_i)$ (as in Line 2) is a viable alternative to using $R(v_i)$ in order to encode residence times since $R(v_i) = t_{\text{global}} - T(v_i)$. Practically, $T(v_i)$ is more convenient, as it avoids explicit updates of $R(v_i)$ for all agents after each event. We perform simulation steps following Algorithm 8 until some stopping criterion is fulfilled. The main difference from previous approaches is that, traditionally, the rate has to be updated for all neighbors of a firing agent. In RED, only the rate of the firing agent has to be updated.

Algorithm 8: Non-Markovian Simulation with RED

Input: Labeled graph; Event queue; t_{global}.
Output: Updated labeled graph; event queue; t_{global}.

1. Take the first event $(v_i, \widehat{\mu}, \widehat{t}_i)$ from the event queue.
   $\Rightarrow$ Update $t_{global} \coloneqq \widehat{t}_i$.
2. Evaluate the true instantaneous rate $\mu = \phi_i\big(L(v_i), t_{global} - T(v_i), M(v_i)\big)$ of v_i at the current system state.
3. With probability $1 - \frac{\mu}{\widehat{\mu}}$, **reject** the firing and go to 5.
4. Randomly choose the next state s' of v_i according to the distribution $p = \psi_i\big(L(v_i), t_{global} - T(v_i), M(v_i)\big)$.
   $\Rightarrow$ If $L(v_i) \neq s'$: set $L(v_i) \coloneqq s'$, $T(v_i) \coloneqq t_{global}$.
5. Generate a new event for agent v_i and push it to the event queue.

Event Generation. Here, we specify how the event generation in Line 5 is done. We sample a random time delay t_i according to $\widehat{\lambda}_i(\cdot)$ and set $\widehat{t}_i = t_{global} + t_i$ (because the event contains the absolute time). To sample t_i according to the over-approximated rate, we either use the numerical integration of Eq. (4.2) or sample directly from an exponential distribution which upper-bounds the intensity function (cf. Figure 4.1d, p. 81). Finally, we set $\widehat{\mu} = \widehat{\lambda}_i(t_i)$.

4.5.3 Asymptotic Time Complexity

Here, we discuss how the runtime of RED scales with the number of agents. Assume that a binary heap is used to implement the event

queue and that the graph structure is implemented using a hashmap. Each step starts by popping an element from the queue, which has constant time complexity. Next, we compute μ. Therefore, we have to look up all neighbors of v_i in the graph structure and iterate over them. We also have to look up all states and residence times. This step has linear time-complexity in the number of neighbors. More precisely, lookups in the hashmaps have constant time-complexity on average and are linear in the number of agents in the worst case. Computing the rejection probability has constant time complexity. When no rejection events occur, we update $L(v_i)$ and $T(v_i)$. Again, this has constant time-complexity on average. Generating a new event does not depend on the neighborhood of an agent and has, therefore, constant time-complexity. Note that this step can still be somewhat expensive when it requires integration to sample t_i but not in an asymptotic sense. Thus, a step in the simulation is linear in the number of neighbors of the agent under consideration.

In contrast, previous methods require that, after each update, the rate of each neighbor v_j is re-computed. A particular disadvantage is that the rate of v_j depends on the entire neighborhood of v_j. Hence, it is necessary to iterate over all neighbors v_k of every single neighbor v_j of v_i (2-hop neighborhood).

4.5.4 Correctness

As in the previous chapter, the correctness of RED can be shown similarly to [Cota and Ferreira, 2017]. Here, we provide a proof sketch. First, consider the rejection-free version of the method in Algorithm 9.

Algorithm 9: Rejection-Free non-Markovian Simulation

Input: Labeled graph; Event queue; t_{global}.
Output: Updated labeled graph; Event queue; t_{global}.

1. Take the first event $(v_i, \mu, \widehat{t}_i)$ from the event queue.
   $\Rightarrow$ Update $t_{global} \coloneqq \widehat{t}_i$.
2. Randomly choose the next state s' of v_i according to the distribution $\mathbf{p} = \psi_i\big(L(v_i), t_{global} - T(v_i), M(v_i)\big)$.
3. **If** $L(v_i) = s'$: ▷ Null event.
   $\Rightarrow$ Generate a new event for v_i, push it to the event queue.

   Else:
   $\Rightarrow$ Set $L(v) \coloneqq s'$, generate a new event for v_i and add it to the event queue.
   $\Rightarrow$ For each neighbor v_j of v_i: ▷Expensive!
   Remove the event corresponding to v_j and generate a new one (using the new state of v_i).

Rejection events are unnecessary for this version of the algorithm because all events in the queue are generated using the "real" rate and, therefore, are consistent with the current system state. In other words, the rejection probability would always be zero. It is easy to see that Algorithm 9 is a direct event-driven implementation of

Algorithm 7. Therefore, it suffices to show that Algorithm 9 and Algorithm 8 are statistically equivalent.

First, note that it is possible to include *self-loop events* to our model without changing the underlying dynamics (resp. statistical properties). These are events where an agent fires but transitions into the same internal state it already occupies. Until now, we did not allow such self-loop behavior. In the algorithm, self-loop events correspond to the condition $L(v_i) = s'$ in the third step. Such events do not alter the network-state and, therefore, do not change the statistical properties of the generated trajectories. The key idea is now to change $\phi_i(\cdot)$ and $\psi_i(\cdot)$ to $\widehat{\phi}_i(\cdot)$ and $\widehat{\psi}_i(\cdot)$, respectively, such that the events related to $\widehat{\phi}_i(\cdot)$ and $\widehat{\psi}_i(\cdot)$ also admit self-loop events with a certain probability. Specifically, self-loops have the same probability as rejection events in the RED method but, apart from that, $\widehat{\phi}_i(\cdot)$ and $\widehat{\psi}_i(\cdot)$ induce the same dynamical evolution as $\phi_i(\cdot)$ and $\psi_i(\cdot)$. Formally, this is achieved by using the shadow-processes; sometimes also referred to as *complementing process* [Gerhard and Gerstner, 2010] (see also Chapter 3). A shadow-process does not change the state of the corresponding agent but still fires at a specific rate. In the end, we can interpret the rejection events not as rejections but as the statistically necessary application of the shadow-process.

We define the rate of the shadow-process, denoted by $\widetilde{\lambda}(\cdot)$, to be the difference between the rate over-approximation and the true rate. For all v_i, t, this gives rise to the invariance:

$$\widehat{\lambda}_i(t) = \lambda_i(t) + \widetilde{\lambda}_i(t) \ .$$

We define $\widehat{\phi}_i(\cdot)$ such that it includes the shadow-process and use $\widehat{\mu}$ to denote the samples from $\widehat{\phi}_i(\cdot)$.

The only remaining task is to define $\widehat{\psi}_i(\cdot)$ so that the shadow-process does not influence the network-state. Therefore, we trigger a *null event* (or self-loop) with the probability given by the proportion of

the shadow process of $\widehat{\phi}_i(\cdot)$. Consequently, if $\psi_i(s, t, m) = \mathbf{p}$, we define $\widehat{\psi}_i(s, t, m) = \widehat{\mathbf{p}}$ with

$$\widehat{\mathbf{p}}(s') = \begin{cases} \frac{\widehat{\mu}-\mu}{\widehat{\mu}} & \text{if } s' = s \quad \text{(null event)} \\ \left(1 - \frac{\widehat{\mu}-\mu}{\widehat{\mu}}\right)\mathbf{p}(s') & \text{otherwise.} \end{cases}$$

W.l.o.g., we assume that the original system has no inherent self-loops. Using shadow processes, we can now omit the generation of new events for neighboring agents v_j of the firing node v_i. This is because the firing time densities no longer depend on the local state of v_i and are, therefore, not affected by the firing.

In summary, we find that:

1. Adding a shadow process to a multi-agent system does not change its semantics.
2. The rejection-free method (Algorithm 9) generates statistically correct samples.
3. Simulating Algorithm 9 including the shadow-process directly leads to RED (Algorithm 8).

We can conclude the correctness of RED.

4.5.5 Limitations

Our approach's practical and theoretical applicability depends on how well the intensity function of an agent can be over-approximated. The larger the difference between $\lambda(\cdot)$ and $\widehat{\lambda}(\cdot)$ becomes, the more rejection events occur and the slower our method becomes. In general, since rejection events are extremely cheap, it is not a problem for our method when most of the events in the event queue will be rejected.

However, it is easy to think of examples in which RED will perform exceptionally poorly. For instance, consider an SIS-type model, but agents can only become infected if exactly half of their neighbors are infected. In this case, the over-approximation would assume that for all susceptible nodes this is always the case, causing too many rejection events. Likewise, the problem can also occur in the time domain. Consider the case where infected nodes only infect their susceptible neighbors in the first t_Δ time-units of their infection with rate λ, where t_Δ is extremely short (e.g., 0.001) and λ is extremely high (e.g., 1000). Given a susceptible node, we do not know how many of its neighbors will be newly infected in the future, so we have to assume that all neighbors are always infectious.

Moreover, finding a theoretical upper bound for the rate might not be possible. Consider the case where an infected agent with residence time t "attacks" its neighbors at rate $|-\log(t)|$ (which converges to infinity for $t \to 0$). Interestingly, this still gives rise to a well-defined stochastic process because the integration of $|-\log(t)|$ leads to non-zero inter-event times, and it is, therefore, possible to sample inter-event times even though the rate starts at infinity. However, we cannot build an upper bound because, again, we have to assume that all neighbors of a susceptible node are always newly infected.

There are also more practical examples, such as special cases of networked (self-exiting) Hawkes processes [Farajtabar et al., 2015]. Here, a neighbor's firing increases an agent's instantaneous rate. As it is not possible to bound (in advance) the number of times the neighbors fire (at least not without additional assumptions), it is not possible to construct an upper bound for the intensity function for any future point in time.

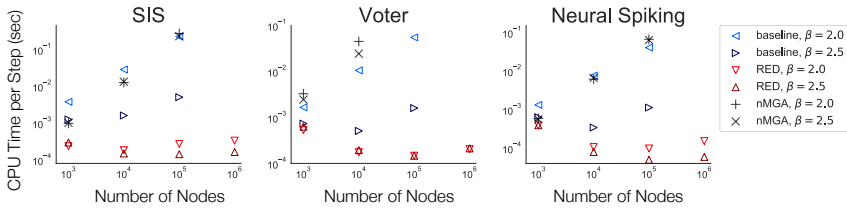


Fig. 4.3.: Results: [Lower is better.] Computation time of a single simulation step w.r.t. network size and connectivity (smaller $\beta \Rightarrow$ higher connectivity). We measure the CPU time per simulation step by dividing the simulation time by the number of successful (i.e., non-rejection) steps.

4.6 Case Studies

We demonstrate the effectiveness of RED in three case studies.

Setup. We use synthetic graphs as contact networks using the configuration model where the degree distribution is specified by a truncated power-law [Fosdick et al., 2018]. That is, for a degree k, $P(k) \propto k^{-\gamma}$ for $3 \leqslant k \leqslant |n|$. We use $\gamma \in \{2, 2.5\}$ (a smaller value for γ leads to a larger average degree and higher connectivity).

The evaluation was performed on a 2017 MacBook Pro with a 3.1 GHz Intel Core i5 CPU. Runtime results for different models are shown in Figure 4.3. To compute the step-wise CPU time, we ignore the rejection steps for a fair comparison. We remark that RED and Baseline are both statistically correct, meaning that they sample from the correct distribution specified by the model semantics, while NMGA provides an approximation.

Baseline. We compare the performance of RED with a rejection-free algorithm (simply called BASELINE) and and an NMGA-type approach. BASELINE is the rejection-free variant of the algorithm where, when an agent fires, all of its neighbors are updated (Algorithm 9). In

the Voter model, the baseline uses an LGA-type approach to sample inter-event times (following Eq. (4.3)). In the other experiments, we sample inter-event times using the rejection-based approach from Figure 4.2 (p. 81). We do note that LGA and RED are not directly comparable, as they are associated with different objectives. In short, LGA focuses on optimizing the generation of inter-event times while RED aims at reducing the number of times that inter-event times need to be generated. We want to emphasize that the reason we include an LGA-type and NMGA-type sampling approach is to highlight that our performance gain is not part of the specifics of how inter-event times are generated.

We use an NMGA-type method as a second comparison. It is a re-implementation of the approximate version of NMGA. The method stores all agents with their associated residence times in a list. In each step, we iterate over the list and generate a new firing time (candidate) for each agent, assuming that the instantaneous rate remains constant (note that assuming a constant rate means sampling an exponentially distributed time delay). Then, the agent with the shortest time delay candidate fires, and the residence times of all agents are updated. The approximation error decreases with an increasing network size because the time periods between events become smaller.

Experiment 1: `SIS` Model

As our first test case, we use a non-Markovian modification of the classical `SIS` model. Specifically, we assume that infected nodes become (exponentially) less infectious over time. That is, the rate at which an infected agent with residence time t "attacks" its susceptible neighbors is ue^{-ut} for $u = 0.4$. This does not directly relate to a

probability density because the infection event might not happen at all. Empirically, we choose parameters that ensure that the infection spreads over the network. We upper bound the rate at which a susceptible agent v_i (with degree k_i) gets infected with $\widehat{\lambda}_i(t) = uk_i$. The upper bound is constant in time and conceptually similar to the earlier example (cf. Section 4.5.1). We sample t_i using an exponential distribution (i.e., without numerical integration). The time until an infected agent recovers is independent from its neighborhood and uniformly distributed in $[0, 1]$ (similar to Röst et al. (2016)). Hence, we can sample it directly. We start with 5% randomly selected infected agents.

Experiment 2: Voter Model

The voter model describes the spread of two competing opinions, denoted as `A` and `B`. Agents in state `A` switch to `B` and vice versa, thus $\psi(\cdot)$ is deterministic.

In this experiment, we use an inter-event time that can be sampled using an LGA-type approach (cf. Eq. (4.3)). Moreover, to take full advantage of the LGA-formulation, we assume that the neighborhood of an agent modulates the PDF $p_i(\cdot)$ which specifies the continuous mixture of rates (otherwise, we could simply pre-compute it). Here, we choose $p_i(\cdot)$ to be a uniform distribution in $[0, o_i]$, where o_i is the fraction of opposing neighbors of agent v_i. That is, if v_i is in state `A` (resp. `B`), then o_i is the number of neighbors in `B` (resp. `A`) divided by the degree k_i.

Hence, we can sample a firing time candidate by sampling a uniformly distributed random variate $\lambda_i \in [0, o_i]$ and then sampling the firing time candidate $t_i \sim \text{Exp}(\lambda_i)$. The resulting inter-event time distribution resembles a power-law with a slight exponential cut-off

[Masuda and Rocha, 2018]. The cut-off becomes more dominant for larger o_i. Formally,

$$\gamma_j(t) = \int_0^{o_i} \frac{1}{o_i} \lambda e^{-\lambda t} d\lambda \quad = \frac{1 - e^{-o_i t}(1 + o_i t)}{o_i t^2} \quad \text{and}$$
$$\lambda_j(t) = \frac{1}{t} - \frac{o_i}{e^{o_i t} - 1} \quad t \geqslant 0 \,.$$

To upper-bound the instantaneous rate, we set $o_i = 1$. To sample t_i in RED, we use rejection sampling (Figure 4.2, p. 81). BASELINE uses the LGA-based approach, but changing to rejection sampling does not noticeably change the performance. We initialize the simulation with 50% of agents in `A` and `B`, respectively.

Experiment 3: Neural Spiking

To model neural spiking behavior, we propose a networked (i.e., multivariate) *temporal point-processes* [W. Wu et al., 2019]. In temporal point-processes, agents are naturally excitable (`S`) and can get activated (`I`) for an infinitesimally short period of time. After that, they become immediately excitable again. Point-processes model scenarios where one is only interested in the firing times of an agent, not in their local state. They are commonly used to model the spiking behavior of neurons [Truccolo, 2010] and for information propagation in social networks (like re-tweeting) [Farajtabar et al., 2015]. A random trajectory of a system identifies each agent v_i with a list of timepoints $\mathcal{H}_i$ of its activations. Here, we consider multivariate point-process, where each agent represents one point-processes and neighboring agents influence each other by inhibition or excitement. Therefore, we identify each (undirected) edge (v_i, v_j) with a weight $w_{i,j}$ of either 1 (excitatory connection) or -1 (inhibitory connection).

We modify the neighborhood $M(v_i)$ to also contain the target node of an edge. Note that adding edge weights to RED is straightforward. Moreover, neurons can spontaneously fire with a baseline intensity of $\alpha \in \mathbb{R}_{\geqslant 0}$. Formally,

$$\phi_i(s, t, m) = f\Big(\alpha + \sum_{s', t', v_j \in M(v_i)} \frac{w_{i,j}}{1 + t'}\Big)$$
$$\text{with} \quad f(x) = \max\big(0, \tanh(x)\big) \ .$$

The function $f(\cdot)$ is called a *response function*, it converts the synaptic input into the actual firing rate. We use the same as Benayoun et al. (2010). Without $f(\cdot)$, the intensity could become negative. Note that $\psi_i(\cdot)$ is deterministic. Our model can be seen as a non-Markovian modification of the spiking neuron model proposed by Benayoun et al. (2010). Contrary to Benayoun et al., we do not assume that active neurons stay in their active state for a specific (in their case, exponentially distributed) amount of time. Instead, we assume that they become immediately excitable again and that an activation affects the neighboring neurons through a kernel function $1/(1 + t')$. The kernel ensures that neighbors who fired more recently (i.e., have a smaller residence time t') have a more significant influence on an agent.

The residence time of an agent itself does not influence its rate. In contrast to multivariate self-exiting Hawkes processes, only the most recent firing—and not the whole event history $\mathcal{H}_v$—contributes to the intensity of neighboring agents [Farajtabar et al., 2015; Dassios, Zhao, et al., 2013]. Taking the whole history into account is not easily possible with a finite amount of local states and introduces intensity functions that cannot be upper-bounded (cf. Section 4.5.5). For our experiments, we set $\alpha = 0.01$, define 20% of the edges to be inhibitory, and use the trivial upper bound of one (induced by the response function).

4.6.1 Discussion

Our experimental results clearly indicate that rejection-based simulation (and the corresponding over-approximation of the instantaneous rate) can dramatically reduce the computational costs of stochastic simulation in the context of non-Markovian simulation on networks.

As expected, we see that the runtime behavior is influenced by the number of agents (nodes) and the number of interconnections (edges). Interestingly, for RED, the number of edges seems to be much more relevant than the number of agents. Most noticeably, the CPU time of each simulation step practically does not increase (beyond statistical noise) with the number of nodes. Moreover, one can clearly see that RED consistently outperforms BASELINE up to several orders of magnitude (Figure 4.3 (p. 95)), while the gain in computational time (i.e., RED's CPU time by BASELINE's CPU time) ranges from 10.2 (10^3 nodes, voter model, $\beta = 2.5$) to 674 (10^5 nodes, SIS model, $\beta = 2.0$).

Note that we only compared an LGA-type sampling approach with our method in the voter model experiment. The other case-studies could not straightforwardly be simulated with LGA due to its constraints on the time delays. However, we still assume that the rejection-free baseline algorithm is comparable with LGA in the other experiments, as both update the rates of the relevant agents only after an event. We also tested an NMGA-like implementation where rates are considered to remain constant until the next event. However, the method scales—albeit it is only approximate—worse than the baseline.

Note that the SIS model is somewhat unfavorable for RED as it leads to the generation of a large number of rejection events, especially when only a small fraction of agents are overall infected. For concreteness, consider an agent with many neighbors of which only very few are infected. The over-approximation assumes that *all* neighboring

agents are infected *all the time*. Nevertheless, the low computational costs of each rejection event seem to easily atone for their large number. In contrast, the neural spiking model is very favorable for our method as the $\tanh(\cdot)$ response function provides a global upper bound for the instantaneous rate of each agent. Performance-wise, the differences between the two models are, surprisingly, pretty slight.

4.7 Conclusions

We proposed RED, a rejection-based algorithm for the simulation of non-Markovian agent models on networks. The key advantage and most significant contribution of our method is that it is no longer required to update the instantaneous rates of the whole neighborhood in each simulation step. This practically and theoretically reduces the time complexity of each step compared to previous simulation approaches and makes our method viable for the simulation of dynamical processes on real-world networks, which often have millions of nodes. In addition, rejection steps are a fast alternative to integrating the intensity function (for suitable inter-event time distributions). Currently, the most notable downside of the method is that the over-approximations $\widehat{\lambda}(\cdot)$ have to be constructed manually. It remains to be determined if it is possible to automate the construction of $\widehat{\lambda}$ in an efficient way as the trivial way of searching in the state space of all reachable neighborhoods is not feasible. We also plan to investigate how correlated events (as in [Jo, Lee, et al., 2019; Masuda and Rocha, 2018]) can be integrated into RED.

5 Vaccine Allocation Optimization

This chapter addresses the problem of mitigating the spread of an epidemic over a contact network by vaccinating a limited number of nodes.

Our **key idea** is to study simulated trajectories to estimate a node's impact on an epidemic. For better efficiency, we analyze the "empirical" trajectories and the static contact network simultaneously. This results in cheap scores that act as a surrogate for the true impact of each node.

We propose SEPIA [1] ([S]imulation-based [epi]demic mitig[a]tion), a combination of (i) numerous simulation runs, (ii) a PageRank-type influence analysis on an empirical *transmission graph* which is learned from the simulations, and (iii) discrete stochastic optimization.

SEPIA scales very well with the size of the network and proposes a vaccination strategy that considers specific clinical and transmission parameters of the epidemic.

[1] SEPIA is implemented in Rust and Python and available at `github.com/gerritgr/Simba`

5.1 Introduction

Vaccine allocation strategies can help design complex systems to make them more resilient against (cascading) failures. This is particularly relevant regarding infrastructure networks where a "vaccination" might represent the installation of a protective safeguard. Another example is the mitigation of fake news in online social networks, which can be achieved by removing the accounts of particularly relevant and malicious influencers or by providing warnings and fact-checking. In the context of infectious diseases, vaccination strategies are necessary for prioritization.

Generally speaking, the vaccine allocation problem is computationally challenging. Intuitively, it is often reasonable to vaccinate those nodes with many neighbors (or with a high *centrality* in the network) and those close to the initially infected nodes. If possible, it is even better to identify the nodes that lie between the initially infected nodes and many susceptible nodes. If we represent the spreading process by a *transmission tree* (cf. Figure 5.1 (p. 106)), in which the direct children of a node v_i correspond to those nodes that were infected by v_i, the size of a v_i's subtree gives the number of multi-hop infections that originated from v_i. The premise of our work is that a node's number of multi-hop infections is a good indicator of whether that node is a suitable vaccination candidate.

To this end, we propose SEPIA, a method that relies on the efficient rejection-based simulation form Chapter 3 (*Simulation of Markovian Spreading*). Based on many simulations, SEPIA constructs a generalization of the transmission tree called *transmission graph*. By analyzing this graph, we obtain an impact score for every node. Repeated evaluation of the current vaccination strategy and re-computation of the impact scores yields an iterative optimization procedure. The corresponding objective is to maximize the expected number of nodes that remain healthy.

The key methodological novelty of our proposed vaccination strategy is constructing and analyzing an empirical transmission graph. Using this graph, our vaccination strategy can take the dynamics of the epidemic into account. The transmission graph potentially has many more use cases in assessing network dynamics, such as influence maximization, controllability of networks, impact or centrality quantification, and flow prediction. We also provide a numerical evaluation and compare SEPIA to several baselines from the literature.

5.2 Related Work

Most methods that suggest nodes for vaccination use static analysis of the contact network, for instance, by looking at the *betweenness centrality* of nodes [Schneider et al., 2011] or their degree [Prakash, Tong, et al., 2010]. Likewise, NETSHIELD tries to minimize the epidemic threshold of the contact graph (i.e., its general ability to support epidemics) [Tong et al., 2010]. A more advanced method is GRAPHSHIELD which starts with degree centrality, but then takes the flow of information in the contact graph into account [Wijayanto and Murata, 2017]. Eventually, researchers focused more on the dynamical aspects, for instance, by utilizing linear programming [Sambaturu and Vullikanti, 2019] or reinforcement learning [Wijayanto and Murata, 2018; Wijayanto and Murata, 2019]. For an overview, we refer the reader to Nowzari et al. (2016).

Conceptually most relevant for us is the work of Yao Zhang and Prakash (2015) who propose DAVA and Song et al. (2015) who propose NIIP. Both methods are based on a *dominator tree* architecture that tries to capture the direction of the epidemic. DAVA merges all initially infected nodes and analyzes the paths from this node to all other nodes. Nodes that block a large number of paths are suitable

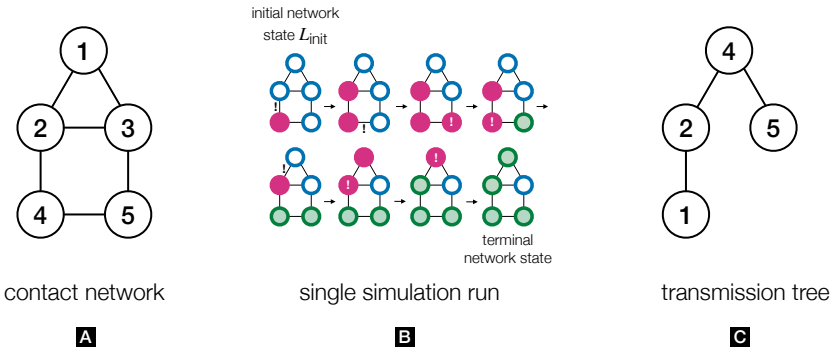


Fig. 5.1.: **(a)**: Example contact network with possible SIR dynamics (S: blue line; I: red, filled; R: green, shaded). **(b)**: The firing node/edge is annotated with an exclamation mark. **(c)**: The trajectory results in a transmission tree.

vaccination candidates. NIIP focuses on a problem setting where not all vaccination units are distributed simultaneously. Therefore, NIIP extracts a maximum DAG from the contact graph, uses Monte-Carlo simulation to find the best nodes to vaccinate and combines this with a greedy simulation-based approach; the simulation's goal is to determine *when* to distribute a vaccination dose.

5.3 Problem Statement

As a dynamical model, we consider SIR dynamics (Model 7). W.l.o.g. we typically assume $\alpha = 1$ and only vary β as the spreading dynamics is determined by the fraction $\frac{\beta}{\alpha}$ and that there is at least one infected node (called patient zero) in the initial network-state $L_{init}(\cdot)$.

Model 7: SIR

The SIR model is a version of the SIRS model (Model 2) where immunity loss is not happening.

States:	$\mathcal{S} = \{\mathtt{S}, \mathtt{I}, \mathtt{R}\}$	
Parameters:	Recovery rate:	$\alpha \in \mathbb{R}_{>0}$
	Infection rate:	$\beta \in \mathbb{R}_{>0}$
Rules:	Recovery:	$\mathtt{I} \xrightarrow{\alpha} \mathtt{R}$
	Infection:	$\mathtt{S} \xrightarrow{\beta \mathbf{m}[\mathtt{S}]} \mathtt{I}$

Simulating SIR dynamics will eventually lead to a *terminal* labeling or network state where all nodes are either recovered or susceptible. Given a random simulation run, we use the term *transmission tree* (cf. Figure 5.1, p. 106) to describe a tree where patient zero is the root (if there are more than one infected nodes in the beginning, we merge them) and every node that became infected during the epidemic is connected to the node which infected it. Thus, all nodes in the subtree of a node are called its *children*. That is, that node directly or indirectly infected them.

5.3.1 Vaccination Allocation Problem

We are given a finite contact network $\mathcal{G}$ with corresponding initial labeling $L_{\text{init}}(\cdot)$, a vaccination budget $k \in \mathbb{Z}_{>0}$, as well as the recovery and infection rate constants α and β. Let S_{init} denote the set of nodes that are initially susceptible.

We want to find a set X of nodes to be vaccinated, where

$$X \subset S_{\text{init}} \quad \text{and} \quad |X| = k\,. \tag{5.1}$$

Moreover, for a given $\mathcal{G}$, $L_{init}(\cdot)$, k, α, β, we use $F(X)$ to denote the objective function which we define as the expected number of susceptible nodes in the terminal network-state (when initially all nodes in X are vaccinated). We define the *Vaccine Allocation Problem* as:

Find a set X that maximizes $F(X)$ such that Eq. (5.1) holds.

In practice, we approximate $F(\cdot)$ based on many Monte-Carlo simulation runs. We assume that at least k nodes exist that can be vaccinated and that there is at least one infected node in the initial network-state. We model the vaccination by setting $L_{init}(v_i) = R$ for all $v_i \in X$ at the beginning of the simulation. Note that (assuming the vaccination works perfectly) recovered, deceased, and vaccinated nodes do not differ from the simulation's point of view.

Complexity. The problem is computationally difficult because there are $\binom{n}{k}$ possibilities to distribute k vaccines to n nodes. The corresponding decision problem is $\mathcal{NP}$-hard. Specifically, for a given input $\mathcal{G}$, $L_{init}(\cdot)$, β, α, and threshold τ, it is $\mathcal{NP}$-hard (in n) to decide if a solution X exists, s.t. $F(X) > \tau$. It can be shown that for this type of problem, $\mathcal{NP}$-hardness holds for any propagation model that can mimic an independent cascade (IC) model [Yao Zhang and Prakash, 2015]. We can do this by making α (resp. β) arbitrary small (resp. large).

5.4 Our Method: SEPIA

We first explain the main ingredients of SEPIA: the rejection-based simulation method and the transmission graph construction for identifying high-impact nodes.

5.4.1 Ingredient 1: Rejection-Based Simulation

Our rejection-based SIR simulation is based on CORAL from Chapter 3. That is, we perform event-driven simulations using a priority queue. For the initialization, we create one recovery event and one infection event for each infected node and push them into the queue. The firing time of node v_i is exponentially distributed with rate α (recovery event) or $\beta \cdot k_i$ (infection event). In each simulation step, we take the first event from the queue. If it is a recovery event, we simply set the corresponding node to state R. If it is an infection event, we first check if the corresponding node is still in state I; if not, we reject the event and proceed with the next step. If it is, we pick a random neighbor that will be the target of the infection. We check if the random neighbor is susceptible. If this is the case, we set the neighbor to I and create two events (recovery and infection) for the newly infected neighbor. We also create a new infection event for the source node. Then, we proceed with the next step. The simulation ends when there are no more nodes in state I.

We store the number of susceptible nodes when the simulation ends. Moreover, each time a node gets infected, we store from which (infected) neighbor the infection originated (or all nodes it could have originated from; cf. Section 5.4.2).

5.4.2 Ingredient 2: Impact Score Estimation

To estimate each node's impact, we build an empirical transmission graph (Figure 5.2 (p. 110)), an extension of the transmission tree from Figure 5.1 (p. 106) to multiple simulation runs. The transmission graph is directed, and one can perform a random walk on the graph which (on average) visits nodes with higher impact more often. In the end, we determine the impact of each node in S_{init} by ranking the nodes similar to the well-known PAGERANK-score developed by

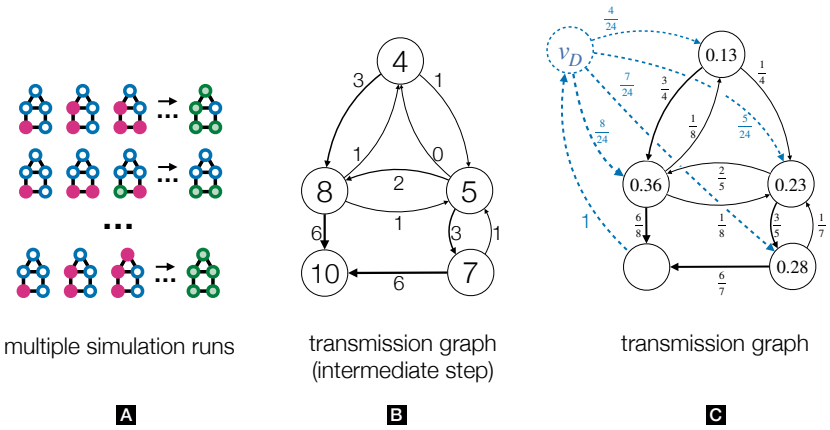


Fig. 5.2.: Transmission graph construction based on the same setting as Figure 5.1. **(a)**: We consider 10 simulation runs. **(b)**: Contact graph with I_i as node labels and $I_{(i,j)}$ as edge labels. **(c)**: Adding the dummy node and normalizing outgoing weights yields a discrete-time Markov chain (DTMC). Nodes in S_{init} are annotated with their impact score based on the equilibrium of the DTMC.

Page et al. (1999) (i.e., we use the equilibrium of the corresponding Markov chain).

Given a set of simulated trajectories, let I_i denote the number of trajectories in which node v_i became infected. Furthermore, let $I_{(j,i)}$ denote the number of trajectories in which v_j directly infected v_i. Note that $I_i = \sum_j I_{(j,i)}$.

Transmission Graph

Consider the contact network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. We construct the corresponding transmission graph $\mathcal{G}_T = (\mathcal{V}_T, \mathcal{W}_T)$ ($\mathcal{W}_T \in \mathbb{R}_{\geqslant 0}^{(n+1)\times(n+1)}$ being a weight matrix) as follows: We start with a dummy node v_D as a sink, that is, $\mathcal{V}_T = \mathcal{V} \cup \{v_D\}$ (we can remove unreachable nodes

later), and add an edge from each initially infected node with weight one, i.e., for all $v_i \in \mathcal{V}$:

$$\mathcal{W}_T[i, D] = \begin{cases} 1 & \text{if } v_i \in I_{\text{init}}, \\ 0 & \text{otherwise,} \end{cases}$$

where I_{init} denotes the set of nodes that are initially infected. Then we add an edge from v_D to each $v_i \in S_{\text{init}}$ with a weight proportional to the estimated probability of that node becoming infected, i.e., for all i:

$$\mathcal{W}_T[D, i] = \begin{cases} \frac{I_i}{\sum_{v_j \in S_{\text{init}}} I_j} & \text{if } v_i \in S_{\text{init}}, \\ 0 & \text{otherwise.} \end{cases}$$

Moreover, for all nodes $v_i \neq v_D$ (we consider $0/0$ as 0):

$$\mathcal{W}_T[i, j] = \frac{I_{(j,i)}}{I_i} .$$

Note that, by construction, the outgoing weights in G_T are normalized and, therefore, represent a discrete-time Markov chain. A random walk in the chain will preferably visit nodes of high impact on the epidemic as the transition probabilities are proportional to the estimated infection probabilities. We compute the equilibrium distribution of the corresponding Markov chain using the power iteration method [G. Stewart and J. Miller, 1975]. We call the equilibrium probability of a node normalized over S_{init} its *impact score*. We only consider the impact score for nodes in S_{init} because only those are eligible for vaccination. Note that a transmission graph for a single simulation run is equivalent to the transmission tree where all edges have weight one.

We can make the transmission graph even more accurate with a variance reduction trick. During the simulation, instead of only storing the node that actually transmitted the infection, we store all neighbors that could potentially have been the source of the infec-

tion. Typically, each infected neighbor is the source of transmission with equal probability. It is straightforward to adapt the construction of the transmission graph accordingly, even in non-Markovian settings.

5.4.3 Introducing SEPIA

We combine rejection-based simulations and the transmission graph analysis with an iterative optimization scheme to arrive at SEPIA.

Greedy Initialization. We use C_i to denote the set of vaccinated nodes in iteration i. We start with an empty set, C_0, of nodes to be vaccinated. Until $|C_i| = k$, we compute the impact score for all nodes $v_i \in S_{init}$ (assuming nodes in C_i are vaccinated) and add the node with the highest impact to C_i, leading to C_{i+1}.

Optimization. In each optimization step i, we randomly remove one node (with equal probability) from C_i (leading to set B_i) and compute the impact score of all nodes $v_i \in S_{init} \setminus C_i$ (assuming nodes in B_i are vaccinated). Then we add one of the nodes with the highest impact to B_i (nodes with higher impact are more likely to be chosen), leading to C_{i+1}. We estimate $F(C_i)$ in each iteration step and repeat until some stopping criterion is reached. Then, we return the set that yielded the highest estimated score.

5.4.4 Discussion

Here, we want to address three non-obvious questions: (i) *Why build a transmission graph?*, (ii) *How is the graph related to the objective?*, and (iii) *Why is it necessary to consider the dynamics at all?*

Q.1: Building a Transmission Graph. Using the transmission graph has multiple advantages. Most importantly, transmission trees only mimic a subset of possible infection flows. In contrast, transmission graphs allow to aggregate information over many runs in a principled manner (cf. Figure 5.3, p. 114). This way, they capture the interplay between connectivity and infection flow more precisely. Moreover, computing the equilibrium of the Markov chain is computationally fast and theoretically well principled. It is also possible to efficiently build the transmission graph on-the-fly during the simulations.

Example 5.1: Transmission Graph

Consider Figure 5.3 (p. 114). Assume we want to estimate the impact of the two successor nodes of patient zero based on two simulation runs. Using, for example, the size of their corresponding subtree in the transmission tree leads to misleading results in this case. Specifically, both nodes would be assigned drastically different values. Combining the two runs in a transmission graph yields a more realistic impact score than considering both runs separately. Note that edges point to the origin of the infection, and the transmission graph is shown without its dummy node.

Q.2: Impact Score and Objective. Note that we handle two different problems. The impact score quantifies the question "*How many nodes became infected as a direct ('multi-hop') consequence from each node?*" However, the objective $F(\cdot)$ is concerned with "*How many nodes will*

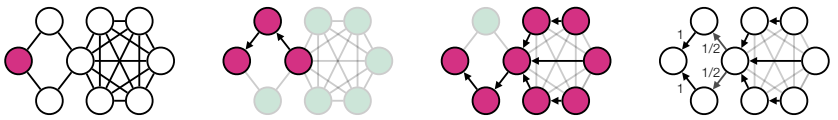
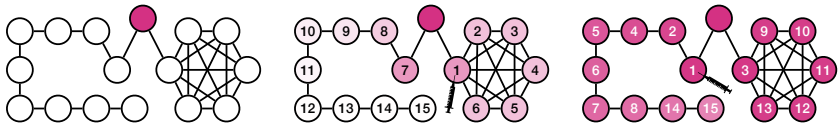


Fig. 5.3.: Using the transmission graph (shown without dummy node) leads to more realistic impact scores compared to considering both simulations separately.

(on average) not become infected if a specific set of nodes is vaccinated?" The latter question is notoriously more difficult to answer. They differ because if we vaccinate a node, all of its children in the transmission tree can still become infected via alternative paths. In this sense, the impact score gives an over-approximation of the effect of vaccinating a node. Colloquially, if we vaccinate a node with m children (on average), then the best we can hope for is that these m nodes do not become infected. Our optimization procedure picks nodes depending on their theoretical (and over-approximated) capability to reduce the spreading.

Q.3: Importance of Dynamics. The goal is to vaccinate nodes such that the network becomes less "supportive" of epidemics spreading in it. But why should the specific dynamics matter? In other words, how can vaccinating a specific node be the right decision for some infection rate constants and the wrong decision for other ones? We give an example for this in Figure 5.4 where we have a single patient zero and a budget of $k = 1$. We can either vaccinate the node to the "right" to protect the fully connected component (FCC) with six nodes or we can vaccinate the node to the "left" to protect the line-graph with nine nodes. If the epidemic is "weak", it will die out anyway over the line graph, so it makes sense to protect the FCC. In contrast, protecting the line-graph "saves" more nodes if the epidemic is strong enough to conquer the whole graph.



contact network with patient zero

low infectiousness $\alpha = 1, \beta = 1$

high infectiousness $\alpha = 1, \beta = 20$

Fig. 5.4.: Assume $k = 1$. The best node to vaccinate depends on the dynamics. If β is small, the infection will die out on its own in the line graph, and it makes more sense to protect the FCC even though it contains fewer nodes. The opacity illustrates a node's probability of becoming infected. The nodes are numbered in decreasing order of their impact scores.

5.4.5 Generalizations

Our framework can easily be extended to various epidemic-type models. The only necessity is that (i) the model can be simulated (efficiently), (ii) there is a clear objective (e.g., maximize susceptible nodes in terminal states), and (iii) the transmission graph can capture a direction of the information flow. Potential generalizations include models with more disease stages (like SEIR), non-Markovian dynamics (e.g., where the infectiousness of nodes changes over time), weighted and directed networks, as well as temporal or adaptive networks and time-discrete models. SEPIA can also be adapted to different objectives. For instance, in the SIS model (where infected nodes become susceptible again) the goal is typically to minimize the number of infected nodes in the equilibrium. In that case, our method would identify the nodes that are generally most impactful for the epidemic spreading and not only with regards to a specific initial set of infected nodes. Likewise, we could optimize the time-points of vaccine distribution [Song et al., 2015]. When the rate parameters are unknown, we suggest using an infection rate slightly higher than the *epidemic threshold*.

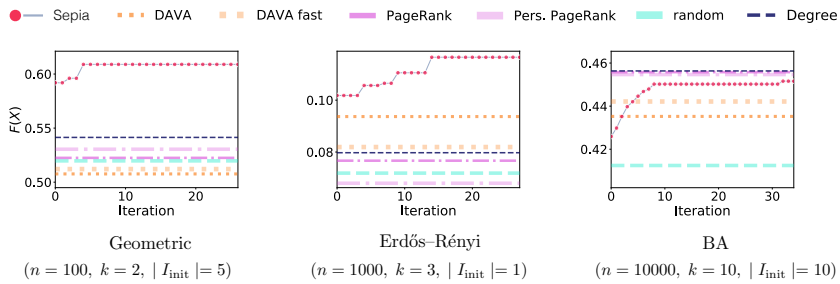


Fig. 5.5.: [Higher is better.] Optimization of $F(X)$ (terminal fraction of expected susceptible nodes) on three sample networks.

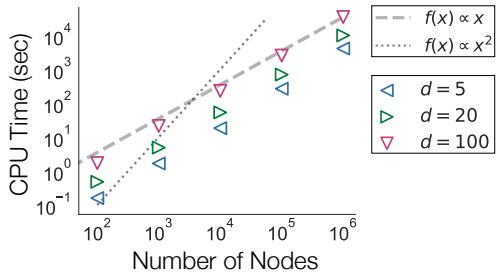


Fig. 5.6.: [Lower is better.] Runtime of 10^3 simulations and solution of the corresponding transmission graph based on random d-regular graphs.

5.5 Experimental Results

We compare different vaccine allocation methods on synthetic contact networks.

Baseline. The following baselines are used: RANDOM (expected $F(X)$ when random nodes are vaccinated), DAVA, and DAVA-FAST [Yao Zhang and Prakash, 2015], PAGERANK, PERS. PAGERANK (personalized PageRank) [Jeh and Widom, 2003; Yao Zhang and Prakash, 2015], and DEGREE (pick nodes with highest degree).

Setup. We used synthetic networks following three random graph models (Erdős-Rényi, Geometric, and Barabási-Albert (BA)) with 10^2, 10^3, and 10^4 nodes, respectively. The corresponding budgets are $k = 2$, $k = 3$, and $k = 10$. We use 10^3 simulation runs for each construction of the transmission tree.

Results. We analyze the runtime of a complete construction and solution of a transmission graph based on d-regular random graphs (i.e., all nodes have precisely d neighbors) with varying degree d and n. Practically, the runtime is almost linear in n. Theoretically, the number of simulation steps in each run increases linearly. The costs of each simulation step increase sub-linearly. The costs of solving the DTMC also increase linearly. We see that, even though the number of iteration steps is relatively small, SEPIA is superior to or (almost) on par with the baselines in the experiments (Figure 5.5, p. 116). SEPIA struggles the most with BA graph, which is a special case that highlights potential problems. It seems that the general strategy of SEPIA to separate the initially infected from the susceptible nodes does not work better than identifying the nodes, which are generally crucial for the graph's resilience against epidemics. This is because BA graphs typically possess a small subset of nodes that are highly effective candidates for vaccination regardless of the infection source. Note that DAVA also struggles in this case while DEGREE and both PAGERANK methods shine. Runtime results are reported in Figure 5.5 (p. 116).

5.6 Implications and Future Work

This chapter presented a novel technique to find a network's most suitable vaccination candidates. Unlike other methods, our approach is based on statistically correct simulations, which are analyzed using the transmission graph. The transmission graph represents the flow of a pathogen in the network as a directed weighted graph and provides a principled over-approximation of the node's impact on an epidemic. The method is suitable for all epidemic models that can be efficiently simulated. An interesting extension would be to perform different types of information flow analysis (and ranking methods) on the transmission graph, not only random walks. It remains to be determined which flow analysis is most useful for which type of objective (e.g., vaccination, control, influence maximization).

Implications. That this chapter is a highly theoretical work that cannot—and should not—be taken as a directive to act in the event of a real-world pandemic. The assumption that we have perfect knowledge of the underlying contact network and the transmission parameters of the pathogen is unrealistic for real-world applications (at least for epidemic spreading). Accounting for dynamics through simulation is particularly useful in the early stages of a pandemic. However, often no vaccine is available at this point (the 2022 monkeypox outbreak might be an exception).

In the Covid-19 pandemic, most countries prioritized health care workers in the vaccination order. This can be seen as a basic form of impact score estimation. Health care workers are close to infected individuals/nodes and have, therefore, a high chance of becoming infected themselves. Moreover, their high degree (many contacts) increases their impact. In contrast, prioritizing elderly people assumes heterogeneity in the population (where the infection of vulnerable individuals is particularly costly) that we did not model in this work.

Some researchers also advocated for a more DEGREE-like vaccination order [C. Cox, 2020]. While DEGREE provides an intuitive and simple baseline, it should be pointed out that the performance of DEGREE on synthetic static networks might not be directly translatable to dynamic real-world networks. A DEGREE-based policy would also raise ethical concerns. For instance, it could lead to a situation where people would be "punished" for following social distancing and having few contacts. Meanwhile, others would be rewarded with early vaccination for having many contacts and posing a greater threat to society. Further discussions on the ethical implications of vaccination prioritization cannot be made here but are essential to consider for real-world decision making.

6 Covid-19 and the Limitations of Modeling

In the course of the Covid-19 pandemic, mathematical modeling quickly became a topic of public interest and a controversial asset in the increasingly polarized social discourse. This chapter identifies misconceptions and limitations of common computational epidemic modeling techniques. Particularly, we highlight the importance of population heterogeneity and explain why many models used in the current Covid-19 pandemic cannot adequately capture it. Code is made available[1].

Our **key idea** is to synchronize the rate parameters of different epidemiological models to compare their dynamics in a principled way. In addition, we use node-level infection parameters, allowing us to control and study (variations in) connectivity and infectiousness separately.

6.1 Introduction

At the beginning of 2020, the world was hit by the coronavirus (SARS-CoV-2) pandemic. Faced with the approaching overload of healthcare systems, the international community turned to non-pharmaceutical interventions (NPIs) in an attempt to contain the spread of the pathogen [Brauner et al., 2020]. Computational epidemiological modeling became a vital asset to predict the propagation and to

[1] `github.com/gerritgr/Covid19Dispersion`

evaluate the prospective effectiveness of various measures such as school closures and travel restrictions [David Adam, 2020; Bui et al., 2020]. For an overview of Covid-19 models and their successes and failures, we refer the reader to Kuhl (2020) and Holmdahl and Buckee (2020).

This chapter discusses the consequence of population heterogeneity for computational epidemiology. We study two types of heterogeneity and their influence on the emerging pandemic:

Individual Variations in Contact Numbers

As in all previous chapters, we assume a contact network that specifies the interaction structure. Individual connectivity variations naturally follow from this.

Individual Variations in Infectiousness

Infectiousness describes how infectious an individual becomes when infected (e.g., how much viral load the subject emits). Unlike the previous chapters, this chapter assumes variations among nodes/individuals.

Note that we use the term *infectiousness* as a property of the host. It denotes the probability of passing a pathogen over an edge.

We qualitatively study the dynamical evolution based on different properties, such as the height of the infection peak and the fluctuations of the *effective reproduction number* R_t (average number of secondary infections at timepoint t). Furthermore, we study how this heterogeneity influences the *dispersion* during an epidemic's evolution. Covid-19 is associated with an exceptional high dispersion and understanding how it emerges is a crucial asset in controlling the pandemic [Althouse et al., 2020].

Terminology and Concepts

A noticeable example of heterogeneity in a population's interaction structure are individuals with extraordinarily many contacts, so-called *hubs*. Similarly, *super-spreader* events refer to temporary gatherings where a single infected individual (potentially) infects many others. The evidence for the importance of hubs and super-spreader events became increasingly conclusive over time [Cave, 2020; Riou and Althaus, 2020; Dillon Adam et al., 2020; Hasan et al., 2020]. It was pointed out early in the pandemic that many models do not accurately capture them [Shen et al., 2020].

The concept of *(over-)dispersion* is closely related [Lloyd-Smith et al., 2005; Lloyd-Smith, 2007; Müller and Hösel, 2020] and is consistently reported for the Covid-19 pandemic [Cevik et al., 2020; Endo et al., 2020; Tariq et al., 2020; Hébert-Dufresne et al., 2020; K. Sun et al., 2020]. In short, this concept reflects that a small number of infected individuals infect many others, while most infected individuals infect no one or only very few. Overdispersion can be caused by hubs (many contacts, average transmission probability) but also by individuals with high infectiousness (average contact number, high transmission probability).

Viral load levels (and other properties that determine a host's infectiousness) differ between individuals and within individuals over time [T. C. Jones et al., 2020; Walker et al., 2020; Goyal et al., 2020; Walsh et al., 2020]. While many models include the temporal aspect, the effects of individual variations are not well explored.

6.1.1 Method Overview

To study the effects of heterogeneity, we translate a typical ODE model for the spread of Covid-19 to a stochastic network-based

model. Then, we modulate the connectivity (by looking at different contact networks) and the infectiousness (by sampling the infection rate parameters from a distribution). We study how this modulation alters the dynamical evolution. The critical part of the comparison is that we keep population averages fixed. For instance, we only compare networks with the same overall connectivity (mean degree). We also only compare epidemic models with the same mean infection rate parameters and study different degrees of deviation. Hence, we can study the effects of variations, not the effects of different population averages.

Our contributions are as follows:

1. We give an overview of popular Covid-19 models based on ODEs, branching processes, and networks and discuss their (implicit) assumptions about a population's heterogeneity.

2. We show that imposing common interaction structures (i.e., using a graph to determine how infections can propagate) drastically changes an epidemic's evolution.

3. We analyze the additional effects of individual viral load variations.

4. We propose a novel method for quantifying time-dependent dispersion based on an empirical analysis of simulation runs.

6.1.2 Descriptive Statistics

Here, we briefly summarize the three new aspects used in this chapter to characterize epidemic processes.

Basic Reproduction Number R_0

R_0 describes how many susceptible individuals are (on average)

infected by a single infectious individual in a completely susceptible population (i.e., expected number of *secondary infections* of patient zero).

Effective Reproduction Number R_t

Describes the average number of secondary infections of a random infected individual at timepoint t.

Dispersion

Dispersion is a way to quantify the variance in the number of secondary infections. *Overdispersed* epidemics admit a high variance in the number of secondary infections (w.r.t. some baseline).

Typically, R_t decreases over time due to the increase in immunity in the population. In theoretical models, we can analytically derive R_0 from the model parameters. To study R_t, we typically resort to the analysis of simulation runs.

In branching processes, dispersion can be described in terms of a dispersion parameter k when using a negative binomial distribution to generate offspring. The probability mass functions can be re-parametrized in terms of the dispersion k and the mean offspring number [Lloyd-Smith, 2007]). (Over-)dispersion is then measured compared to the Poisson offspring distribution. This chapter proposes alternative methods (cf. Section 6.3.4).

6.2 A Tale of Three Models

We follow the network-based spreading paradigm and study its relation to other model types. In particular, we study which types of population heterogeneity can be expressed and how models are used in the current Covid-19 pandemic. We focus on the three model

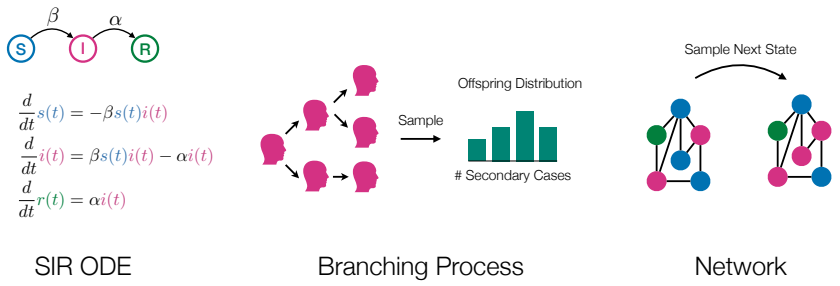


Fig. 6.1.: Schematic overview of epidemic model types.

types that we consider most relevant for epidemiological modeling in general. While ODE models and network-based models are directly relevant for the evaluation, we include branching processes in this section because they are the de facto standard for formalizing and studying dispersion in epidemics. For a comparative analysis of models specific to Covid-19, we refer the reader to Adiga et al. (2020) and Kuhl (2020). Note that all models that study Covid-19 quantitatively suffer from poor data quality and uncertainty about parameters [Ioannidis, 2020; Sanguinetti, 2020].

ODE Models

The most widespread epidemiological model type is based on a system of ordinary differential equations (ODEs) in which coupled fractions of individuals in disease compartments change deterministically and continuously over time [Roy M Anderson et al., 1992]. For an overview on various applications, we refer the reader to [Nelson and Williams, 2014; Frauenthal, 2012; S. Ma and Xia, 2009; Brauer et al., 2008]. Compartments refer to different disease stages (e.g., susceptible (`S`), infected (`I`), recovered (`R`), exposed (`E`), dead (`D`)). Most commonly used is the three-compartment `SIR`-model (cf. Figure 6.1). Note that ODE models use a single parameter (β) to model the

chance of meeting someone (interaction structure) and the probability of transmitting the infection.

Population Heterogeneity. Expressing population heterogeneity is only possible to a minimal degree. The typical way is to introduce additional compartments that encode a membership in a certain group (e.g., *susceptible* and "*younger than 20*"). These extended models are often referred to as *meta-population* models [Watts et al., 2005]. Apart from that, a homogeneous interaction structure is assumed. Effects such as super-spreaders (or overdispersion) are practically not expressible. The same holds for local die-outs. Moreover, the deterministic nature makes it difficult to conceptualize risk and uncertainty. ODE models arise as the mean-field limit of a well-mixed Markov population model corresponding to a complete graph in the network-based paradigm (cf. Chapter 8 (*From Networks to Population Models*) and Excursus 5).

Covid-19. Literature abounds with ODE-based Covid-19 models, some methods and applications are summarized in [Tang et al., 2020; Mello et al., 2021]. For instance, Dehning et al. (2020) use a model where the infection rate may change over time to predict a suitable timepoint to loosen NPIs in Germany. Lourenço et al. (2020) infer epidemiological parameters. Khailaie et al. (2020) analyze how changes in the reproduction number affect the epidemic dynamics. Stutt et al. (2020) evaluate the effectiveness of NPIs. The spread of Covid-19 was studied for many more countries and settings [M. S. Boudrioua and A. Boudrioua, 2020; De Visscher, 2020; Barbarossa et al., 2020; Dolbeault and Turinici, 2020; Wilson et al., 2020]. Other studies use a meta-population approach and group individuals according to age [R. Singh and Adhikari, 2020; Ellison, 2020; Prem et al., 2020; Klepac et al., 2020] or region [Afshordi et al., 2020; Humphries et al., 2020; Cooper et al., 2020]. Moreover, Neipel et al.

(2020) and Gomes et al. (2020) modify ODE models to account for individual variation in susceptibility.

Roda et al. (2020) use an ODE model to illustrate the general difficulty of predicting the spread of Covid-19 data. Limitations in the applicability of ODE models regarding data from Italy are reported by Comunian et al. (2020). Similar results are found by Castro et al. (2020) using Covid-19 data from Spain. General concerns are articulated by Bertozzi et al. (2020).

Branching Processes

Stochastic branching processes operate in discrete or continuous-time and are useful when studying the underlying stochastic nature of an epidemic. They are based on a tree that grows over time and represents the infected individuals. The children (offspring) of each node represent an individual's secondary infections, and the number of children is drawn from an *offspring distribution* with mean R_0 that is provided by the modeler [L. J. Allen, 2015; Farrington et al., 2003; Harris et al., 1963; Müller and Hösel, 2020].

Population heterogeneity. The offspring distribution makes it straightforward to encode individual variations in infectiousness or connectivity. The paradigm allows studying random extinction probabilities of the epidemic and the effects of super-spreaders/overdispersion [Lloyd-Smith et al., 2005]. However, branching processes do not admit a (model intrinsic) saturation due to growing immunity in the population. Moreover, the high level of abstraction makes it difficult to study the effects of NPIs and the characteristics of the spatial diffusion of the pathogen.

Covid-19. Yunjun Zhang et al. (2020) use a branching process to measure the dispersion of Covid-19 within China and Endo et al. (2020) estimate the dispersion based on local clusters outside China. Moreover, Tuite and Fisman (2020) use a branching process to infer epidemiological values and Goyal et al. (2020) study the influence of temporal viral load variation. Alternative model types that are used to study dynamical properties specific to Covid-19 were proposed in [Slavtchova-Bojkova, 2020; Yanev et al., 2020; Levesque et al., 2020].

Network-Based Models

Network-based epidemic models use graphs to express interactions (edges) among individuals (nodes). They are stochastic in nature and can be formulated in discrete or continuous time (cf. Chapter 2 (*Background*)).

Population heterogeneity. The network-based paradigm decouples the population's connectivity from the virus's infectiousness. Moreover, each individual is represented by an autonomous agent, which adds flexibility and makes it straightforward to include individual variations of the population. The key advantage of networks is that they represent a universal way of encoding different types of complex interaction structures like hubs, communities, households, small-worldness, mixing within in population-groups, etc. The contact network can also represent spatial or geographical constraints. Network-based models relate to ODE models in the sense that the ODE model represents the mean propagation of an epidemic on an infinite complete graph (all nodes are directly connected), assuming that all nodes are attributed with exponentially distributed jump times. Conceptually, the completeness "removes" the heterogene-

ity from the interaction structure, and the infinite size eliminates artifacts due to randomness.

Covid-19. Effects of different contact networks were studied in [Wolfram, 2020; Reich et al., 2020; C. Liu et al., 2020]. Contact networks are being used to build realistic simulations of a society, for instance, by creating household-structures with various types of inter-household connections [Munday et al., 2020; Nande et al., 2021; Kerr et al., 2020; Aleta et al., 2020]. The flexibility of networks makes it easy to model NPIs [Nande et al., 2021; Karaivanov, 2020; Kerr et al., 2020; Nielsen and Sneppen, 2020]. Moreover, [Silva et al., 2020; Biswas et al., 2020; Pujari and Shekatkar, 2020] use a network-based approach for spatial properties (e.g., flow between geographical regions). Although the importance of hubs was recognized very early, for instance by Pastor-Satorras and Vespignani (2001), the concrete relation to overdispersion as it is studied in branching processes remains under-explored. Networks, where the contact structure changes over time, are particularly well-suited to study quarantine measures and social distancing [Mancastroppa et al., 2020; Horstmeyer et al., 2020; Fagiolo, 2020].

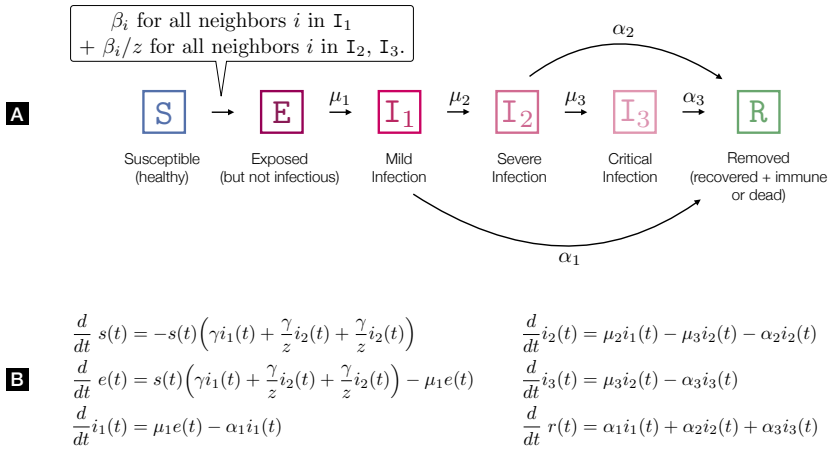


Fig. 6.2.: Covid-19 model. **(a)** Transitioning system of the network model with subject-level infectiousness (β_i for subject i). The transition rates refer to exponentially distributed residence times. **(b)** Corresponding ODE model with infection rate γ, where γ encodes connectivity and infectiousness.

Tab. 6.1.: Model parameters. We refer to Nande et al. (2021) for clinical justification of μ_j, β_j.

Parameter	Value	Meaning
β	0.0706	Infection rate (fixed, $R_0 = 2.5$, $k_{mean} = 8$)
β_i	—	Infection rate (variable) of subject i, $\beta_i \sim \nu(\cdot)$
$\nu(\cdot)$	—	Density of β_i with $E[\beta_i] = \beta$. E.g., $\nu = \text{Exp}(\beta^{-1})$
z	5.0	Reduction in infectivity in disease stages I_2, I_3
R_0	2.5	Basic reproduction number (for fixed β)
k_{mean}	≈ 8	Mean number of neighbors (by construction)
μ_1	1/5	Disease progression rate in E
μ_2	0.2/6	Disease progression rate in I_1
μ_3	0.25/6	Disease progression rate in I_2
α_1	0.8/6	Recovery rate in I_1
α_2	0.75/6	Recovery rate in I_2
α_3	1/8	Recovery/death rate in I_3
γ	≈ 0.394	Infection/connectivity for ODE ($R_0 = 2.5$)

6.3 Method

In this section, we show how to translate an ODE model to a network-based model in order to impose variation in connectivity and infectiousness while keeping the population averages of clinical and transmission parameters fixed. We use a Covid-19 ODE model that is heavily inspired by the SIR-extension of Nande et al. (2021). A summary of the model is depicted in Figure 6.2 and Table 6.1 (p. 131). We note upfront that we are only interested in qualitative results and do not rely on exact parameter values.

6.3.1 ODE Model

Our model contains six disease stages or compartments (cf. Figure 6.2, p. 131): *susceptible* (S), *exposed* (E) (infected but not yet infectious), *removed* (R) (immune or dead), as well as *mild*, *severe*, and *critical* infection stages (I_1, I_2, I_3). In contrast to Nande et al. (2021), we merge dead and recovered stages into a single *removed* stage, as both do not influence the infection dynamics further (we assume immunity after recovery). Note that perfect and permanent immunity is not given for Covid-19. We ignore the impact of re-infections. The fraction of individuals in each compartment evolves according to a system of ODEs given in Figure 6.2b. Because ODE models are invariant to the population size, we assume that the population is normalized. A further difference to Nande et al. (2021) is that we only have a single infection parameter γ. All other parameters have a meaningful clinical interpretation and can be specified accordingly (cf. Table 6.1).

The set of transition parameters γ, μ_j, α_j gives rise to a specific R_0. Hence, we can fix R_0 and thereby control γ (Excursus 4). We use $R_0 = 2.5$ which leads to $\gamma \approx 0.394$.

Excursus 4: Basic Reproduction Number of our Covid-19 Model

In an SIR ODE model (Figure 6.1, p. 126) with infection rate γ and recovery rate α, we find that

$$R_0 = \frac{\gamma}{\alpha}\,.$$

We can compute R_0 by assuming that an infinitesimal fraction ϵ (representing patient zero in an infinitely large population) is infected (I) and that the rest of the population $(1-\epsilon)$ is susceptible. Thus, we can naturally interpret R_0 as the slope of $i(t)$ (representing the fraction of the population in (I)). The slope of $i(t)$ is the ratio of inflow and outflow. The outlaw ($\alpha i(t)$) is proportional to the recovery rate, the inflow is proportional to the infection rate ($\gamma s(t) i(t) \approx \gamma i(t)$).
In the extended SIR ODE model, we, therefore, consider the outflow from S to E caused by the initially infected fraction ϵ while this fraction passes the three disease stages (taking into consideration that only a small part of ϵ reaches I_2 and even smaller part reaches I_3):

$$R_0 = \frac{\gamma}{\mu_2+\alpha_1} + \frac{\mu_2}{\mu_2+\alpha_1}\cdot\frac{\gamma/z}{\alpha_2+\mu_3} + \frac{\mu_2}{\mu_2+\alpha_1}\cdot\frac{\mu_3}{\mu_3+\alpha_2}\cdot\frac{\gamma/z}{\alpha_3}.$$

For instance, $\frac{\mu_2}{\mu_2+\alpha_1}$ refers to the fraction of ϵ that reaches I_2 and $\frac{\gamma/z}{\alpha_2+\mu_3}$ corresponds to the outflow of S attributed to this fraction.

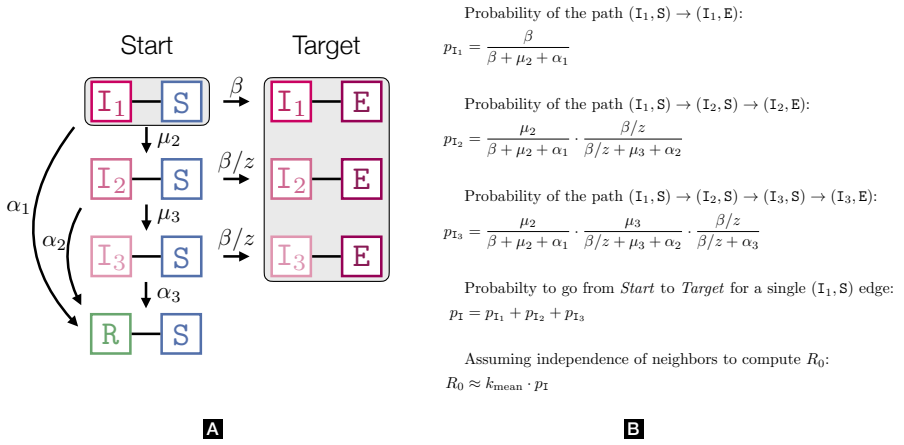


Fig. 6.3.: Computation of R_0 for fixed β. **(a)**: Representation of p_I as a reachability probability (from *Start* to *Target*) in a CTMC. **(b)**: The probability that an I_1-S edge transmits the infection, p_I, is the sum of p_{I_1}: probably that the infection is transmitted while the infected node is in I_1; p_{I_2}: Probability that I_1 transitions to I_2 (before transmitting the infection) and transmits while in I_2; and p_{I_3}: probability that the infection happens in I_3 (and not earlier). For individually varying β_i, $R_0 \approx k_{mean} E[p_I]$ is based on an integral over $\nu(\cdot)$.

6.3.2 Network-Based Dynamics

We use the compartments described in Figure 6.2 (p. 131). Nodes change their compartment following exponentially distributed residence times corresponding to specific instantaneous rates. For the transition from *susceptible* to *exposed*, the rate depends on the neighborhood of the node (cf. Figure 6.2a). We consider two cases: (i) all nodes have the same infection rate β and (ii): each node v_i has an individual infection rate β_i, sampled from a probability distribution with density $\nu(\cdot)$. We start with the former case. The relationship between γ and β is further discussed in Excursus 5.

Excursus 5: ODE Models and Networks

We use a simple SIR model to elucidate the relationship between γ (ODE model) and β (network model). The general form of the ODE model (assuming a population of N) is

$$\begin{aligned} \frac{d}{dt}s(t) &= -\frac{\gamma}{N}s(t)i(t) \\ \frac{d}{dt}i(t) &= \frac{\gamma}{N}s(t)i(t) - \beta i(t) \\ \frac{d}{dt}r(t) &= \beta i(t), \end{aligned} \tag{6.1}$$

where $s(t)$ determines the (real-valued) number of individuals in compartment S at time t. Typically, we can assume $N = 1$ and speak of fractions in each compartment because the population size is irrelevant (only determines the curves' scaling).

Computing R_0 using same idea as in Figure 6.3 (p. 134) yields:

$$R_0 = k_{mean}\frac{\beta}{\alpha + \beta}$$

which we can reorder as follows:

$$\beta = \frac{\beta R_0}{k_{mean} - R_0}\,.$$

However, $\beta R_0 = \gamma$ (following from Excursus 4). Hence, we can express β in terms of γ:

$$\beta = \frac{\gamma}{k_{mean} - R_0}\,,$$

Complete Graph. Consider the complete graph with n nodes in the limit of $n \to \infty$, then

$$\lim_{n\to\infty} \beta = \frac{\gamma}{n}$$

which is equivalent to the effective infection rate in the ODE model $\frac{\gamma}{N}$ for population size N (cf. Eq. (6.1)).
This means that fixing R_0 yields the same rate for the ODE model and for the limit of the complete graph. This carries over to the (expected) trajectory of the dynamics. Thus, we can even go one step further and consider the ODE as a mean-field approximation of the complete graph CTMC. In our results (Figure 6.6, p. 142), we already see a considerable similarity between the complete graph and the ODE for only 10^3 nodes.

Case (i): Homogeneous Infectiousness

Each $S\text{-}I_1$ pair transmits an infection with rate β. If the infected node is in compartment I_2 or I_3, the infectiousness decreases and is given by β/z. Note that we use exponentially distributed residence times which are potentially less realistic than, for instance, beta-distributions [Nande et al., 2021], but these relate directly to ODE models. Hence, observed differences in the dynamics can be attributed to the connectivity/stochasticity, not to the residence time's shape.

R_0 is defined as the expected number of neighbors that patient zero infects in a susceptible population. Thus, R_0 cannot be larger than the mean degree. In the case of a fixed infection rate β, fixing k_{mean} also determines R_0. We can approximate R_0 as shown in Figure 6.3. We use that each infection happens independently and approximate $R_0 \approx p_I k_{mean}$ where p_I denotes the probability that patient zero infects a random neighbor (while potentially transitioning to I_2, I_3). The approximation comes from the fact that an already infected neighbor can infect another neighbor of patient zero, violating the independence assumption, rendering this an over-approximation.

Note that p_I is conceptually similar to the *secondary attack rate* in a completely susceptible population. We construct the networks such that $k_{mean} = 8$ (except for the complete graph where k_{mean} is the number of nodes minus one). Like in the ODE-approach, we fix $R_0 = 2.5$ and determine $\beta = 0.0706$ (cf. Figure 6.3 (p. 134)).

Case (ii): Individual Differences in Infectiousness

In the case of individually varying infectiousness, we associate each node v_i with infection rate β_i that is drawn from a distribution with density $\nu(\cdot)$. Again, our goal is to introduce variation while keeping the population mean unchanged. Hence, we construct $\nu(\cdot)$ such that $E[\beta_i] = 0.0706$. We define R_0^i as the node-dependent basic reproduction number when the infection starts in node i. Moreover, we define the node-independent basic reproduction number as the corresponding unweighted mean $R_0 = E[R_0^i]$. Interestingly, different $\nu(\cdot)$ (with the same mean) can lead to different R_0. Theoretically, this follows from the computation of p_I which is now based on an integral over $\nu(\cdot)$. In the next section, we set $\nu(\cdot)$ to an exponential distribution and study the resulting changes in the dynamics. A key takeaway of our study is that increasing the variance in the degree distribution does not change R_0. Increasing the variance in individual infectiousness does so (in fact, it decreases R_0). For the evaluation, we use an exponentially distributed β_i with an expected value of 0.0706. That is,

$$\nu(x) = \beta^{-1} e^{-\beta^{-1} x}$$

with $\beta^{-1} = 1/0.0706$ (recall that $\beta = 0.0706$ lead to $R_0 = 2.5$ if no population heterogeneity is present).

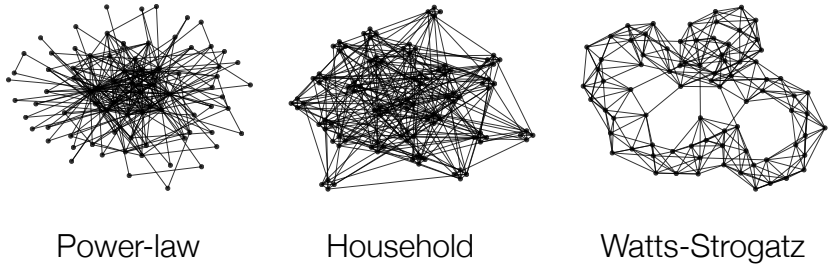


Fig. 6.4.: Schematic visualizations of random graph models with 80 nodes and $k_{mean} = 8$.

6.3.3 Human-to-Human Contact Networks

We test different types of contact networks that highlight different characteristics of real-world human-to-human connectivity. To this end, we describe the contact networks using random graph models. We create a specific realization (variate) of such a random graph model in each simulation run. A schematic visualization of example networks is provided in Figure 6.4. We use a **complete graph** (each possible pair of nodes is connected) as a baseline to study the evolution of the epidemic when no contact structure is present. Thereby, we can mimic the effects of stochasticity and variation in infectiousness while keeping the simulation as close as possible to the assumptions underlying the ODE. We use **Power-law Configuration Model** networks to study the effects of hubs (potential super-spreaders). These networks are—apart from being constrained on having power-law degree distribution—completely random. The power-law degree distribution is omnipresent in real-world networks and entails a small number of nodes with a very high degree. We fix the minimum degree to be two and choose the power-law parameter numerically so that the network admits the desired mean degree. We also test a synthetically generated **Household** network that was loosely inspired by Ball et al. (2010). Each household is a clique. The edges between households represent connections due to work,

education, shopping, leisure, etc. We use a geometric network to generate the global inter-household structure. The household size is 4. In the case of $k_{mean} = 8$, each node has 3 edges within its household and (on average) 5 outgoing edges. We also compute results for **Watts–Strogatz** (WS) random networks. They are based on a ring topology with random re-wiring. Each node has exactly k_{mean} neighbors. We use a small re-wiring probability of 5% to highlight the locality of real-world epidemics.

Apart from the baseline (complete graph), we specifically use these three network models because they are well-studied in literature and very different in their respective global properties. Moreover, they all encode important properties of human-to-human connectivity like hubs (power-law), small-worldness (power-law and WS), and tightly connected household structures.

6.3.4 Dispersion in Networks

Given independent simulation runs, we measure dispersion by analyzing the empirical offspring distribution at day t. Specifically, we consider the offspring distributions of the nodes that were exposed within day t (the actual secondary infections may happen later). We also perform a discretization of time over intervals of one day. We quantify dispersion in three ways:

Coefficient of variation (CoV)

Together with the mean of the offspring distribution R_t, we report the CoV, the ratio of standard deviation to mean. The CoV is a widely-used measure of dispersion of a probability distribution.

Top-k

We explicitly report how many new infections within day t are

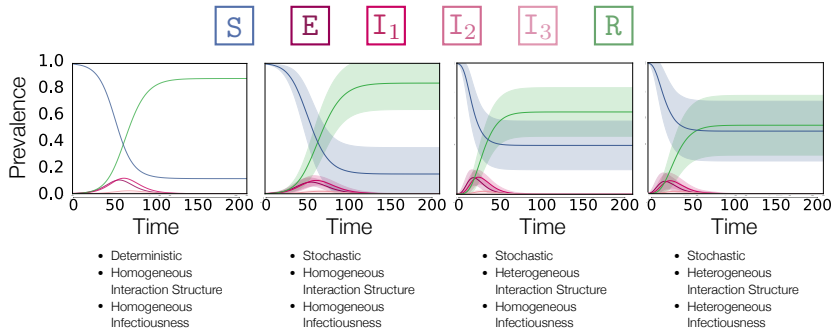


Fig. 6.5.: Overview of how population heterogeneity shapes an epidemic. **f.l.t.r.:** ODE model, a complete graph, a power-law network, and a power-law network with exponentially distributed infectiousness. The mean fraction of the population in each compartment at each point in time is shown. Shaded areas indicate standard deviations, not confidence intervals.

caused by which fraction of infected nodes (e.g., 80% of the new infections are caused by 20% of the nodes). We report which fraction of new infections can be traced back to (the most harmful) 10%, 20%, and 30% of infected nodes.

Offspring

We report the fraction of nodes (that were infected within day t) that lead to $0, 1, 2, \ldots$ children.

Note that overdispersion inevitably indicates not only the existence of super-spreaders but also the existence of nodes that are unlikely to pass the infection at all.

6.4 Experiments

We compare the evolution and dispersion of the four network models. We have two main experiments. In the first experiment (overview in Figure 6.7, p. 144), we study a fixed infection rate (mimicking

the case that there is only variation in connectivity). In the second experiment (Figure 6.8, p. 145), we additionally impose individual variation in infectiousness β_i. Recall that the networks (aside from the complete graph) have approximately the same number of edges and that nodes approximately admit the same mean infectiousness, thereby ensuring that the resulting differences are solely a consequence of the corresponding variation.

Figures Figure 6.5 (p. 140) and Figure 6.6 (p. 142) summarize some of our key findings. Figure 6.5 visualizes the effects of adding stochasticity and individual variation to a population. Figure 6.6 highlights the different dynamics emerging on different contact networks.

Setup. For each network, we perform 10^3 simulation runs on a network with 10^3 nodes. Networks are generated such that the mean degree is approximately eight. For network models where we cannot directly control k_{mean}, we start by generating sparse networks and increase the density until k_{mean} has the desired value. In each simulation run, we start with three randomly chosen infected nodes in I_1 (to reduce the likelihood of instantaneous initial die-outs). The ODE (cf. Figure 6.6, p. 142) starts with a value of $3/1000$ in I_1. The number of simulation runs is enough to estimate the mean fractions (and the standard deviations) corresponding to each compartment with high accuracy (confidence intervals are not shown, but would be barely visible anyway). The number of 10^3 nodes was used for practical reasons, however, increasing the network size preserves the qualitative characteristics of the dynamics.

Quantities of Interest. We characterize epidemics in terms of the evolution of population fractions, that is, mean fraction (prevalence) of nodes in compartment S , I_1, R for each timepoint t (the remaining compartments evolve approximately proportional to I_1, thus,

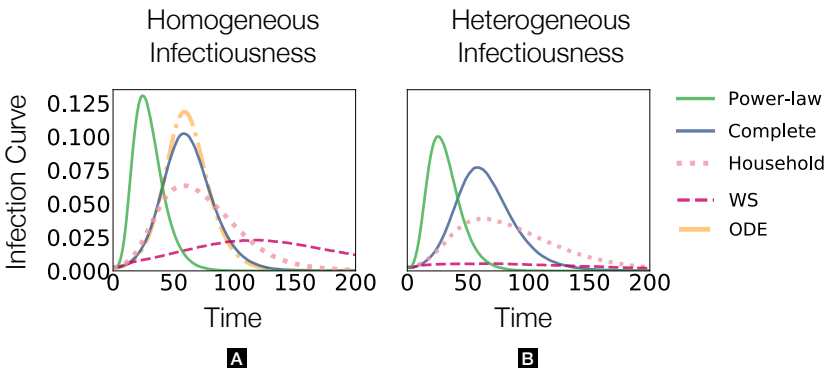


Fig. 6.6.: Fraction of nodes in I_1 (y-axis) over time **(a)**: Fixed infection rate β. **(b)**: Node-based infection rate β_i drawn from an exponential distribution. Note the large difference between the two evolutions on the Watts–Strogatz (WS) networks.

we leave them out for clarity). This evolution informs about the timepoint and the height of the infection peak and the final epidemic size (or *herd immunity threshold*) that is equivalent to the (mean) fraction of recovered nodes when the epidemic is over (which is mostly the case at 200 time units). Note that the final death toll is proportional to the final epidemic size.

Moreover, we study the effective reproduction number R_t (2nd row in Figure 6.7 and Figure 6.8, p. 145). We define R_t as the average number of secondary infections for a node exposed at day $t \in \mathbb{N}_{\geqslant 0}$. We also report an empirical R_0 based on the three initially infected nodes that diverges slightly from the theoretical R_0 in Table 6.1. Dispersion is quantified using the three techniques explained in Section 6.3.4 (2nd to 4th row in Figure 6.7 and Figure 6.8, p. 145).

6.4.1 Experiment 1: Connectivity Heterogeneity

Results for a fixed β on different network types are shown in Figure 6.7 (p. 144). In most simulation runs, the power-law dynamics

admits a very early peak, and the epidemic dies out early with a comparably small final epidemic size. This effect can directly be attributed to the hubs that get infected very early and jump-start the epidemic. In contrast, in household networks—and even more so in WS networks—the infection peak is lower and happens at a later timepoint. This is no surprise as the connectivity in both networks imposes a level of locality that slows down the propagation. For better visibility, the differences in the infection curve (based on I_1) are summarized in Figure 6.6 (p. 142). We also see that the complete graph behaves similarly to the ODE model.

The effective reproduction number R_t starts from around 2.5 (the theoretical over-approximation) and decreases in most cases monotonically. The exception is again the power-law network where hubs cause a massive jump of R_t in the first day from 2.5 to around 12. This jump is also reflected in the dispersion measure, most noticeable in the offspring plot (4th row). The power-law topology generally admits a higher dispersion than the other networks. For instance, the fraction of nodes with zero offspring is much higher. Moreover, on most days, the top 20% of the infected nodes account for more than 80% of the new infections, roughly fitting the estimations for Covid-19 in a typical population. In the other network types, there is less temporal variation in the dispersion. The dispersion is the lowest in the WS networks, which is unsurprising as all nodes have degree 8 (providing an upper bound to the offspring number). Generally speaking, we see that dispersion can be robustly measured using the empirical offspring distribution.

Note that measuring the dispersion becomes increasingly difficult over time for the power-law network. This is because the epidemic tends to die out early with a high probability. Thus, the dispersion is estimated on a relatively small amount of samples. At the same time, the variance of the distribution is comparably high. This leads to a noticeable amount of noise.

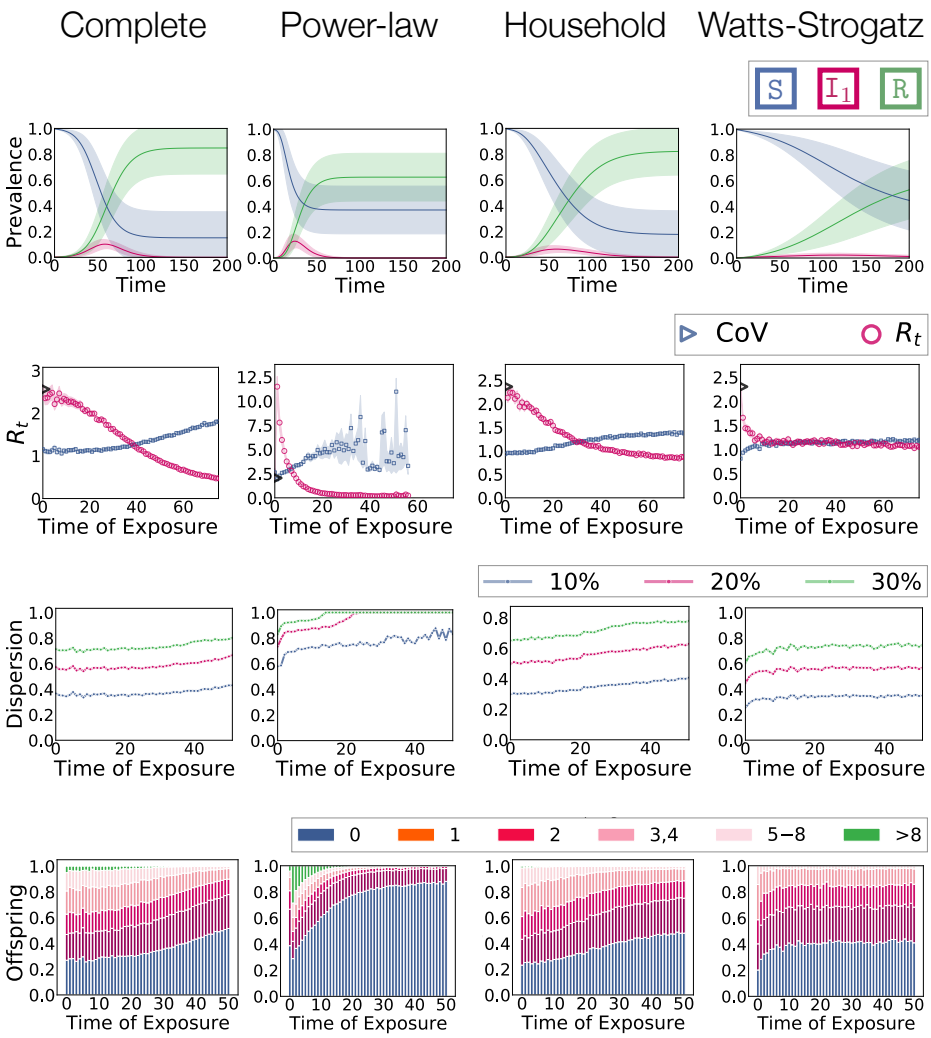


Fig. 6.7.: **Experiment 1**: Epidemic dynamics of different network types. **Row 1**: Evolution in terms of mean fractions (and standard deviation) in each compartment over time. **Row 2**: Effective reproduction number R_t (the empirical R_0 is shown as a black triangle) and coefficient of variation of the offspring distribution (with 95% CI, note the significant amount of noise in the power-law case). **Row 3**: Top-k plots: The fraction of new infections that can be attributed to a particular fraction of infected nodes. **Row 4**: Characterization of the offspring distribution in terms of the fraction of nodes that cause a specific number of secondary infections.

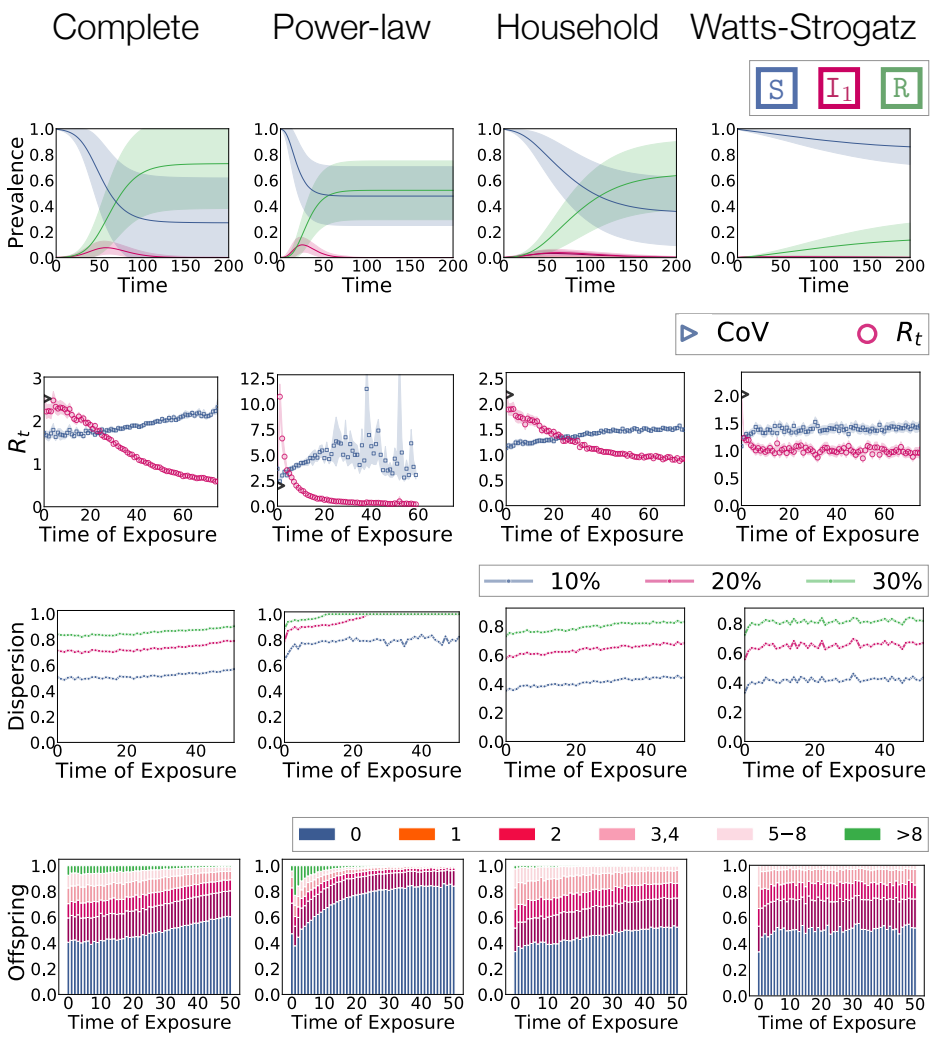


Fig. 6.8.: **Experiment 2**: Epidemic dynamics with individual infectiousness β_i **Row 1**: Evolution in terms of mean fractions in each compartment over time **Row 2**: Effective reproduction number R_t and coefficient of variance of the offspring distribution. **Row 3**: Top-k plots: what fraction of new infection can be attributed to which fraction of infected nodes. **Row 4**: Characterization of the offspring distribution in terms of what fraction of nodes has how many offspring (infect how many neighbors).

6.4.2 Experiment 2: Infectiousness Heterogeneity

In this experiment, we draw in each simulation run for each node v_i, a random β_i that is distributed according to $\nu(\cdot)$. Here we use an exponential distribution. The effects on the evolution and dispersion are reported in Figure 6.8 (p. 145). In all networks, the epidemic becomes "weaker" in terms of final epidemic size and infection peak height (see also Figure 6.6, p. 142). This effect is strongest in the WS network (where the epidemic dies out almost immediately) and weakest in the complete graph. This is also mirrored in the difference of R_0 compared to Experiment 1 (as explained in Figure 6.3 (p. 134), the relationship between β and R_0 is now non-linear). The complete graph leaves R_0 almost unchanged (i.e., around 2.5) while it goes down to around 2.0 in the WS network. Effects on the household and power-law network are less drastic but still evident.

Regarding the dispersion, the differences to Experiment 1 are expected. The variance in the empirical offspring distribution generally increases. Interestingly, this happens in all networks by roughly the same amount, regardless of whether the dispersion was high or low in the first case. We can also consistently see the change in the dispersion in all three dispersion measures, but the differences are especially evident in the top-k plots (3rd row). It is also interesting to see that all networks admit a characteristic signature in the histogram of the offspring distribution (4th row). The infection rate variation shifts these plots (in particular because the number of nodes without offspring increases), but they still entail a clear distinction between networks.

We also tested uniform and gamma distributions for $\nu(\cdot)$ (results not shown), and found that the epidemic generally becomes weaker with higher variance. We expect this to be due to an increased likelihood of local and global die-outs.

6.4.3 Discussion

The two experiments show that population heterogeneity strongly influences the fundamental quantities of an epidemic. However, there are important differences between the sources of variations:

- The existence of hubs in the network can cause R_t to increase, but individual variations in infectiousness cannot.
- Different networks with the same k_{mean} and a fixed β will approximately admit the same R_0. A variable β (with a fixed mean) will change R_0 depending on the shape of the underlying density.
- Individual variations in the infectiousness generally weaken the epidemic's impact. However, individual variations in connectivity may worsen some aspects (e.g., the height of the infection peak).
- β has the weakest influence when the interaction structure is homogeneous (i.e., on a complete graph) and the strongest when the interaction structure is based on locality (the average distances in the graph increase) as in the WS network.
- Varying the infectiousness increases the stochasticity (e.g., the variance of the number of infected nodes at any given time).
- The interaction structure has a large influence on the dispersion. Individual variations in infectiousness induce a smaller but consistent increase in dispersion.
- Hubs influence how the dispersion changes over time. Individual variations in infectiousness increase the dispersion consistently for all timepoints.

Our results underline that networks are a feasible tool to encode various features of a population's interaction structure. Generally speaking, it is not surprising that some networks support the formation of epidemics better than others. To some extent, this has been studied in terms of the epidemic threshold of graphs [Prakash, Chakrabarti, et al., 2012]. However, the variety of the influence of the networks and the interplay between heterogeneity in the infectiousness and dispersion is remarkable. There are even further possibilities to adjust population heterogeneity, e.g., by adding non-Markovian residence times in the compartments, varying the remaining transition rates, or by imposing more temporal variability in infectiousness. Our results show that models based on point estimators of population averages (i.e., most mean-field ODE models) are not adequate for analyzing or predicting the dynamics of an epidemic.

Regarding the dispersion, we see that none of the considered network structures by itself leads to a dispersion where 80% of the infections are caused by only 15% of the infected nodes as it is reported for Covid-19 [K. Sun et al., 2020] (at least not right from the beginning). From branching process theory, it is known that a higher dispersion increases the die-out probability [Lloyd-Smith et al., 2005]. Generally, this effect also holds for networks. For a fixed network, increasing dispersion by using a probabilistic infection rate does increase the die-out probability. However, the network topology strongly modulates the strength of this effect.

In conclusion, we find that, in most cases, population diversity makes an epidemic less harmful but increases the dispersion and the variability of the evolution. Hubs in contact networks are a notable exception to this rule.

Hubs vs. Highly Infectious Subjects. Hubs (high-degree individuals) and individuals that are very infectious (but with an average degree) are both drivers of the epidemic. However, hubs are a special case because they infect many others and also become infected very early. Hence, they can cause an explosive surge at the epidemic's beginning. In contrast, high infectiousness alone does not make individuals more likely to be infected earlier than others. Hubs also highlight that the effective reproduction number can change significantly while the environmental conditions remain the same simply because in the beginning the prevalence shifts towards highly connected individuals.

Implications. Considering that an exponentially distributed β_i can be regarded as a fairly strong assumption about individual differences, our work can—with necessary caution—be seen as further evidence that the network structure plays a more critical role in the dispersion than individual viral load variability. Transferring these characteristics to NPIs, our work indicates that reducing long-range connections (e.g., by corresponding mobility restrictions) and keeping degree-variability small (to avoid hubs) are particularly effective control strategies. Reducing mobility seems to be especially effective for overdispersed epidemics. We further investigate the relationship between overdispersion and mobility restrictions in [Großmann, Backenköhler, and Wolf, 2021a].

6.5 Conclusions

We tested the influence of heterogeneity in the population's interaction structure and degree of individual infectiousness on the dynamical evolution of an epidemic. We find that the dynamics depend strongly on these properties and witness an intriguing interplay between these two sources of variation. Our work also highlights the role of small-worldness, local die-outs, and super-spreaders in an epidemic.

Naturally, mathematical modeling is based on assumptions and abstractions. However, heterogeneity seems particularly crucial, and excluding it should only be done with great caution. It is particularly challenging to capture population heterogeneity in the widely-used class of ODE models. Due to their inherent homogeneity assumptions w.r.t. each compartment, the complex interplay of varying infectiousness and connectivity remains elusive for such models. Discussing epidemics in terms of population averages may not adequately reflect the complexity of the emerging dynamics.

At a broader level, this work highlights limitations of certain model classes and shows that subtle differences in assumptions can make substantial differences. This simulation study is a reminder that models are prone to hidden assumptions and that we should be cautious with their interpretation.

Part III

Reduction and Inference

7 Birth-Death Process Abstraction

Until now, we have used stochastic simulations to explore spreading processes on networks. These are particularly useful when one is interested in descriptive statistics on mean behavior, but are limited in scope otherwise. Unfortunately, the direct computational analysis of these processes (i.e., of the induced CTMC of Section 2.5) is hindered by the enormous size of the underlying state space.

This chapter studies the SIS epidemic model (Model 1) and its induced CTMC. We propose using *lumping* to reduce the original dynamics to a parameterized birth-death process. Building and solving the reduced model is computationally efficient even for large-scale contact networks. The chapter also acts as an introduction to key CTMC concepts. Julia code is made available[1].

Our **key idea** is to exploit that the infection rate of a network-state only depends on the number of SI-edges. Given a contact network and the number of infected nodes, we bound this number using graph analysis. Their over- and under-approximation of SI-edges translate to the over- and under-approximation of the prevalence of infected nodes.

[1] github.com/gerritgr/BD-Reduction

7.1 Problem Setting

A crucial value for any SIS model is the *effective infection rate* $\frac{\beta}{\alpha}$ [Prakash, Chakrabarti, et al., 2012], defined as the ratio between infection and recovery rate. It determines the epidemics' characteristics, such as their long-time behavior. For simplicity, we typically assume a recovery rate of $\alpha = 1$ and consider β to be the effective infection rate. We define the *infection footprint* $\tau(\beta)$ to be the expected fraction of infected nodes in equilibrium (of the induced CTMC) as our value of interest. Let $\mathcal{X}$ denote the set of network-states for a given contact network and let $\pi_\beta \in \text{Cat}(\mathcal{X})$ denote the equilibrium (we discuss its existence and uniqueness later) distribution over the network-states of the CTMC for infection rate β. We define

$$\tau(\beta) = \sum_{\mathbf{x}_i \in \mathcal{X}} \pi_\beta[i] \cdot N_I(\mathbf{x}_i) \,, \tag{7.1}$$

where $N_I(\mathbf{x}_i)$ denotes the number of infected nodes in network-state $\mathbf{x}_i$ (and we assume an implicit enumeration over $\mathcal{X}$).

The goal of this study is to estimate $\tau(\cdot)$ efficiently.

We could approximate $\tau(\cdot)$ using simulations. However, we would need (i) a massive number of (ii) very long (to reach equilibrium) simulation runs for (iii) each potential value of β. This makes simulation-based approaches unsuitable. Solving the CTMC (using numeral integration) is also intractable due to the enormous size of its state space (2^n network-states).

Trap State. The original SIS model admits what is called *extinction* or *die-out* behavior [Bovenkamp and P. Van Mieghem, 2014]. That is, the Markov chain will eventually reach a network-state where

all nodes are susceptible. In CTMC terminology, this is called a *trap state* because it cannot be left. The trap state renders the analysis of the equilibrium meaningless. To circumvent this problem, we always ensure that at least one node is infected (formally, we set the recovery rate to zero if precisely one node is infected). Thus, we obtain a meaningful equilibrium distribution (technically, we translate the *meta-stable* macro-state into an equilibrium). Another viable method would be to include self-infections ($\mathtt{S} \to \mathtt{I}$) with a small rate ϵ, which leads to comparable results.

7.2 Method

Given a contact network and $\beta \in \mathbb{R}_{\geqslant 0}$, we build two parametrized birth-death processes, denoted $\mathrm{BD}_{\mathrm{min}}$ and $\mathrm{BD}_{\mathrm{max}}$. A birth-death process is a continuous-time Markov chain that represents the current population size of infected agents. $\mathrm{BD}_{\mathrm{min}}$ aims to under-approximate and $\mathrm{BD}_{\mathrm{max}}$ aims to over-approximate the infection footprint.

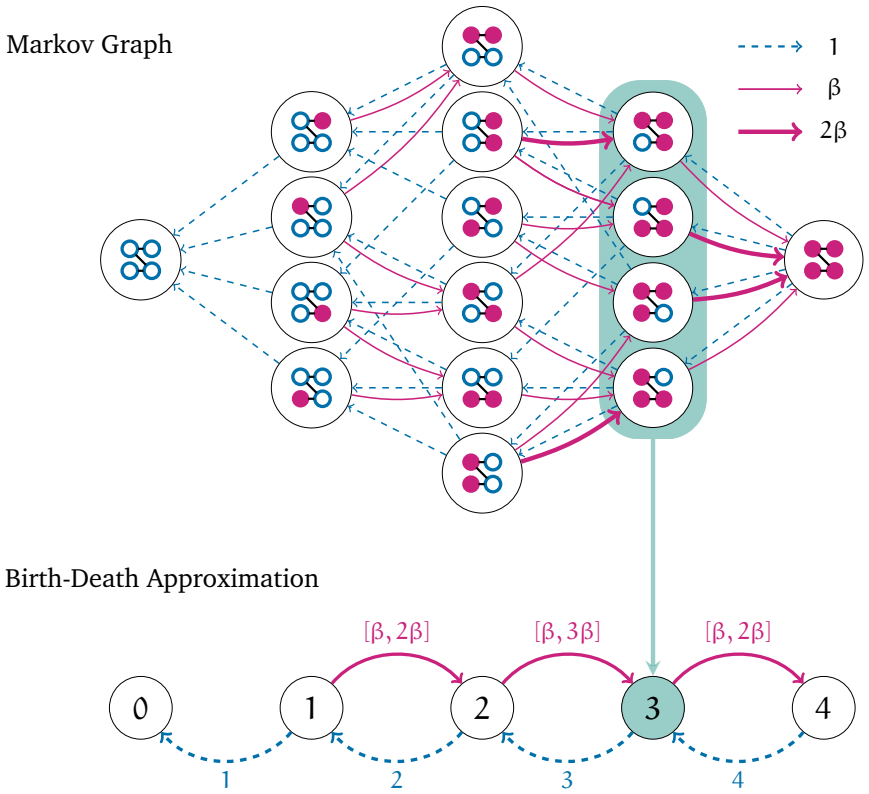


Fig. 7.1.: Original and reduced SIS model. **Top**: Original Markov graph (S: *blue*, I: *pink, filled*) for a 4-node contact network. The complete Markov graph is shown without self-infections and including the trap state with all susceptible nodes (left). **Bottom**: Birth-death model abstraction with minimal and maximal birth rates. The state identifies the number of infected nodes. The reduction is shown in green.

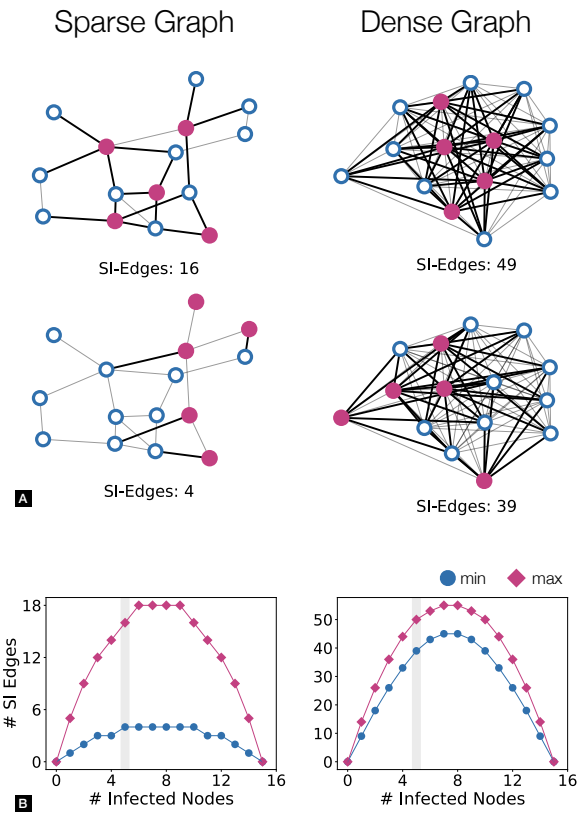


Fig. 7.2.: **(a)**: Highest (top row) and lowest (bottom row) possible number of SI-edges (bold) on a sparse (left) and dense (right) contact network for 5 infected nodes. **(b)**: Highest and lowest possible number of SI-edges as a function of the number of infected nodes. The symmetry of the curve emerges from the fact that opposite points relate to networks where the S and I labels are switched. The shaded area indicates the case of 5 infected nodes, as shown in **(a)**.

7.2.1 Birth-Death Process Construction

The birth-death-processes have n states (or $n+1$ states when we include the case of zero infected nodes) (cf. Figure 7.1, p. 156). The state space is denoted

$$\mathcal{Z}_{\min} = \mathcal{Z}_{\max} = \{1, \ldots, n\}$$

for $BD_{\min}$ and $BD_{\max}$, respectively.

In each step, one can only go from state i to $i+1$ (infection) or to $i-1$ (recovery). We use $\lambda^{\min}$ and $\lambda^{\max}$ to denote the rates of $BD_{\min}$ and $BD_{\max}$, respectively. The recovery rate is equal in both birth-death processes and (using $\alpha = 1$) is given by

$$\lambda^{\min}_{i \to i-1} = \lambda^{\max}_{i \to i-1} = i\,.$$

In the original CTMC, the joint rate for a new infection is determined by the number of SI-edges of the current network-state. In the BD-process, we only know the number of infected nodes, which might correspond to a wide range of possible SI-edge counts (cf. Example 7.1).

Thus, we use the approximations:

$$\lambda^{\min}_{i \to i+1} = \beta \mathrm{SI}_{\min}(\mathcal{G}, i) \quad \text{and}$$
$$\lambda^{\max}_{i \to i+1} = \beta \mathrm{SI}_{\max}(\mathcal{G}, i)$$

where $\mathrm{SI}_{\min}(\mathcal{G}, i)$ denotes the minimum number of SI-edges for a given graph $\mathcal{G}$ assuming that i nodes are infected. Likewise, $\mathrm{SI}_{\max}(\mathcal{G}, i)$ denotes the maximum number of SI-edges. Note that birth rates ($i \to i+1$) are parameterized by the effective infection rate β.

Example 7.1: Number of SI-Edge Counts

Consider a graph with 15 nodes, where 5 nodes are infected (cf. Figure 7.2a, p. 157). The joint recovery rate (from i to $i-1$) will always be 5 (regardless of which nodes are actually infected). In contrast, the joint infection rate depends on the number of SI-edges which may vary depending on which nodes are infected. For sparse graphs, the range of possible SI-edge counts is particularly large (cf. Figure 7.2b, p. 157).

We can efficiently determine the equilibria of those two processes and compute the induced footprint analogously to Eq. (7.1), yielding

$$\tau_{\min}(\beta) \leqslant \tau(\beta) \leqslant \tau_{\max}(\beta) \;\forall \beta > 0\,.$$

However, while solving the birth-death processes is cheap, we first need to compute $\mathrm{SI}_{\min}(\mathcal{G}, i)$ and $\mathrm{SI}_{\max}(\mathcal{G}, i)$ in order to identify the rates. Computing the exact values is, however, intractable. Specifically, due to the non-monotonicity of the problem, we would need to solve an intractable problem for each i (cf. Excursus 6). To make our method computationally efficient, we use a simple heuristic. Empirically, we observe that the approximate bounds capture $\tau(\cdot)$ very well.

Excursus 6: Cutting Graphs and Counting Edges

To determine the rate in the birth-death process, we need to identify the minimum (and maximum) number of SI-edges, given the number of infected nodes. The problem is related to the well-known *maximum cut* (max-cut) and *minimum cut* (min-cut) problems in graph theory, where a graph is split into two partitions. The two partitions encode the infected and susceptible nodes, respectively. The size (i.e., the boundary) of the cut represents the number of SI-edges. The decision problem corresponding to max-cut is known to be $\mathcal{NP}$-complete [Karp, 1972]. Min-cut can be solved efficiently (for instance, following the work of Orlin (2013) that is based on the famous max-flow min-cut theorem [Gomory and T. C. Hu, 1961]. We refer to Gawrychowski et al. (2019) for an overview).

However, in contrast to standard max-cut and min-cut, we have the additional constraint that one of the two partitions has size i (and we need to compute the cut size for each i). The given number of infected nodes i makes even the exact computation of the minimum SI-edge count much harder. Specifically, W. Chen et al. (2016) show that this is $\mathcal{NP}$-hard (denoted the *minimum* s–t *cut with exactly* k *vertices problem*). Similar problems are studied by Ji (2004) and Devriendt and Piet Van Mieghem (2019).

7.2.2 Our Method: MAGENTA

We propose MAGENTA ([Ma]ximal Disa[g]r[e]eme[nt] [A]lgorihtm) to efficiently approximate $SI_{min}(\mathcal{G}, i)$ (resp. $SI_{max}(\mathcal{G}, i)$), the minimum (resp. maximum) number of SI-edges for a given contact network and number of infected nodes. We apply two tricks:

1. Formulate a surrogate problem that is monotone.

2. Use an efficient greedy method to approximate the surrogate problem.

We use the term *monotone* to indicate that a solution to $SI_{max}(\mathcal{G}, i)$ is a subset of the solution to $SI_{max}(\mathcal{G}, i+1)$ (and analogously for $SI_{min}(\mathcal{G}, i)$). Generally, this is not the case. For instance, assume we want to maximize the number of SI-edges. The set of infected nodes for $i = 1$ might be disjunct with the set of infected nodes for $i = 2$.

We consider a monotone surrogate because we do not want to solve an independent problem for each i.

Surrogate Problem

We optimize a ranking $r(\cdot)$ over the nodes. A ranking is a bijective mapping

$$r : \{1, \dots, n\} \to \{1, \dots, n\},$$

where node v_i is mapped to position $r(i)$. We measure the quality of a ranking by evaluating its "(dis)agreement" over edges:

$$S(r) = \sum_{(i,j)\in\mathcal{E}} \left| r(i) - r(j) \right| .$$

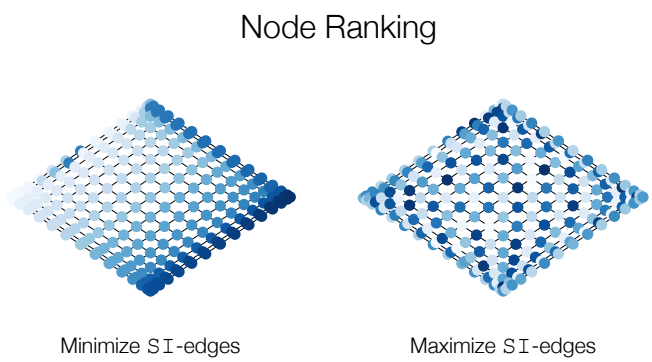


Fig. 7.3.: Node ranking according to our greedy approach on a 15×15 grid graph. The position $r(i)$ of node v_i is indicated by its color (light to dark).

To approximate the maximum (resp. maximum) number of SI-edges, we then try to find a ranking r that minimizes (resp. maximizes) the disagreement score $S(r)$.

Solving the Surrogate. We use a simple greedy algorithm to find a solution for $S(r)$. We start with an empty set (representing zero infected nodes). In each step, we add one node to this set and assume all nodes in the set are infected. We always add the node to the set that minimizes (maximizes) the current number of SI-edges. To make the algorithm efficient, we use a priority queue to decide which node to choose next. Figure 7.3 depicts the node rankings on two small graphs.

There are many other viable ways to approach this problem. For instance, discrete optimization (like simulated annealing) could be used to improve the ranking w.r.t. $S(r)$. Relaxation of the problem is also possible. Notably, the Laplacian matrix of the contact network [Piet Van Mieghem, 2010] can be used to score and optimize a (relaxed) node ranking. Moreover, we could use the so-called *Fiedler vector*. That is the eigenvector of the second-smallest eigenvalue of the Laplacian matrix of the contact network (named based on the

seminal work of Fiedler (1973)). It directly provides a node ranking for SI-edge minimization (the corresponding optimization problem is similar to $S(r)$). Likewise, the eigenvector of the largest eigenvalue of the Laplacian matrix can be used for SI-edge maximization. However, using spectral approaches or discrete optimization, we found only minor improvements over the greedy method (results not shown).

Applying the Surrogate

We denote the final rankings $r_{min}(\cdot)$ and $r_{max}(\cdot)$, respectively. For a given i, we can then define the corresponding partitions:

$$\mathcal{V}_{min}(i) = \{v_j \in \mathcal{V} \mid r_{min}(j) \leqslant i\} \text{ and}$$
$$\mathcal{V}_{max}(i) = \{v_j \in \mathcal{V} \mid r_{max}(j) \leqslant i\}\,.$$

Hence, $SI_{min}(\mathcal{G}, i)$ is the number of edges between $\mathcal{V}_{min}(i)$ and $\mathcal{V}\backslash\mathcal{V}_{min}(i)$ (and $SI_{max}(\mathcal{G}, i)$ is the number of edges between $\mathcal{V}_{max}(i)$ and $\mathcal{V}\backslash\mathcal{V}_{max}(i)$). Furthermore, we can exploit the symmetry of the problem (cf. Figure 7.2b, p. 157) and consider:

$$SI^{sym}_{min}(\mathcal{G}, i) = \min\Big(SI_{min}(\mathcal{G}, i),\ SI_{min}(\mathcal{G}, n-i)\Big)$$
$$SI^{sym}_{max}(\mathcal{G}, i) = \max\Big(SI_{min}(\mathcal{G}, i),\ SI_{max}(\mathcal{G}, n-i)\Big)\,.$$

The key advantage of MAGENTA is that we only need to optimize two rankings (one for minimization and one for maximization). While this is only an approximation without formal guarantees, it works well in practice. We believe that the reason for that lies in the fact that the dynamics of the SIS model is such that the extreme values do not affect the mean behavior of the process significantly.

7.2.3 Solving the Birth-Death Process

Finally, we need to find the equilibrium of the birth-death-processes. At equilibrium, the probabilities of a CTMC satisfy the *global-balance equation*, that is, the outflow of probability mass of each state needs to be equal to its inflow. Hence, the equilibrium distribution $\pi \in \text{Cat}(n)$ needs all i to satisfy:

$$\underbrace{(\lambda_{i\to i+1} + \lambda_{i\to i-1})\,\pi[i]}_{\text{outflow of } i} = \underbrace{\lambda_{i+1\to i}\pi[i+1] + \lambda_{i-1\to i}\pi[i-1]}_{\text{inflow towards } i}\,.$$

We can do this by solving a corresponding equation system with n equations (cf. Excursus 2, p. 37). However, there is a more elegant way. For birth-death processes, the balance condition also holds when we consider only the state to the left, leading to the *detailed balance equation* [Whitt, 2006]:

$$\underbrace{\lambda_{i\to i-1}\pi[i]}_{\substack{\text{outflow of } i\\ \text{(to the left)}}} = \underbrace{\lambda_{i-1\to i}\pi[i-1]}_{\substack{\text{inflow towards } i\\ \text{(from the left)}}},$$

reordering leads to:

$$\pi[i] = \pi[i-1]\frac{\lambda_{i-1\to i}}{\lambda_{i\to i-1}}\,. \tag{7.2}$$

This equation can be solved easily in two steps. First, we compute the unnormalized equilibrium probability $\widehat{\pi}[i] \in \mathbb{R}^n_{\geqslant 0}$ in an iterative fashion (starting from state $i = 1$ and assuming $\widehat{\pi}[0] = 1$). Second, we normalize $\widehat{\pi}[i]$ to form a probability distribution:

$$\pi[i] = \frac{\widehat{\pi}[i]}{\sum_{j=0}^{n}\widehat{\pi}[j]}\,.$$

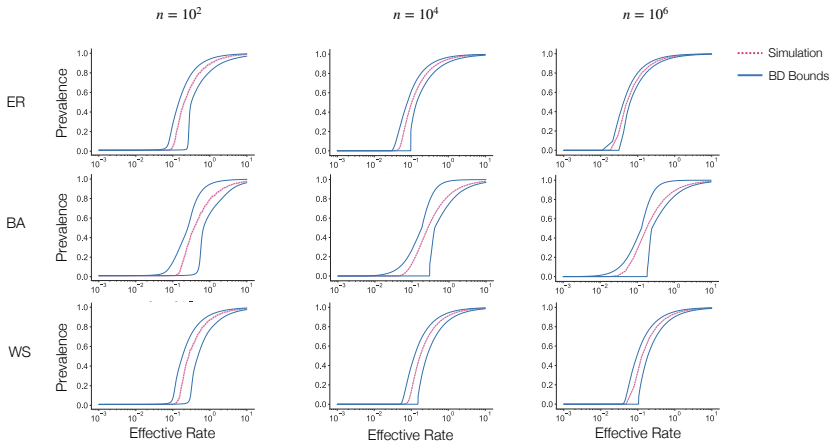


Fig. 7.4.: Results for: Erdős-Rényi, Barabási–Albert, Watts-Strogatz random graphs (top to bottom). Networks become larger and denser f.l.t.r.

Putting Things Together. Given a contact network, we first use MAGENTA to approximate the minimum (maximal) number of SI-edges (for any given number of infected nodes). Then, we construct $\mathrm{BD}_{\min}$ and $\mathrm{BD}_{\max}$ (which contain transitions that are parametrized by β). We use Eq. (7.2) to compute $\pi_{\min}$ and $\pi_{\max}$ for all relevant values of β and compute $\tau_{\min}(\beta)$ and $\tau_{\max}(\beta)$ analogously to Eq. (7.1).

7.3 Results

We tested many different synthetic contact networks. Specifically, we investigated how network size and density influence the quality of the approximated bounds. Results are shown in Figure 7.4. Generally speaking, we see that dense networks have much tighter bounds, but even for sparse networks, the bounds move closer to each other when the network size is increased.

7.4 Related Work

The presented reduction method can be seen as a form of state space lumping. We discuss lumping in more detail in the next chapter and refer the reader to the literature referenced therein. The reduction also highlights the importance of interaction topology in general. In particular, this is a nice illustration that the complexity of a contact network depends on the range of possible SI-edges. Consequently, contact networks, like the complete graph, where the number of infected nodes determine the number of SI-edges are simpler to handle analytically and computationally (as we already saw in the last chapter).

Particularly exciting applications of birth-death processes were introduced by Di Lauro et al. (2020) where a birth-death approximation was used to infer the underlying network type, and by Devriendt and Piet Van Mieghem (2017) where a birth-death approximation was used as a part of deriving mean-field approximation approaches.

8 From Networks to Population Models

This chapter investigates the relationship between spreading models and the class of *Markov Population Models* (MPMs). We demonstrate that lumping can be used to reduce any epidemic model to an MPM.

Therefore, we propose BLUE[1] (Partition-[b]ased [lu]mping m[e]thod), a novel lumping scheme based on a partitioning of the contact network.

Our **key idea** is to partition the contact network into regions and assume uniform connectivity within each region. The size of the regions gives us control over the resolution at which the Markov graph is analyzed. Conveniently, the reduced model falls into a well-studied class of MPMs for which already many approximation techniques exist.

8.1 Introduction

In the last chapter, we lumped (aggregated) network-states based on their corresponding number of infected nodes. Here, we consider a more general lumping scheme. For instance, instead of looking at the number of infected nodes in the entire contact network, we can partition the contact network and count the number of infected

[1] Code is available at `github.com/gerritgr/Reducing-Spreading-Processes`.

nodes in each partition. The more partitions we use, the better our approximation becomes.

To be more general, we first partition the network nodes. Second, we impose a *counting abstraction* on each partition. Last, we (only) lump two network-states together when their corresponding counting abstractions coincide on each partition.

Markov Population Models. As we will see, the counting abstraction induces a natural representation of the lumped CTMC as an MPM. In an MPM, the CTMC states are vectors that, for different types of species, count the number of entities of each species. The dynamics can be elegantly represented as species interactions. More importantly, a rich pool of approximation techniques has been developed based on MPMs, which can now be applied to the lumped model. These include efficient simulation techniques [Cao et al., 2006; G. E. Allen and Dytham, 2009; Sanft et al., 2011], dynamic state space truncation [Henzinger et al., 2009; Mateescu et al., 2010], moment-closure approximations [Soltani et al., 2015; Ramon Grima, 2012], linear noise approximation [Van Kampen, 1992; R. Grima, 2010], and hybrid approaches [Bortolussi, 2016; A. Singh and Hespanha, 2010].

8.2 Related Work

The general idea behind *lumping* is to reduce a system's complexity by aggregating the system's individual components. Lumping is a popular model reduction technique that has been used to reduce the number of equations in a system of ODEs and the number of states in a Markov chain, particularly in the context of biochemical reaction networks [Wei and Kuo, 1969; Cardelli et al., 2017; Ganguly et al.,

2014]. Generally speaking, one can distinguish between *exact* and *approximate* lumping [Li and Rabitz, 1990; Buchholz, 1994].

Most work on the lumpability of epidemic models has been done in the context of exact lumping [István Z Kiss et al., 2017; P. L. Simon et al., 2011; Ward and Evans, 2018]. The general idea is typically to reduce the state space by identifying symmetries in the CTMC, which can be found using symmetries (i.e., automorphisms) in the contact network. However, those methods are limited in scope because these symmetries are infeasible to find in real-world networks, and the state space reduction is insufficient to make realistic models tractable.

This work proposes an approximate lumping scheme. Approximate lumping has been shown to be useful when applied to mean-field approximation approaches of epidemic models such as the degree-based mean-field and pair approximation equations [Kyriakopoulos et al., 2018], as well as the approximate master equation [Großmann, Kyriakopoulos, et al., 2018]. However, mean-field equations are inflexible as they do not take topological properties into account or make unrealistic independence assumptions between neighboring nodes. Moreover, Petrov and Tognazzi (2021) use lumped model representations for stochastic simulation.

KhudaBukhsh et al. (2019) proposed using local symmetries in the contact network instead of automorphisms to construct a lumped Markov chain. This scheme seems promising, particularly on larger graphs, where automorphisms often do not exist. However, the limitations for real-world networks due to a limited amount of state space reduction and high computational costs seem to persist.

Conceptually similar to this work is also the *unified mean-field framework* (UMFF) proposed by Devriendt and Piet Van Mieghem (2017). Devriendt et al. also partition the nodes of the network but directly derive a mean-field equation from it. In contrast, this work focuses

on analyzing the lumped CTMC and its relation to MPMs. Moreover, we investigate different types of counting abstractions, not only node-based ones.

8.3 Our Method: BLUE

BLUE is composed of three basic ingredients:

Node Partitioning
The partitioning over the nodes $\mathcal{V}$ that the modeler provides.

Counting Pattern
The type of features that we count, i.e., nodes or edges.

Implicit State Space Partitioning
The CTMC state space is implicitly partitioned by counting nodes (or edges) using the node partitioning.

Example 8.1: BLUE's Ingredients

Jumping ahead, we can see a node partitioning in Figure 8.1a (p. 172) and the induced state space partitions as shaded areas in Figure 8.1b. Using node-based or edge-based counting abstractions results in different state space partitions.

Overview

We start by discussing the partitioning of the state space. Then we show how to obtain it from a given node partitioning and counting pattern. Consider a spreading model with an induced CTMC (cf. Section 2.5). Let $\mathcal{X}$ denote the state space of the CTMC (we call this the *original* CTMC). That is, elements $\mathbf{x} \in \mathcal{X}$ denote network-states. We

use $\mathcal{Y}$ to denote the new *lumped* state space. Their relationship is given by a surjective[2] lumping function

$$\mathcal{L} : \mathcal{X} \to \mathcal{Y} \,.$$

Two network-states $\mathbf{x}$, $\mathbf{x}'$ are lumped, iff $\mathcal{L}(\mathbf{x}) = \mathcal{L}(\mathbf{x}')$. Later in this section, we will construct $\mathcal{L}(\cdot)$ based on a node partitioning and a counting pattern of our choice.

We construct the transition rates $t(\mathbf{y}, \mathbf{y}')$ (where $\mathbf{y}, \mathbf{y}' \in \mathcal{Y}$, $\mathbf{y} \neq \mathbf{y}'$) between the states of the lumped Markov chain:

$$t(\mathbf{y}, \mathbf{y}') = \frac{1}{|\mathcal{L}^{-1}(\mathbf{y})|} \sum_{\mathbf{x} \in \mathcal{L}^{-1}(\mathbf{y})} \sum_{\mathbf{x}' \in \mathcal{L}^{-1}(\mathbf{y}')} q(\mathbf{x}, \mathbf{x}') \,. \tag{8.1}$$

This is simply the mean transition rate from a random network-state belonging to $\mathbf{y}$ to any network-state belonging to $\mathbf{y}'$.

Technically, Eq. (8.1) corresponds to the following *lumping assumption*:

We assume that, at each point in time, all network-states belonging to a lumped state $\mathbf{y}$ are equally likely.

8.3.1 Partition-Based Lumping

Next, we construct the lumping function $\mathcal{L}(\cdot)$. Because we want to make our lumping aware of the contact network's topology, we assume a given partitioning $\mathcal{P}$ over the nodes $\mathcal{V}$ of the contact network. That is,

$$\mathcal{P} \subset 2^{\mathcal{V}}, \; \bigcup_{P \in \mathcal{P}} P = \mathcal{V} \,, \text{ and all } P \in \mathcal{P} \text{ are disjoint and non-empty.}$$

[2] If $\mathcal{L}$ is not surjective, we enforce it by removing all elements from $\mathcal{Y}$ that have no counterpart in $\mathcal{X}$.

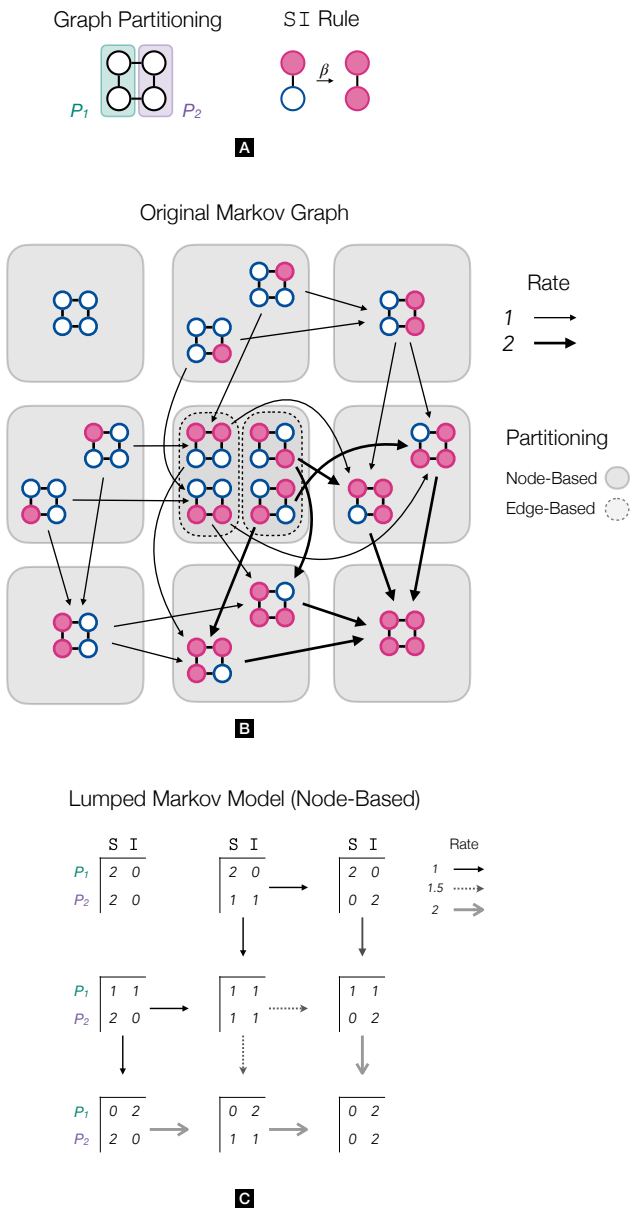


Fig. 8.1.: Simple example. **(a)**: We use a simple SI model with $\beta = 1$. The 4-node contact graph is divided into two partitions. **(b):** The original $2^4 = 16$ states. The graph partition induces the edge-based and node-based lumping. The edge-based lumping refines the node-based lumping and generates one partition more (center). **(c)**: The lumped CTMC using the node-based abstraction with only 9 states. The rates are the averages from the original CTMC.

We use n_P to denote the number of partitions and assume an implicit enumeration $\{P_1, \ldots, P_{n_P}\}$.

Based on the node partitioning, we can now impose different counting abstractions on the network-state. This work considers two types: counting nodes and counting edges. A full example of how a lumped CTMC of an SI model is constructed using the node-based counting abstraction is given in Figure 8.1 (p. 172). The counting abstractions are visualized on a larger contact network in Figure 8.2 (p. 174).

Possibility 1: Node-Based Counting

We count the number of nodes in each state and partition. Therefore, we use $n_S = |\mathcal{S}|$ to denote the number of node-states and $L(v_i)$ to denote the node-state of node v_i in a given network-state.

Hence, for a given network-state $\mathbf{x} \in \mathcal{X}$, we use a matrix to store the number of nodes in state $s \in \mathcal{S}$ for each partition $P \in \mathcal{P}$. The lumping function $\mathcal{L}(\cdot)$ projects $\mathbf{x}$ to the corresponding counting abstraction. Formally (we drop the node-index for clarity):

$$\begin{aligned} & \mathcal{Y} = \mathbb{Z}_{\geqslant 0}^{n_S \times n_P} \\ & \mathcal{L}(\mathbf{x}) = \mathbf{Y} \\ \text{with:} \quad & \mathbf{Y}[i,j] = \left| \left\{ v \in \mathcal{V} \;\middle|\; \mathbf{x}[v] = s_i, v \in P_j \right\} \right| . \end{aligned} \tag{8.2}$$

Here, $\mathbf{x}[v]$ denotes the node-state of v, given the network-state $\mathbf{x}$. Abusing notation in a familiar way, we might write $\mathbf{Y}[s, P]$ instead of $\mathbf{Y}[i, j]$ (where s is the i-th node-state and P is the j-th partition).

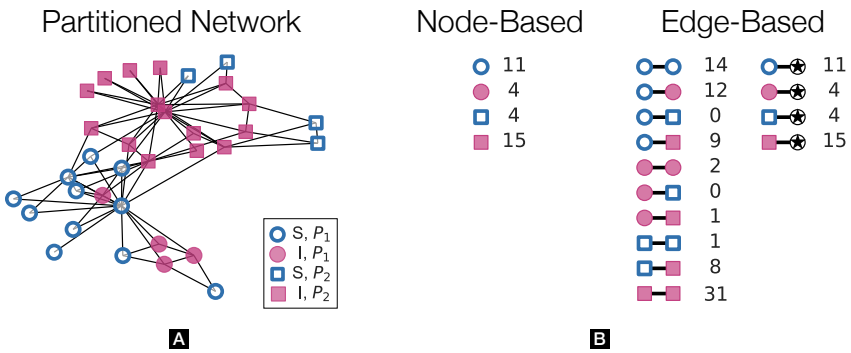


Fig. 8.2.: **(a)**: A partitioned network (Zachary's Karate Club graph from Girvan and M. E. Newman (2002)) (S: *blue*, I: *magenta, filled*). The network is partitioned into P_1 (○-nodes) and P_2 (□-nodes). **(b)**: The corresponding counting abstractions.

Possibility 2: Edge-Based Counting

Instead of counting nodes for each partition, we now count edges. We store the counts in a 4-dimensional counting tensor. Thus,

$$\mathcal{Y} = \mathbb{Z}_{\geqslant 0}^{n_S \times n_P \times n_S \times n_P}$$
$$\mathcal{L}(\mathbf{x}) = \mathbf{Y}$$
$$\text{with: } \mathbf{Y}[s, P, s', P'] = \left|\left\{(v, v') \in \mathcal{E} \;\middle|\; \mathbf{x}[v] = s, v \in P, \mathbf{x}[v'] = s', v' \in P'\right\}\right|$$

Intuitively, for each pair of node-states $s, s' \in \mathcal{S}$ and pair of partitions $P, P' \in \mathcal{P}$, we count $\mathbf{Y}[s, P, s', P']$, which is the number of edges from nodes of state s in one partition to nodes in state s' in the other partition. This includes cases where $P = P'$ and $s = s'$.

However, only counting the edges does not determine how many nodes there are in each state (cf. Figure 8.3, p. 175). To encode this information, we modify the network structure by adding a new dummy node $v_\star$ and connecting it to each node. The dummy node

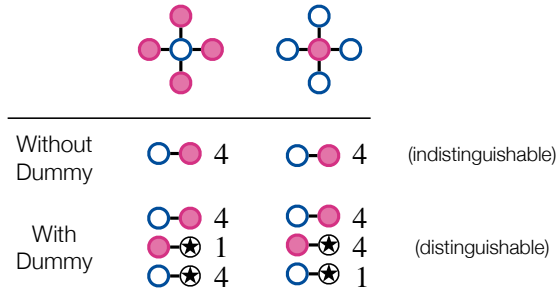


Fig. 8.3.: By adding the dummy-node, the edge-based abstraction can differentiate the two graphs. Adding the dummy-node ensures that the nodes in each state are counted in the edge-based abstraction.

has a dummy state denoted by $\star$, which never changes, and it can be assigned to a new dummy partition $\mathrm{P}_\star$. Formally,

$$\mathcal{V} \coloneqq \mathcal{V} \cup \{v_\star\} \quad \mathcal{S} \coloneqq \mathcal{S} \cup \{\star\} \quad \mathrm{L}(v_\star) = \star \quad \mathcal{P} \coloneqq \mathcal{P} \cup \{\mathrm{P}_\star\}$$
$$\mathcal{E} \coloneqq \mathcal{E} \cup \{(v, v_\star) \mid v \in \mathcal{V}, v \neq v_\star\}\,.$$

The dummy node does not influence the transition rates. Figure 8.1 (p. 172) illustrates how a given partitioning and the node-based counting approach induce a lumped CTMC. We illustrate an SI model (an SIS model where $\alpha = 0$). The partitions induced by the edge-based counting abstracting are also shown. In this example, the edge-based lumping aggregates only isomorphic network-states.

8.3.2 Graph Partitioning

Broadly speaking, we have three options to partition nodes: according to (i) local features (e.g., its degree), (ii) global features (e.g., communities in the graph), or (iii) randomly. As a baseline, we use random node partitioning. Therefore, we fix the number of partitions and randomly assign each node to a partition while enforcing that all partitions contain roughly the same number of elements.

Moreover, we investigate a degree-based partitioning, where we define the distance between nodes v, v' as their relative degree difference (similar to Kyriakopoulos et al. (2018)):

$$d_k(v, v') = \frac{|k_v - k_{v'}|}{\max(k_v, k_{v'})} .$$

We can then use any reasonable clustering algorithm and build partitions (i.e., clusters) with the distance function. In this work, we focus on bottom-up hierarchical clustering as it provides the most principled way of precisely controlling the number of partitions. As always (in particular, to avoid infinite distances), we only consider contact networks where each node is reachable from every other node. We break ties arbitrarily.

To get a clustering considering global features, we use a spectral embedding of the contract network. Specifically, we use the `spectral_layout` function from the `NetworkX` Python-package [Hagberg, Swart, et al., 2008] with three dimensions and perform hierarchical clustering on the embedding. In future research, it would be interesting to compute node distances based on more sophisticated graph embedding. Note that in the border cases $|\mathcal{P}| = 1$ and $|\mathcal{P}| = n$ all methods yield the same partitioning.

8.4 Markov Population Models

Next, we explain MPMs. Later, we illustrate the construction of MPMs given lumped CTMCs.

Species. MPMs are a special form of CTMCs where each CTMC state is a population vector over a set of species. We use $\mathcal{Z}$ to denote the finite set of species (again, with an implicit enumeration) and $\mathbf{y} \in \mathbb{Z}_{\geqslant 0}^{|\mathcal{Z}|}$ to denote the population vector. We use $\mathbf{y}[z]$ to identify the population counts of species $z \in \mathcal{Z}$.

Dynamics. In spreading processes, we used a set of rules to specify the dynamics. Similarly, in MPMs, we use a set of *reactions* $\mathcal{A}$. Each reaction, $(\alpha(\cdot), \mathbf{b}) \in \mathcal{A}$, is comprised of a propensity function $\alpha : \mathbb{Z}_{\geqslant 0}^{|\mathcal{Z}|} \to \mathbb{R}_{\geqslant 0}$ and a change vector $\mathbf{b} \in \mathbb{Z}^{|\mathcal{Z}|}$. When reaction $(\alpha, \mathbf{b})$ is applied, the system moves from state $\mathbf{y}$ to state $\mathbf{y} + \mathbf{b}$. The propensity function gives the corresponding rate.

CTMC Semantics. The transition rates of the induced CTMC are given as[3]:

$$t(\mathbf{y}, \mathbf{y}') = \begin{cases} \alpha(\mathbf{y}) & \text{if } \exists (\alpha, \mathbf{b}) \in \mathcal{A}, \mathbf{y}' = \mathbf{y} + \mathbf{b} \\ 0 & \text{otherwise} \end{cases} .$$

Next, we show that the counting abstractions from the previous section have a natural interpretation as MPMs.

[3] Without loss of generality, we assume that different reactions have different change vectors. If this is not the case, we can merge reactions with the same update by summing their corresponding rate functions.

8.4.1 Node-Based Abstraction

Species. We use the species to encode the entries in the counting matrix. This gives rise to the set of species:

$$\mathcal{Z} = \{(s, P) \mid s \in \mathcal{S}, P \in \mathcal{P}\}\,.$$

Again, we assume an implicit enumeration of $\mathcal{Z}$. We use $z.s$ and $z.P$ to denote the components of a given species z. Note hat representing the system state as a population count vector $\mathbf{y} \in \mathbb{Z}_{\geqslant 0}^{n_S \cdot n_P}$ instead of a matrix $\mathbf{Y} \in \mathbb{Z}_{\geqslant 0}^{n_S \times n_P}$ is simply a matter of re-ordering.

Reactions. Next, we express the dynamics by translating rules into reactions. For each rule $r = s_1 \xrightarrow{f(\cdot)} s_2$ and each partition $P \in \mathcal{P}$, we define a reaction $(\alpha_{r,P}(\cdot), \mathbf{b}_{r,P})$ with propensity function as:

$$\alpha_{r,P} : \mathbb{Z}_{\geqslant 0}^{n_S \times n_P} \to \mathbb{R}_{\geqslant 0}$$

$$\alpha_{r,P}(\mathbf{y}) = \frac{1}{|\mathcal{L}^{-1}(\mathbf{y})|} \underbrace{\sum_{\mathbf{x} \in \mathcal{L}^{-1}(\mathbf{y})}}_{\substack{\text{all aggregated} \\ \text{network-states}}} \underbrace{\sum_{v \in P}}_{\substack{\text{all nodes in} \\ \text{the partition}}} f(\mathbf{m}_{\mathbf{x},v}) \underbrace{\mathbb{1}_{\mathbf{x}[v]=s_1}}_{\substack{\text{Consider only if} \\ \text{node-state is } s_1}}\,,$$

where $\mathbf{m}_{\mathbf{x},v}$ denotes the neighborhood vector of node v in network-state $\mathbf{x}$ and $\mathbf{x}[v] \in \mathcal{S}$ is the node-state of v in $\mathbf{x}$. Note that this equation is just the instantiation of Eq. (8.1).

The change vector $\mathbf{b}_{r,P} \in \mathbb{Z}^{|\mathcal{Z}|}$ is defined element-wise as:

$$\mathbf{b}_{r,P}[z] = \begin{cases} 1 & \text{if } z.s = s_2, P = z.P \\ -1 & \text{if } z.s = s_1, P = z.P \\ 0 & \text{otherwise.} \end{cases}$$

Note that s_1, s_2 refer to the current rule and $z.s$ to the entry of $\mathbf{b}_{r,P}$.

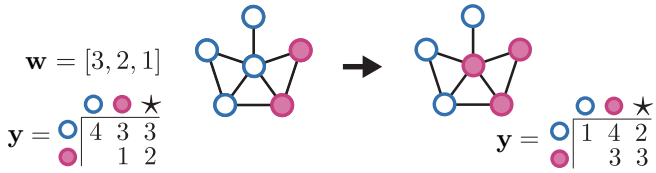


Fig. 8.4.: Example of how the neighborhood $\mathbf{w}$ influences the update in the edge-based counting abstraction on an example graph. Here, all nodes belong to the same partition and the node-states are ordered $[\mathtt{S}, \mathtt{I}, \star]$. The population vector $\mathbf{y}$ is given in matrix form for ease of presentation. The center node changes from $\mathtt{S}$ to $\mathtt{I}$. Therefore, we subtract $\mathbf{w}$ from the first line in $\mathbf{y}$ and add it to the second line.

8.4.2 Edge-Based Counting Abstraction

Representing the edge-based abstraction in terms of population counts and reactions is more technical but follows the same principle.

Species. Again, we use the species to encode the entries in the counting tensor. However, the 4D tensor $\mathbf{Y}$ contains symmetries. To get rid of them, we assume an arbitrary ordering of pairs $(s, p) \in \mathcal{S} \times \mathcal{P}$ and construct the set of species as:

$$\mathcal{Z} = \big\{ (s_{src}, P_{src}, s_{tgt}, P_{tgt}) \,|\, s_{src}, s_{tgt} \in \mathcal{S}, P_{src}, P_{tgt} \in \mathcal{P}, \\ (s_{src}, P_{src}) \leqslant (s_{tgt}, P_{tgt}) \big\} \,.$$

Again, we assume an arbitrary ordering of species and find that $\mathbf{y}$ is just a flattened (non-redundant) representation of the counting tensor. We use $\mathbf{y}[z]$ to denote the population number of species $z \in \mathcal{Z}$.

Reactions. Next, we need to establish some technical details. We start by defining a *(*sP*-)species neighborhood*. The species neighbor-

hood of a node v is a vector $\mathbf{w}_v \in \mathbb{Z}_{\geqslant 0}^{n_S \cdot n_P}$, where $\mathbf{w}_v[s, P]$ denotes[4] the number of neighbors in node-state s and partition P. We define $\mathcal{W}_v$ as the set of possible species neighborhoods for a node v, given a fixed contact network and partitioning. Note that we still assume that a dummy node is used to encode the number of states in each partition.

Let us define $\mathcal{W}_P$ to be the set of all possible species neighborhoods the nodes in P can have:

$$\mathcal{W}_P = \bigcup_{v \in P} \mathcal{W}_v \,.$$

For each rule $r = s_1 \xrightarrow{f(\cdot)} s_2$, and each partition $P \in \mathcal{P}$, and each $\mathbf{w} \in \mathcal{W}_P$, we define a reaction $(\alpha_{r,P,\mathbf{w}}(\cdot), \mathbf{b}_{r,P,\mathbf{w}})$, where:

$$\alpha_{r,P,\mathbf{w}} : \mathbb{Z}_{\geqslant 0}^{|Z|} \to \mathbb{R}_{\geqslant 0}$$
$$\alpha_{r,P,\mathbf{w}}(\mathbf{y}) = \frac{1}{\mathcal{L}^{-1}(\mathbf{y})} \sum_{\mathbf{x} \in \mathcal{L}^{-1}(\mathbf{y})} \sum_{v \in P} f(\mathbf{m}_{\mathbf{x},v}) \underbrace{\mathbb{1}_{\mathbf{x}[v]=s_1, \mathbf{w}_v=\mathbf{w}}}_{\substack{\text{Consider nodes in } s_1 \text{ that} \\ \text{have a sp. neighborhood} \\ \text{matching } \alpha}} .$$

Counterintuitively, the propensity is individually defined for each possible $\mathbf{w}$, but the propensity values do not take $\mathbf{w}$ into account. This is because the change vector depends on $\mathbf{w}$. To see this, consider a species $z = (s_{src}, P_{src}, s_{tgt}, P_{tgt})$, corresponding to edges connecting a node in state s_{src} and partition P_{src} to a node in state s_{tgt} and partition P_{tgt}. There are two scenarios in which the corresponding counting variable has to change: (a) when the node changes its state due to an application of rule r is the source node and (b) when it is the target node. Consider case (a): we need to know how many

[4] We continue our notational pattern and write $\mathbf{w}_v[s, P]$ instead of $\mathbf{w}_v[i]$ where i is the index of (s, P), assuming an implicit enumeration over $\mathcal{S} \times \mathcal{P}$.

edges are connecting the updated node (which was in state s_1 and partition P) to a node in state s_{tgt} and partition P_{tgt}. This information is stored in the vector $\mathbf{w}$, specifically in position $\mathbf{w}[s_{tgt}, P_{tgt}]$. The case in which the updated node is the target node is treated symmetrically. This gives rise to the following definition:

$$\mathbf{b}_{r,P,\mathbf{w}}[z] = \begin{cases} \mathbf{w}[z.s_{tgt}, z.P_{tgt}] & \text{if } s_2 = z.s_{src}, P = z.P_{src} \\ -\mathbf{w}[z.s_{tgt}, z.P_{tgt}] & \text{if } s_1 = z.s_{src}, P = z.P_{src} \\ \mathbf{w}[z.s_{src}, z.P_{src}] & \text{if } s_2 = z.s_{tgt}, P = z.P_{tgt} \\ -\mathbf{w}[z.s_{src}, z.P_{src}] & \text{if } s_1 = z.s_{tgt}, P = z.P_{tgt} \\ 0 & \text{otherwise.} \end{cases}$$

The first two lines of the definition handle cases where the node changing state is the source node, while the following two lines deal with the case in which the node changing state appears as the target. Figure 8.4 (p. 179) illustrates how a lumped network-state is influenced by the application of an infection rule.

8.4.3 Direct Construction of the MPM

Approximating the solution of a spreading process on a contact network by lumping the CTMC first already reduces the computational costs by many orders of magnitude. However, this scheme is still only applicable when it is possible to construct the full CTMC in the first place. Recall that the number of network-states is exponential in the number of nodes of the contact network. That is, $|\mathcal{X}| = |\mathcal{S}|^{|\mathcal{V}|} = n_S^n$.

However, in recent years, substantial effort has been dedicated to the analysis of very small networks [Ward and Evans, 2018; Holme, 2015b; Moslonka-Lefebvre et al., 2009; Pautasso et al., 2010]. One reason is that stochastic effects are more dominant in small networks.

However, it would still be great if we could construct the reduced MPM without building the original CTMC first. Luckily, this can be done for the node counting abstraction exactly. Unfortunately, for the edge counting, we need to introduce an additional approximation in the definition of the rate function. Roughly speaking, we introduce an approximate probability distribution over neighboring vectors, as knowing how many nodes have a specific neighboring vector requires full knowledge of the original CTMC. We present full details of such direct construction in Appendix A.1 and illustrate the relationship to chemical reaction networks (CRNs).

8.4.4 MPM Complexity

The size of the lumped MPM is critical for our method, as it determines which solution techniques are computationally tractable and provides guidelines on how many partitions to choose. There are two notions of size to consider: (a) the number of population variables and (b) the number of states of the reduced CTMC. The latter governs the applicability of numerical solutions that build the reduced state space. The former controls the complexity of many approximation techniques, like mean-field methods or moment closure.

Node-Based Abstraction

The population vector is of length $|\mathcal{S}| \cdot |\mathcal{P}|$, i.e., there is a variable for each node-state and each partition. Note that the sum over the population variables for each partition P is $|\mathrm{P}|$, the number of nodes in the partition. This allows us to count the number of states of the CTMC of the population model easily: For each partition, we need to subdivide $|\mathrm{P}|$ different nodes into $|\mathcal{S}|$ different classes, which can be done in $\binom{|\mathrm{P}|+|\mathcal{S}|-1}{|\mathcal{S}|-1}$ ways, giving a number of CTMC states that

is exponential in $|\mathcal{S}|$ and $|\mathcal{P}|$, but only polynomial in the number of nodes:

$$|\mathcal{Y}| = \prod_{P \in \mathcal{P}} \binom{|P| + |\mathcal{S}| - 1}{|\mathcal{S}| - 1}.$$

Edge-Based Abstraction

There is one population variable for each edge-type connecting two different partitions and additional population variables accounting for the dummy state. In total, we have $\frac{q(q-1)}{2} + q$ population variables, with $q = |\mathcal{S}| \cdot |\mathcal{P}|$.

In order to count the number of CTMC states, we start by observing that the sum of all variables for a given pair of partitions P', P'' is the number of edges connecting such partitions in the graph. We use $\epsilon(P', P'')$ to denote the number of edges between P', P'' (resp. the number of edges inside P' if $P' = P''$). Thus,

$$|\mathcal{Y}| \leqslant \prod_{\substack{P', P'' \in \mathcal{P}^2 \\ P' \leqslant P''}} \binom{\epsilon(P', P'') + \mathcal{S}^2 - 1}{\mathcal{S}^2 - 1} \cdot \prod_{P \in \mathcal{P}} \binom{|P| + |\mathcal{S}| - 1}{|\mathcal{S}| - 1}.$$

This is an over-approximation because not all combinations are consistent with the graph topology. For example, a high number of infected nodes in a partition might not be consistent with a small number of `II`-edges inside the partition. Note that this upper bound is exponential in $|\mathcal{S}|$ and $|\mathcal{P}|$, but still only polynomial in the number of nodes $n_{\mathcal{V}}$.

The exponential dependency on the number of species (i.e., dimensions of the population vector) makes the explicit construction of the lumped state space feasible only for small networks with a small number of node-states. However, this is typically the case for spreading models like `SIS` or `SIR`. Yet, also the number of partitions has

to be kept small. We expect that the partitioning is especially useful for networks showing a small number of large-scale homogeneous structures, as it is the case for many real-world networks [Girvan and M. E. Newman, 2002].

An alternative strategy for analysis is to derive mean-field [Bortolussi, Hillston, et al., 2013] or moment closure equations [Schnoerr et al., 2018] for MPMs, which can be done without explicitly constructing the lumped (and the original) state space. These are sets of ODEs describing the evolution of (moments of) the population variables. We refer the reader to Devriendt and Piet Van Mieghem (2017) for a similar approach regarding the node-based abstraction.

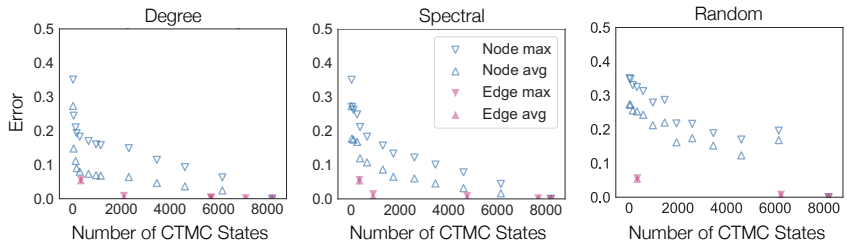


Fig. 8.5.: [Lower is better.] Trade-off between accuracy and state space size for the node-based (blue) and edge-based (magenta, filled) counting abstraction. Results are shown for node partitions based on the degree (l.), spectral embedding (c.), and random partitioning (r.). The accuracy is measured as the mean ($\triangle$) and maximal ($\triangledown$) difference between the original and lumped solution over all timepoints.

8.5 Numerical Results

In this section, we compare the numerical solution of the original model—referred to as baseline model—with different lumped MPMs. This comparison aims to provide evidence supporting the claim that the lumping preserves the dynamics of the original system, with accuracy increasing with the resolution of the MPM.

Setup. We perform the comparison by solving the original and the lumped system numerically and comparing each state's probability at each timepoint (cf. Excursus 2). Because we solve the original CTMC, the size of the contact network (and number of node-states) is strongly limited.

Let $P(X(t) = \mathbf{x})$ denote the probability that the original CTMC occupies network-state $\mathbf{x} \in \mathcal{X}$ at time t. Let $P(Y(t) = \mathbf{y})$, for $\mathbf{y} \in \mathcal{Y}$, denote the same probability for a lumped MPM. To measure their difference, we approximate the probability distribution of the original model from the lumped solution. Hence, we invoke the lumping assumption, which states that all network-states that are lumped together

have the same probability mass. We use $P_L(\cdot)$ to denote the *lifted* probability distribution over the original state space given a lumped solution. Formally,

$$P_L\big(Y(t) = \mathbf{x}\big) = \frac{P\big(Y(t) = \mathbf{y}\big)}{|\mathcal{L}^{-1}(\mathbf{y})|} \quad \text{where } \mathbf{y} \text{ is s.t. } \mathcal{L}(\mathbf{x}) = \mathbf{y}.$$

We measure the difference between the original and a lumped solution at a specific time point by summing up the difference in probability mass of each state, then take the maximum error in time:

$$d(P, P_L) = \max_t \sum_{\mathbf{x} \in \mathcal{X}} \left| P_L\big(Y(t) = \mathbf{x}\big) - P\big(X(t) = \mathbf{x}\big) \right| .$$

In our experiments, we used a small toy network with 13 nodes and 2 states ($2^{13} = 8192$ network-states). We generated a synthetic contact network following the Erdős–Rényi graph model with a connection probability of 0.5. We use an `SIS` model with an infection rate of $\beta = 1.0$ and a recovery rate of $\alpha = 1.3$. Initially, we assign an equal amount of probability mass to all network-states.

Discussion. Figure 8.5 (p. 185) shows the relationship between the error of the lumped MPM, the type of counting abstraction, and the method used for node partitioning. We also report the mean difference together with the maximal difference over time.

From our results, we conclude that the edge-based counting abstraction yields a significantly better trade-off between state space size and accuracy. However, it generates larger MPM models than the node-based abstraction when adding a new partition. We also find that spectral and degree-based partitioning yield similar results for the same number of CTMC states. Random partitioning performed

noticeably worse for both edge-based and node-based counting abstractions. For larger networks with actual components, we expect that the spectral-based method performs significantly better.

8.6 Conclusions and Future Work

This work developed the first steps in unifying the analysis of stochastic spreading processes on networks and Markov population models. Since the obtained MPM can become very large in terms of its species, it is crucial to be able to control the trade-off between state space size and accuracy.

However, there are still many open research problems to be solved. Most evidently, it remains to be determined which of the many techniques developed for the analysis of MPMs (e.g., linear noise, moment closure) work best on our proposed epidemic-type MPMs and how they scale with increasing size of the contact network. We also expect that these reduction methods can provide a good starting point for deriving advanced mean-field equations, similar to the ones in Devriendt and Piet Van Mieghem (2017). Moreover, the literature is very rich in proposed moment-closure-based approximation techniques for MPMs, which can now be utilized [Soltani et al., 2015; Ramon Grima, 2012]. Finally, we expect that there are many more possibilities of partitioning the contact network that remain to be investigated, which might have a significant impact on the final accuracy of the abstraction.

9 Neural Relational Inference

In this chapter, we explore the problem of inferring (reconstructing) the unknown contact network (or interaction graph) of a complex dynamical system from node-level observations (the nodes represent agents or components). We consider a setting where the underlying dynamical model is unknown and where different measurements (i.e., *snapshots*) may be independent (e.g., may stem from different experiments).

Our **key idea** is to utilize the concept of *masked reconstruction* for graph inference: We mask (erase) node-states and hypothesize that the true interaction graph provides the best basis to recover the node-states from the states of adjacent vertices.

We propose TEAL[1] (Ne[t]work R[e]construction [Al]gorithm), a graph neural network to simultaneously learn the latent interaction graph and, conditioned on the interaction graph, the prediction of a node's state based on its immediate neighborhood.

9.1 Introduction

Stochastic dynamical systems in which local interactions give rise to complex emerging phenomena are ubiquitous in nature and society (cf. Chapter 2). However, their analysis remains challenging.

[1] PyTorch code is available at `github.com/GerritGr/GINA`.

Inferring the functional organization of a complex system from measurements is relevant for its analysis [Fornito, Zalesky, and Breakspear, 2015; Prakash, Vreeken, et al., 2012; Amini et al., 2016; Finn et al., 2019], design [Zitnik et al., 2018; Hagberg and Schult, 2008; Memmesheimer and Timme, 2006], control [Gu et al., 2015; Großmann, Backenköhler, Klesen, et al., 2020], and prediction [Kipf et al., 2018; Z. Zhang et al., 2019].

In this chapter, we focus on the internal interaction structure (i.e., graph or network) of a complex system. We propose a machine learning approach to infer this structure based on observational data from the nodes (i.e., components or constituent agents). We refer to these observations as *snapshots* and assume that the observable states of all components are measured simultaneously. However, we make no prior assumption about the relationship between snapshots. Specifically, snapshots are not labeled with time stamps. They may be taken from different experiments with varying initial conditions (see Figure 9.1, p. 191, for an overview).

For instance, in the `SIS` model, a snapshot could assign each node to a disease stage (infected or susceptible), and the goal would be to reconstruct the unknown set of edges, $\mathcal{E}$, from many different (uncorrelated) snapshots. However, by doing so, we would not know that the `SIS` model was used to generate the data in the first place.

Most recent work on graph inference focuses on time series data, where observations are time-correlated and the interaction graph is inferred from the joint time evolution of the node-states [Z. Zhang et al., 2019; Kipf et al., 2018]. Naturally, time series data contains more information on the system's internal interactions than snapshot data. However, in many cases, such data is not available: In some cases, one has to destroy a system to access its components (e.g., slice a brain [Rossini et al., 2019], observe a quantum system [Martinez et al., 2019], or terminate a cell [Chan et al., 2017]). Sometimes, the relevant time scale of the system is too small (e.g., in particle physics)

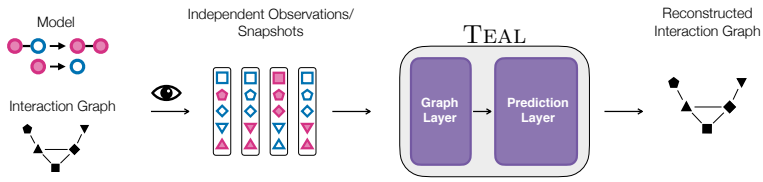


Fig. 9.1.: Schematic illustration of the problem setting. We are interested in inferring the latent (unweighted and undirected) interaction graph from observational data of the process.

or too large (e.g., in evolutionary dynamics) to be observed. Often, there is a trade-off between the spatial and temporal resolution of a measurement [Sarraf and J. Sun, 2016]. Finally, measurements may be temporally decoupled due to large observation intervals and thus become unsuitable for methods that exploit correlations in time. It is also known that, generally speaking, many different graph typologies can be used to predict the same dynamics [Prasse and Piet Van Mieghem, 2020b]. Yet, machine learning techniques for graph inference from independent data remain underexplored in the literature.

In contrast to many state-of-the-art approaches, our method is model-free and makes no prior assumptions about the dynamical laws. Conceptually, our analysis is based on identifying patterns within the snapshots. These patterns are spatial manifestations of the temporal co-evolution of neighboring nodes. Thus, they carry information about the underlying connectivity.

We propose an approach inspired by ideas recently popularized for time series-based network inference [H.-F. Zhang et al., 2018; Kipf et al., 2018]. It provides an elegant way of formalizing *graph inference problem* with minimal parametric assumptions on the underlying dynamical model:

The key hypothesis is that the interaction graph "best describing" the observed data is the ground truth.

In the time series setting, we can operationalize "best describing" by assuming that it provides the best grounds for time series forecasting.

This chapter assumes that time series data is not available. Hence, we aim to find the graph that best enables us to predict a node's state based on its immediate neighborhood within a snapshot (not at future times). This technique is commonly referred to as *masking* [Mishra et al., 2020]. That is, we *mask* (i.e., erase) the state of a node and then try to recover it by looking at its neighbors. To this end, we use a prediction model to learn a node's observable state (i.e., the *node-state*) given the joint state of all adjacent nodes. Then we maximize the prediction accuracy by jointly optimizing the interaction graph and the prediction model:

We assume that the information shared between a node and its complete neighborhood is higher in the ground truth graph than in other potential graphs.

However, in a trivial implementation, the network that enables the best node-state prediction is the complete graph, because it provides all information available in a given snapshot. This necessitates a form of regularization in which edges that are present—but not necessary—reduce prediction quality. We do this using a counting-based neighborhood aggregation scheme that acts as a bottleneck of information flow.

Therefore, we use a simple counting abstraction over the neighborhood, corresponding to a sum-aggregation in a graph neural network (GNN) architecture. Thus, this task can be framed as a simple single-layer GNN combined with a node-specific multi-layer perceptron (MLP) that predicts node-states from neighborhood aggregations. The node-states can directly relate to the measurements or be the result of an embedding or a discretization.

For an efficient solution to the *graph inference problem*, we propose TEAL.

TEAL tackles this problem by simultaneously learning the interaction graph and the dynamics. The approach employs a computationally simple neighborhood aggregation and an efficient weight-sharing mechanism between nodes. Combined with a differentiable graph representation, our method can be applied to systems with hundreds of nodes. TEAL is model-free (it can learn arbitrary patterns in snapshots) and threshold-free (no arbitrary edge binarization is needed afterward).

While we make only minimal parametric assumptions about the dynamical process itself, we assume that the model can be expressed (or approximated) using the class of multi-state processes on graphs (cf. Section 2.4.3). This means that a node only directly communicates with its immediate neighborhood and that the underlying graph is homogeneous. That is, all nodes are equivalent and can only be differentiated by their corresponding node-state or node attribute.

In this contribution, we conceptualize and test the hypothesis that network reconstruction can be formulated as a prediction task. Specifically:

1. We propose a masking technique to formalize the prediction problem;

2. We derive a suitable neighborhood aggregation mechanism that automatically acts as a regularization mechanism;

3. We develop the neural architecture TEAL to efficiently solve the prediction and reconstruction task;

4. We test our hypothesis using synthetically generated snapshots using various combinations of graphs and diffusion models.

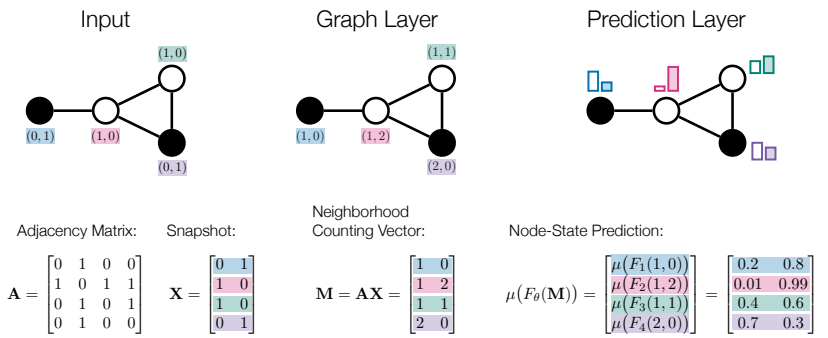


Fig. 9.2.: Schematic architecture using 4-node graph with $\mathcal{S} = \{(0,1),(1,0)\}$. Nodes are color-coded, node-states are indicated by the shape (filled: I, blank: S). First, we compute $\mathbf{m}_i$ for each node v_i (stored as $\mathbf{M}$), then we feed each $\mathbf{m}_i$ into a predictor that predicts the original state.

9.2 Foundations and Problem Formulation

The goal is to find the latent interaction graph of a complex system with n agents/nodes.

Notational Refresher

We follow the notation from Chapter 2. That is, a graph is represented as an adjacency matrix $\mathbf{A}$ of size $n \times n$ (with node set $\{v_i \mid 1 \leqslant i \leqslant n\}$). An entry $a_{ij} \in \{0,1\}$ indicates the presence ($a_{ij} = 1$) or absence ($a_{ij} = 0$) of an edge between node v_i and v_j. We assume that $\mathbf{A}$ is symmetric (the graph is undirected), and the diagonal entries are all zero (the graph has no self-loops). We use $\mathbf{A}^*$ to denote the ground truth matrix.

Each snapshot (similarly to a graph labeling) assigns a node-state to each node. The finite set of possible node-states is denoted $\mathcal{S}$. For convenience, we assume that the node-states are represented using one-hot encodings. For instance, in an epidemic model, a

node might be susceptible or infected. Since there are two node-states, we use $\mathcal{S} = \{(0,1),(1,0)\}$. Each snapshot $\mathbf{X} \in \{0,1\}^{n \times |\mathcal{S}|}$ can then conveniently be represented as a matrix with n rows, each row describing the corresponding one-hot encoded node-state (cf. Figure 9.2, p. 194). We use $\mathcal{X}$ to denote the set of independent snapshots. We make no specific assumption about the underlying distribution or process behind the snapshots or their relationship to one another. For a node v_i (and fixed snapshot), we use $\mathbf{m}_i \in \mathbb{Z}_{\geqslant 0}^{|\mathcal{S}|}$ to denote the (element-wise) sum of all neighboring node-states, referred to as *neighborhood counting vector* (cf. Section 2.4.3). Let's return to our previous example, where $\mathcal{S} = \{(1,0),(0,1)\}$. The state $\mathbf{m}_i = (10,13)$ tells us that node v_i has 10 susceptible ($(1,0)$) and 13 infected ($(0,1)$) neighbors. The set of all possible neighborhood counting vectors is denoted by $\mathcal{M} \subset \mathbb{Z}_{\geqslant 0}^{|\mathcal{S}|}$.

Idea

Assuming we know the adjacency matrix $\mathbf{A}$ and have a given snapshot $\mathbf{X}$. We can then compute the neighborhood counting vectors of all nodes using $\mathbf{M} = \mathbf{AX}$, where the i-th row of $\mathbf{M}$ equals $\mathbf{m}_i$ (cf. Figure 9.2, p. 194, center).

In the next step, we feed each counting vector $\mathbf{m}_i$ into a machine learning model that predicts (resp. recovers) the original state of v_i in that snapshot. Specifically, for each node v_i, we learn a function:

$$F_{\mathbf{W}_i} : \mathcal{M} \to \texttt{Cat}(\mathcal{S}) \,,$$

where $\texttt{Cat}(\mathcal{S})$ denotes the set of all probability distributions over $\mathcal{S}$ (in the sequel, we make the mapping to a probability distribution explicit by adding a $\texttt{Softmax}(\cdot)$ function). To evaluate $F_{\mathbf{W}_i}(\cdot)$, we use some loss function to quantify how well the distribution predicts the true state and minimize this *prediction loss*. We assume that $F_{\mathbf{W}_i}(\cdot)$

is fully parameterized by a node-dependent parameter matrix $\mathbf{W}_i$ (e.g., in a NN, $\mathbf{W}_i$ contains the weights and biases of all layers). The weights for a set of vertices are given by the list $\mathcal{W} = \{\mathbf{W}_1, \ldots, \mathbf{W}_n\}$. We define $\mathsf{F}_{\mathcal{W}}(\cdot)$ as a node-wise application of $\mathsf{F}_{\mathbf{W}_i}(\cdot)$, that is,

$$\mathsf{F}_{\mathcal{W}}(\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_n) = \big(\mathsf{F}_{\mathbf{W}_1}(\mathbf{m}_1), \mathsf{F}_{\mathbf{W}_2}(\mathbf{m}_2), \ldots, \mathsf{F}_{\mathbf{W}_n}(\mathbf{m}_n)\big) \ .$$

The hypothesis is that the ground truth adjacency matrix $\mathbf{A}^*$ provides the best foundation for $\mathsf{F}_{\mathbf{W}_i}$, ultimately leading to the smallest prediction loss. Under this hypothesis, we can use the loss as a surrogate for the accuracy of candidate $\mathbf{A}$ (compared to the ground truth graph).

9.2.1 Graph Neural Network

Next, we formulate the *graph inference problem* using a very elementary graph neural network (GNN), denoted $\mathrm{NN}(\cdot)$, that loosely resembles an autoencoder architecture: For each snapshot $\mathbf{X}$, we predict the node-state of each node using only the neighborhood of that node. Then, we compare the prediction with the actual (known) node-state.

For a given adjacency matrix (graph) $\mathbf{A} \in \{0,1\}^{n \times n}$ and list of weight matrices $\mathcal{W}$, we define the GNN $\mathrm{NN}(\cdot)$, applied to a snapshot $\mathbf{X} \in \{0,1\}^{n \times |\mathcal{S}|}$ as:

$$\begin{aligned} \mathrm{NN}_{\mathcal{W},\mathbf{A}} &: \{0,1\}^{n \times |\mathcal{S}|} \to \mathbb{R}^{n \times |\mathcal{S}|} \\ \mathrm{NN}_{\mathcal{W},\mathbf{A}} &: \mathbf{X} \mapsto \texttt{Softmax}\big(\mathsf{F}_{\mathcal{W}}(\mathbf{A}\mathbf{X})\big) \ , \end{aligned}$$

where $\texttt{Softmax}(\cdot)$ is applied row-wise. Thus, $\mathrm{NN}_{\mathcal{W},\mathbf{A}}(\mathbf{X})$ results in a matrix where each row corresponds to a node and models a distribution over node-states. Like in the auto-encoder paradigm, input and output are of the same form, and the network learns to minimize the difference (note that the one-hot encoding can also be viewed

as a valid probability distribution over node-states). The absence of self-loops in $\mathbf{A}$ is critical as it means that the actual node-state of a node is not part of its own neighborhood aggregation. As we want to predict a node's state, the state itself cannot be part of the input. We say the node itself is *masked*.

We will refer to the matrix multiplication $\mathbf{AX}$ as *graph layer* and to the application of $\texttt{Softmax}(\mathrm{F}_{\mathcal{W}}(\cdot))$ as *prediction layer*. We only perform a single application of the graph layer on purpose, which means that only information from the immediate neighborhood can be used to predict a node-state. While using n-hop ($n > 1$) neighborhoods would increase the network's predictive power, it would be detrimental to graph reconstruction.

Most modern GNN architectures follow the message-passing scheme, where each layer performs an *aggregate*$(\cdot)$ and a *combine*$(\cdot)$ step. The *aggregate*$(\cdot)$ step computes a neighborhood embedding based on a permutation-invariant function of all neighboring nodes. The *combine*$(\cdot)$ step combines this embedding with the actual node-state. In our architecture, aggregation is the element-wise sum. The combination, however, needs to purposely ignore the node-state (in order for the prediction task to make sense) and applies $\texttt{Softmax}(\mathrm{F}_{\mathcal{W}}(\cdot))$. Thus, our method can be seen as the application of a single GNN-layer.

9.2.2 Prediction Loss

We assume a loss function[2] $\mathrm{L}(\cdot)$ that is applied independently to each snapshot:

$$\mathrm{L} : \{0,1\}^{n\times|\mathcal{S}|} \times \mathbb{R}^{n\times|\mathcal{S}|} \to \mathbb{R}$$

[2] A labeling function, $\mathrm{L}(\cdot)$, does not exist in this chapter, so no confusion should arise.

The prediction loss $L(\cdot)$ compares the input (actual node-states) and output (predicted node-states) of the $NN(\cdot)$. We define the loss on a set of independent snapshots $\mathcal{X}$ as the sum over all constituent snapshots:

$$L\big(\mathcal{X}, NN_{\mathcal{W},\mathbf{A}}(\mathcal{X})\big) \coloneqq \sum_{\mathbf{X}\in\mathcal{X}} L\big(\mathbf{X}, NN_{\mathcal{W},\mathbf{A}}(\mathbf{X})\big) \ .$$

In our experiments, we use row-wise MSE-loss.

Note that, in the above sum, all snapshots are treated equally independent of their corresponding initial conditions or time points at which they were made (which we do not know anyway). Formally, this is reflected in the fact that the loss function is invariant with respect to the snapshot order.

9.2.3 Graph Inference Problem

We define the *graph inference problem* for an interacting system with n components as follows:

For given set of snapshots $\mathcal{X} = \{\mathbf{X}_1, \ldots, \mathbf{X}_m\}$, find an adjacency matrix $\mathbf{A}' \in \{0,1\}^{n\times n}$ and NN-parameterization $\mathcal{W}'$ minimizing the prediction loss:

$$(\mathcal{W}', \mathbf{A}') \coloneqq \operatorname{argmin}_{\mathcal{W},\mathbf{A}} L\big(\mathcal{X}, NN_{\mathcal{W},\mathbf{A}}(\mathcal{X})\big) \ .$$

Thus, solving the graph inference problem requires simultaneously optimizing over a discrete space (graphs) and a continuous space (NN-weights). The subsequent section explains how to achieve this by relaxing the discrete space.

Note that, in general, we cannot guarantee that $\mathbf{A}'$ is equal to the ground truth matrix $\mathbf{A}^*$. Prasse and Piet Van Mieghem (2018) show

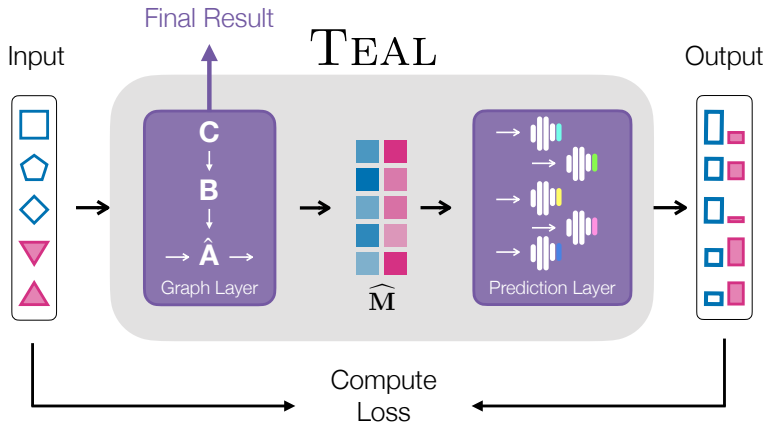


Fig. 9.3.: Illustration of TEAL. Snapshots are processed independently. Each input/snapshot associates each node with a node state (blue, pink). The output is a distribution over states for each node. During training, this distribution is optimized w.r.t the input. The output is computed based on a multiplication with the current adjacency matrix candidate (stored as $\mathbf{C}$) and the application of a node-wise MLP. Ultimately, we are interested in a binarized version of the adjacency matrix. Color/filling indicates the state; shape identifies nodes.

that network reconstruction for epidemic models based on time series data is $\mathcal{NP}$-hard when formulated as a decision problem. We believe this carries over to our setting but leave a proof for future work.

9.3 Our Method: TEAL

As explained in the previous section, it is infeasible to solve the graph inference problem by iterating over all possible graphs/weights. Hence, we propose TEAL to search the vast space of possible graphs (and weights) efficiently. TEAL approximates the graph inference problem by jointly optimizing the graph $\mathbf{A}$ and the prediction layer weights $\mathcal{W}$.

Therefore, we adopt two tricks: First, we impose a relaxation on the graph adjacency matrix representing its entries as real-valued numbers. Second, we use shared weights in the weight matrices belonging to different nodes. Specifically, each node v_i gets its custom MLP, but weights of all layers, except the last one, are shared among nodes. This allows us to simultaneously optimize the graph and the weights using back-propagation. Apart from that, we still follow the architecture from the previous section. That is, a graph layer maps a snapshot to neighborhood counting vectors, and each neighborhood counting vector is pushed through a node-wise MLP.

9.3.1 Graph Layer

Internally, we store the interaction graph as an upper triangular matrix $\mathbf{C}$. In each step, we (i) compute $\mathbf{B} = \mathbf{C} + \mathbf{C}^\top$ to enforce symmetry, (ii) compute $\mathbf{A}' = \mu(\mathbf{B})$ to enforce a reasonable range, and (iii) set all diagonal entries of $\mathbf{A}'$ to zero, yielding $\widehat{\mathbf{A}} = \text{mask}(\mathbf{A}')$.

Here, $\mu(\cdot)$ is a differential function that is applied element-wise and maps real-valued entries to the interval $[0, 1]$. It ensures that $\widehat{\mathbf{A}}$ approximately behaves like a binary adjacency matrix while remaining differentiable (the hat-notation indicates relaxation). Specifically, we use a nested Sigmoid-type function $f(\cdot)$ that is parametrized by a sharpness parameter ν:

$$\begin{aligned} &\mu : \mathbb{R} \to [0, 1] \\ &\mu : x \mapsto f\big(\big(f(x) - 0.5\big) \cdot \nu\big) \ , \end{aligned}$$

where we choose $f(x) = 1/(1 + \exp(-x))$ and increase the sharpness (by increasing ν) over the course of the training. Note that the masking is done implicitly if we ensure that diagonal entries in $\mathbf{C}$ are zero and $\mu(0) = 0$.

Finally, the graph layer matrix is multiplied with the snapshot, which yields a relaxed version of the neighborhood counting abstractions. In summary, for a snapshot $\mathbf{X}$, the graph layer computes:

$$\hat{\mathbf{M}} = \hat{\mathbf{A}}\mathbf{X} = \text{mask}\big(\mu(\mathbf{C} + \mathbf{C}^\top)\big)\mathbf{X}\,,$$

where $\mathbf{C}$ is optimized during training.

9.3.2 Prediction Layer

In $\hat{\mathbf{M}}$, each row corresponds to one node. Thus, we apply the MLPs independently to each row. We use $\hat{\mathbf{m}}_i$ to denote the row corresponding to node v_i (i.e., the neighborhood counting relaxation). Let $\text{FC}_{i,o}$ denote a fully-connected (i.e., linear) layer with input (resp. output) dimension i (resp. o). We use `ReLu` and `Softmax` activation functions. The whole prediction layer MLP contains four sub-layers and is given as:

$$\begin{aligned}
o_i^1 &= \texttt{ReLu}\big(\text{FC}_{|\mathcal{S}|,10}(\hat{\mathbf{m}}_i)\big)\\
o_i^2 &= \texttt{ReLu}\big(\text{FC}_{10,10}(o^1)\big)\\
o_i^3 &= \texttt{Softmax}\big(\text{FC}_{10,|\mathcal{S}|}(o^2)\big)\\
o_i^4 &= \texttt{Softmax}\big(\text{FC}_{|\mathcal{S}|,|\mathcal{S}|}(o^3)\big)\,.
\end{aligned}$$

In our implementation, only the last sub-layer (o_i^4) contains node-specific weights. This enables a node-specific shift of the probability computed in the previous layer. All other weights are shared among nodes which results in strong regularization. In summary, $\text{F}_{\mathbf{W}_i}(\cdot)$ applies the MLP of node v_i and $\mathbf{W}_i$ contains the weights and biases from all layers j in o_i^j.

Note that we use a comparably small dimension (i.e., 10) for internal embeddings, which has shown to be sufficient in our experiments.

The node-specific weights lead to a slight but consistent improvement of the graph reconstruction. All node-specific weights can be updated efficiently in parallel in a single forward-backward pass.

9.3.3 Training

We empirically find that over-fitting is not a problem and, therefore, do not use a test set. However, a natural approach would be to split the snapshots into a training and test set and optimize $\hat{\mathbf{A}}$ and $\mathcal{W}$ on the training set until the loss reaches a minimum on the test set. Another important aspect during training is the usage of mini-batches. For ease of notation, we have ignored batches so far. In practice, mini-batches of snapshots are crucial for fast and robust training. A mini-batch of size $\mathfrak{b}$ can be created by concatenating $\mathfrak{b}$ snapshots (in the graph layer) and re-ordering the rows accordingly (in the prediction layer).

9.3.4 Limitations

There are some relevant limitations to TEAL. First, we cannot guarantee that the ground truth graph is actually the solution to the *graph inference problem*. In particular, simple patterns in the time domain (that enable trivial graph inference using time series data) might correspond to highly non-linear patterns inside a single snapshot. Secondly, TEAL is only applicable if statistical associations among adjacent nodes manifest themselves in a way that renders the counting abstraction meaningful. Statistical methods are more robust in the way they can handle different types of pair-wise interactions but less powerful regarding non-linear combined effects of the complete neighborhood. Another relevant design decision is to use one-hot encoding, which renders the forward pass extremely fast but will

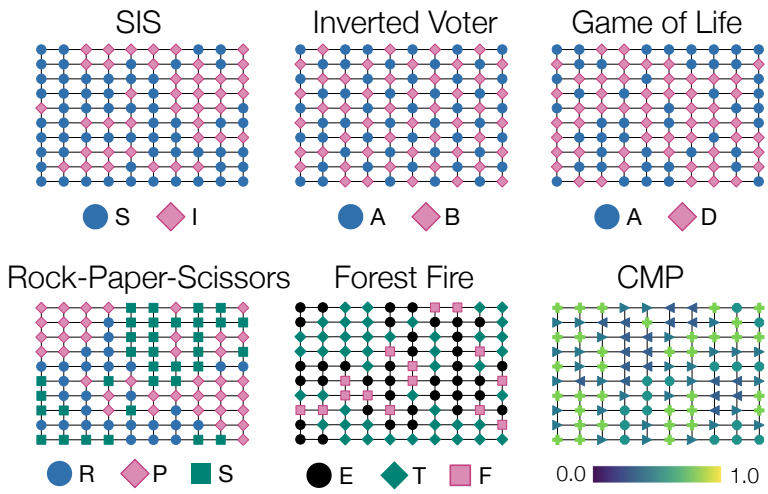


Fig. 9.4.: Examples of typical equilibrium snapshots on a 10×10 grid graph. Different dynamics give rise to different types of cluster formations.

reach limitations when the node-state domain becomes very complex. Lastly, together with relational homogeneity, we also assume that all agents behave reasonably similarly, enabling weight sharing, increasing training efficiency, and reducing the number of required samples.

9.4 Testing TEAL

We conduct four experiments using synthetically generated snapshots. In **Experiment 1**, we analyze the underlying hypothesis that the ground truth graph enables the best node-state prediction. In **Experiment 2**, we study the importance of sample size for the reconstruction accuracy, and in **Experiment 3**, we compare TEAL to statistical baselines. Finally, in **Experiment 4**, we compare TEAL with machine learning baselines on time series data.

Setup. Our prototype of TEAL is implemented using PyTorch [Paszke et al., 2019] and is executed on a standard desktop computer with 32 GB of RAM and an Intel i9-10850K CPU.

Accuracy and Loss. We quantify the performance of TEAL using the *graph loss* and the *prediction loss*. The *graph loss* measures the quality of an inferred graph. It is defined as the L_1 (Manhattan) distance of the upper triangular parts of the two adjacency matrices (i.e., the number of edges to add/remove). We always use a binarized version of inferred graph $\widehat{\mathbf{C}}$ for comparison with the ground truth $\mathbf{A}^*$. The *prediction loss* measures how well TEAL predicts masked node-states and follows the definition in Section 9.2.2. All results are based on a single run of TEAL. Performing multiple runs and using the result with the lowest prediction loss might further improve TEAL's performance. For more details on the architecture and hyperparameters of TEAL, we refer the reader to Appendix B.1.

Dynamical Models

We study six models. A precise description of the dynamics and parameters is provided in Appendix B.2. We focus on stochastic processes, since probabilistic decisions and interactions are essential for modeling uncertainty in real-world systems. The models include a simple SIS-epidemic model where infected nodes can randomly infect susceptible neighbors or become susceptible again. In this model, susceptible nodes tend to be close to other susceptible nodes and vice versa. This renders network reconstruction comparably simple. In contrast, we also propose an Inverted Voter model (*InvVoter*) where nodes tend to maximize their disagreement with their neighborhood (influenced by the original Voter model by Campbell et al. (1954)). Nodes have one of two opinions (A or B), and nodes in A tend to move to B faster the higher their number of A neighbors

and vice versa. For investigating even more complex emerging dynamics, we study a system loosely inspired by Conway's *Game of Life*. Nodes (cells) are either dead (`D`) or alive (`A`). Living conditions are good (i.e., nodes tend to stay alive or be born) when roughly half of a node's neighbors are alive. Likewise, they tend to die (or stay dead) when the neighborhood is highly unbalanced. That is, almost all neighboring cells are either dead (underpopulation) or alive (overpopulation). We also examine a rock-paper-scissors (`RPS`) model to study evolutionary dynamics [Szabó and Fath, 2007] and the well-known *Forest Fire* model [Bak et al., 1990] where a node (spot) can be empty (`E`), occupied by a tree (`T`), or occupied by fire (`F`) induced by stochastic lightning. Finally, we test a deterministic discrete-time dynamical model: a coupled map lattice model (*CML*) [Garcia et al., 2002; Kaneko, 1992; Z. Zhang et al., 2019] to study graph inference in the presence of chaotic behavior. As the CML model admits real node-values in $[0, 1]$, we performed discretization into 10 equally-spaced bins.

For the stochastic models, we use numerical simulations to sample from the equilibrium distribution of the systems. For CML, we randomly sample an initial state and simulate it for a random period. Note that we do not explicitly add measurement errors, but all nodes are subject to internal noise (i.e., they spontaneously flip with a small probability).

Figure 9.4 (p. 203) provides visualizations of typical equilibrium samples from the dynamical models.

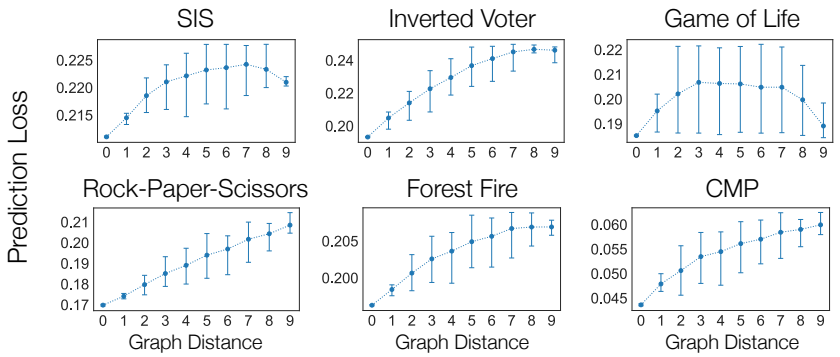


Fig. 9.5.: **Exp. 1:** [Lower is better.] Computing the loss landscape based on all possible 5-node graphs. x-axis: Graph distance to ground truth adjacency matrix. y-axis: Mean prediction loss of corresponding graph candidates. Error bars denote min/max-loss.

9.4.1 Experiment 1: Loss Landscape

For this experiment, we generated 5000 snapshots for all dynamical models on the so-called *bull graph* (as illustrated in Figure 9.1, p. 191). We then trained TEAL and measured the prediction loss for all potential 5×5 adjacency matrices representing connected graphs. Note that the ground truth graph has a graph distance of zero. During training, we fixed the graph layer and only optimized the prediction layer. We observe a large dependency between the prediction loss of a candidate graph and the corresponding graph loss. We conclude that the hypothesis that graphs closer to the ground truth yield a better predictive performance is reasonable. The *Game of Life* dynamical model provides the only example of graph candidates that allow a better prediction than the ground truth graph (by a minimal amount). Interestingly, this is one of the graph candidates furthest from the ground truth.

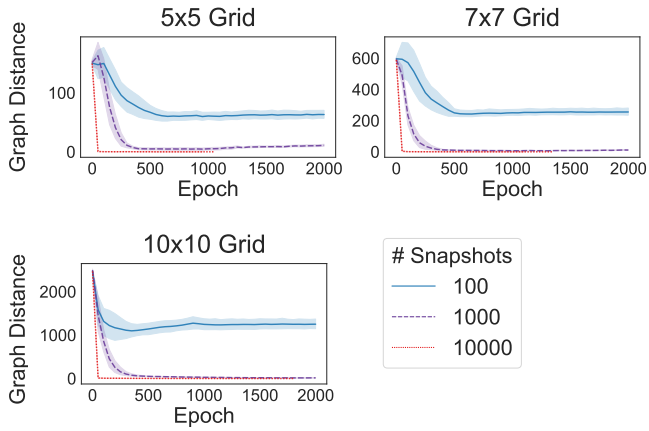


Fig. 9.6.: **Exp. 2:** [Lower is better.] Influence of snapshot number on training convergence (SIS-dynamics). x-axis: Epoch number. y-axis: Graph distance between ground truth graph and the state of TEAL. Clearly, a higher sample size yields better performance. 95% CIs are based on ten independent training runs.

9.4.2 Experiment 2: Sample Size

We tested the effect of the number of snapshots on the graph loss using SIS-dynamics. For this experiment, we used a $L \times L$ grid graph with $L \in \{5, 7, 10\}$ and compared results based on $10^2, 10^3, 10^4$ snapshots. It can be clearly seen that the higher the number of snapshots, the faster the training procedure converges. In the case of 10^4 snapshots, TEAL already converges after a few epochs. This is a natural consequence of each epoch containing more forward passes. More interesting is that a larger sample size yields a better graph loss (reconstruction accuracy). It shows that TEAL can, in fact, absorb the additional information provided by the entirety of the training data. Specifically, 100 snapshots do not contain enough information to approximately solve the graph inference problem. Results are shown in Figure 9.6.

Tab. 9.1.: **Exp. 3**: [Lower is better.] Results of different graph inference methods.

		Graph Loss				Runtime (sec)			
Model	Graph	TEAL	Corr	MI	ParCorr	TEAL	Corr	MI	ParCorr
SIS	ER	19	**0**	**0**	**0**	890	< 1	< 1	8
	Geom	**50**	88	141	54	1366	< 1	2	126
	Grid	12	**0**	**0**	**0**	2131	< 1	9	1862
	WS	**0**	**0**	**0**	**0**	109	< 1	1	22
InvVoter	ER	**1**	66	24	66	96	< 1	< 1	8
	Geom	**0**	556	38	556	167	< 1	2	125
	Grid	**0**	360	90	360	754	< 1	10	1861
	WS	**0**	138	2	138	110	< 1	1	21
Game of Life	ER	44	48	**20**	54	599	< 1	< 1	7
	Geom	**0**	554	104	556	152	< 1	2	132
	Grid	**10**	360	20	360	2181	< 1	10	1901
	WS	**0**	138	22	138	111	< 1	1	21
RPS	ER	**0**	**0**	**0**	**0**	114	< 1	< 1	8
	Geom	**1**	76	74	2	317	< 1	3	129
	Grid	72	**0**	**0**	**0**	2445	< 1	10	1873
	WS	**0**	4	4	**0**	128	< 1	1	21
Forest Fire	ER	13	**0**	**0**	6	1030	< 1	< 1	8
	Geom	**19**	320	130	326	1486	< 1	2	127
	Grid	30	**0**	**0**	**0**	2520	< 1	9	1892
	WS	**0**	2	**0**	4	131	< 1	1	22
CML	ER	**0**	**0**	4	**0**	156	< 1	< 1	7
	Geom	**0**	**0**	46	2	316	< 1	3	125
	Grid	8	**0**	**0**	**0**	5569	< 1	11	1874
	WS	**0**	**0**	4	**0**	192	< 1	< 1	22

9.4.3 Experiment 3: Independent Snapshots

Next, we compare TEAL with statistical baselines.

Ground Truth Graphs. To generate ground truth graphs we use Erdős-Renyi (ER) ($N = 22$), Geometric (Geom) ($N = 50$), and Watts–Strogatz (WS) ($N = 30$). Moreover, we use a 2D-grid graph with 10×10 nodes ($N = 100$, $|E| = 180$). We use 50 thousand samples. Graphs were generated the using *networkX* package [Hagberg, Swart, et al., 2008] (cf. Appendix B.3 for details).

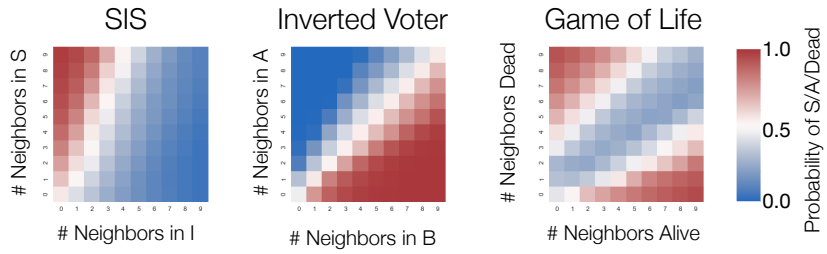


Fig. 9.7.: Output of the prediction layer for a random node in the Watts–Strogatz network. We map neighborhood counting vectors to the probability of the node being in state `S` (`SIS`), `A` (InvVoter), or `D` (Game of Life).

Baselines. As statistical baselines, we use the Python package *netrd* [Hartle et al., 2020]. Specifically, we use the correlation (Corr), mutual information (MI), and partial correlation (ParCorr) methods. The baselines only return weighted matrices. Hence, they need to be binarized using a threshold. To find the optimal threshold, we provide them with the number of edges of the ground truth graph. Notably, especially in sparse graphs, this leads to unfair advantage and renders the results not directly comparable. Furthermore, *netrd* only accepts binary or real-valued node-states. This is a problem for the categorical models `RPS` and FF. As a solution, we simply map the three node-states to real values (1, 2, 3), breaking statistical convention. Interestingly, the baselines handle this well and, in most cases, identify the ground truth graph nevertheless. Results are shown in Table 9.1 (p. 208).

Prediction Layer Visualization. We can visualize the prediction layer for the 2-state models. It encodes the conditional probability to be in a specific node-state given the 2-dimensional neighborhood counting vector $\mathbf{m}_i$. Figure 9.7 illustrates this conditional probability as a function of each possible neighborhood counting vector. The results are given for a Watts–Strogatz graph (where each node has approximately degree 4). The prediction layer belonging to the same node

was used for all three models. We observe that the prediction layer finds conditional probability distributions that capture the specific characteristics of the dynamical models. It also generalizes well (beyond degree 4).

9.4.4 Experiment 4: Time Series Data

In contrast to approaches for network inference based on time series data, TEAL can be used when there is no (known) temporal relationship between snapshots. However, we can still apply TEAL when time series data is available. In this experiment, we generate a single trajectory of a stochastic process and observe it at an interval x ($x \in \{1, 5, 10, 20\}$), e.g., $x = 1$ means that we observe the process after every jump in the underlying stochastic process. The reason to test different intervals between observations is that time series analysis methods are, in principle, sensitive to the time resolution of the observations. If too many (or too few) nodes change their state from one observation to another, this could hinder these approaches' capability of finding and exploiting temporal patterns. In practice, we, however, observe little dependence on the temporal resolution.

Baseline. We compare TEAL with the previous statistical baselines and with the machine learning approaches Automated Interactions and Dynamics Discovery (`AIDD`) [Yan Zhang et al., 2021] and Gumbel Graph Network (`GGN`) [Z. Zhang et al., 2019] which are both general frameworks to infer the network structure based on time series data (cf. Section 9.5 for more details).

Setup. We used 6000 samples (fewer were not possible without further adapting the `GGN` code) and a simple 5×5 grid graph (larger graphs made the `GGN` application too expensive). We used binary

Tab. 9.2.: Exp. 4: [Lower is better.] Results of different graph inference methods on time series data.

		Graph Loss						Runtime (sec)		
Model	Interval	TEAL	AIDD	GGN	Corr	MI	ParCorr	TEAL	AIDD	GGN
SIS	1	8	42	179	4	4	**2**	62	2163	4382
	5	**0**	28	179	**0**	**0**	**0**	62	2318	4396
	10	1	29	85	**0**	**0**	**0**	62	2192	4432
	20	1	36	83	**0**	**0**	**0**	62	2222	4402
InvVoter	1	**0**	18	73	80	22	80	44	2224	4358
	5	**0**	28	31	80	16	80	37	2318	4395
	10	**0**	24	66	80	16	80	37	2181	4432
	20	**0**	27	80	80	16	80	37	2172	4419
GoL	1	**13**	42	199	80	52	80	62	2172	4443
	5	**4**	42	208	80	22	80	62	2166	4353
	10	**4**	40	190	80	26	80	61	2224	4354
	20	**9**	42	163	80	22	80	62	2218	4391

state dynamics because we could only apply the baseline code off-the-shelf to this format. We trained AIDD for 400 epochs and GGN for 40 epochs (we found that accuracy would not improve after that). For comparison, we made the results of the baselines symmetric and binary.

Results. Generally, we find that TEAL outperforms the methods based on time series analysis (AIDD and GGN). This is surprising, as, in principle, the temporal data should contain significantly more information on the connectivity than the individual snapshots. We can only speculate why GGN performs poorly, but we hypothesize that the method is conceptually unsuited for stochastic dynamics of jump processes where only a single agent changes at a time (and not all agents at once). Less surprisingly, we find that TEAL is many orders of magnitude faster than AIDD and GGN (ca. 40 to 70 times) while still processing many more epochs (up to 5000 in TEAL vs 400 and 40 in AIDD and GGN, respectively).

Moreover, we find that TEAL performs slightly better than the statistical methods. Interestingly, Mutual Information is the only baseline that performs well on all dynamical models. Corr and ParCorr fail on the *InvVoter* and *GoL* model (this happens consistently even when increasing the number of snapshots). Using automated thresholding (instead of providing *netrd* with the number of edges) prevents the "collapse" of the statistical methods but leads to a larger graph loss in general. Detailed results are shown in Table 9.2 (p. 211).

9.4.5 Discussion

The results show that graph inference based on independent snapshots is possible and that TEAL is a viable alternative to statistical methods. Compared to the baselines, TEAL yielded the highest accuracy in most of the cases, although we gave the (statistical) baseline methods the advantage of knowing the ground truth number of edges. TEAL performed particularly well in challenging cases where neighboring nodes do not tend to be in the same (or in similar) node-states. TEAL even performed acceptably in the case of CML dynamics despite the discretization of the data and the chaotic and deterministic nature of the process.

9.5 Related Work

Literature abounds with methods to infer the (latent) functional organization of complex systems that is often expressed using (potentially weighted, directed, temporal, multi-) graphs.

Most relevant for this work are previous approaches that use deep learning on time series data to learn the interaction dynamics and the interaction structure. Zhang et al. designed a two-component GNN architecture where a graph generator network proposes interaction graphs and a dynamics learner learns to predict the dynamical evolution using the interaction graph [Z. Zhang et al., 2019; Yan Zhang et al., 2021]. Both components are trained alternately. Similarly, Kipf et al. (2018) learn the dynamics using an encoder-decoder architecture that is constrained by an interaction graph which is optimized simultaneously. Huang et al. (2020) use a compression-based method for this purpose. Another state-of-the-art approach for this problem, based on regression analysis instead of deep learning, is the ARNI framework by Casadiego et al. (2017). However, this method requires time series data and hinges on a good choice of basis functions.

Other methods to infer interaction structures aim at specific dynamical models and applications. Examples include epidemic contagion [M. E. Newman, 2018; Di Lauro et al., 2020; Prasse and Piet Van Mieghem, 2020a], gene networks [Kishan et al., 2019; Omranian et al., 2016], functional brain network organization [Abril et al., 2018], and protein-protein interactions [Hashemifar et al., 2018]. In contrast, our approach assumes no prior knowledge about the laws that govern the system's (co-)evolution.

Statistical methods provide an equally viable and often very robust alternative. These can be based on partial correlation, mutual information, or graphical lasso [Tibshirani, 1996; Friedman et al., 2008].

Here, we not only rely on pairwise correlations among components but also on the joint impact of all neighboring components, which is necessary in the presence of non-linear dynamical laws governing the system. Moreover, we directly infer binary (i.e., unweighted) graphs in order to not rely on (often unprincipled) threshold mechanisms.

Our method is also related to self-supervised machine learning, particularly masking. Masking, in the form of masked language modeling, was popularized for pre-training transformer-based models like BERT, proposed by Devlin et al. (2018). Masking image patches also results in state-of-the-art pre-training for image recognition [J. Chen et al., 2022].

Likewise, node-attribute masking was successfully used as a GNN pre-training technique by W. Hu et al. (2019) and to improve the ability of a network to generalize by Mishra et al. (2020). This work uses masking to establish the general optimization objective (not for pre-training).

Another relevant research area is optimizing discrete structures (like graphs). While traditional methods use gradient-free techniques such as greedy optimization [Netrapalli et al., 2010], genetic [Barman and Kwon, 2018], or memetic [K. Wu et al., 2019] algorithms, SGD-based approaches have gained popularity. For instance, Paulus et al. (2020) apply the Gumble-softmax-trick. Fu et al., 2020 use iterative refinement using a differential (GNN-based) loss function, and Bengio et al., 2021 directly predict a sample from a distribution over discrete objects that is implicitly specified by a reward signal.

9.6 Conclusions and Future Work

We proposed TEAL, a model-free and threshold-free approach to infer the underlying graph structure of a dynamical system from independent observations. TEAL is based on the principle that local interactions among agents manifest themselves in specific local patterns. These patterns can be found and exploited.

More generally, this study confirms that the underlying hypothesis—that the ground truth graph best describes a set of snapshots—gives rise to a promising graph inference paradigm. We also show that node-attribute masking is a principled and practical approach to operationalizing and measuring what it means to "best describe" the observational data. For small graphs, we demonstrated this by enumerating the whole search space (**Experiment 1**). For larger graphs, we demonstrated this by showing that TEAL beats baselines in many cases (**Experiment 3** and **4**).

TEAL is only one possibility to explore the vast space of all possible graphs by utilizing a relaxation of the adjacency matrix. This makes the problem amenable to gradient-based methods. Other methods (e.g., based on genetic algorithms or Gumbel-softmax-based optimization) are also possible and worth exploring. We believe that the main challenge for future work is to find ways of inferring graphs when the types of interaction differ considerably among edges. Moreover, a deeper theoretical understanding of which processes lead to meaningful statistical associations, not only over time but also within snapshots, would be desirable.

Part IV

Concluding Remarks

10 Conclusion

Network science and computational modeling of epidemics are both exciting research topics in their own. However, exploring emerging phenomena when spreading is constrained by a network topology reveals compelling universalities and enables an intriguing unification of different research areas. Properties of a network—like hubs and small-worldness—are the drivers of emergent dynamical patterns.

This thesis contributed to two relevant techniques for the computational analysis of spreading dynamics on networks: simulation and model reduction. We also presented applications, like vaccine distribution, and—using Covid-19 as an example—discussed pitfalls when applying such models to the real world.

Simulation methods and model-reduction techniques can be used to distill high-level properties from model specifications. Simulation-based methods are easy to apply and very flexible. This thesis contributed to the scope of simulation-based methods by reducing their asymptotic run-time and increasing their flexibility to a new model class of non-Markovian dynamics. Model-reduction approaches provide the modeler with more responsibility. Ideally, they offer precise control over the trade-off between accuracy and computational costs. This is particularly true for the translation method to Markov Population Models developed in this thesis. They can also be used when the summary statistics obtained from simulation runs are not the central area of interest. We also showed how to use a simple reduction to simultaneously study a whole family of models (relating to different rates).

An intriguing property of spreading dynamics is the emergence of spatial correlations that are manifestations of the joint time evolution of the nodes. This thesis showed that neighboring nodes form patterns that are so characteristic and informative that they can be found and used to infer the underlying network structure.

One theme that constantly accompanied us was identifying the correct level of abstraction. This challenge is at the heart of all modeling, and networks are no exception. While adding complexity (weights, temporal features, and node-level variety) to networks is generally straightforward, it is easy to misjudge the implications. We extended the standard SIS model, for instance, in order to capture the progression of a Covid-19 disease more accurately and to study the effects of heterogeneity in a society. However, complexity is not an end in itself. Careful consideration must be given to the question whether the complexity of the model is useful or whether, in the worst case, it could obscure important relationships.

All this becomes even more relevant when mathematical models are applied to the real world. The Covid-19 pandemic has revealed to us that the applicability and interpretation of computational models are not an abstract issue but carry real-world responsibilities. Unfortunately, the Covid-19 pandemic was not a shining moment for mathematical modeling, but brought many problems to light. This includes improper use of modeling techniques without discussing their limitations or communicating their underlying assumptions. It became abundantly clear that simulations can never fully capture the diversity of the natural world. Statements derived from models should always be scrutinized for their (implicit) assumptions and the quality of their parameters.

10.1 Future Work

Spreading on networks is far from being solved. While specific perspectives are given in each chapter separately, we want to take this opportunity to provide a broader overview. On a technical level, finding high-level (potentially semi-quantitative) descriptions of the emerging dynamics is still an open problem. Global summary statistics are undoubtedly helpful in many cases, but generating a more informative representation would also be intriguing. Visualizations that indicate dynamic pathways of an infection or modes in the underlying distribution in networks with millions of nodes could provide new insights, especially when they can be linked to graph-theoretical properties.

On a more general level, the messiness of real-world data still hinders many tools. This is especially true for the lack of actual connectivity data. Often, only small subsets of the contact network are known, and providing robust analysis despite this is an open problem.

Another exciting research area is the study of adaptive networks, where the feedback loop between network connectivity and dynamics is investigated. Most work of this thesis could be re-thought in these terms and may contribute to discovering surprising relationships and patterns.

Bibliography

Abril, Ildefons Magrans de, Junichiro Yoshimoto, and Kenji Doya (2018). "Connectivity inference from neural recording data: Challenges, mathematical bases and research directions". In: *Neural Networks* 102, pp. 120–137 (cit. on p. 213).

Adam, David (2020). "Special report: The simulations driving the world's response to COVID-19." In: *Nature* 580.7803, p. 316 (cit. on p. 122).

Adam, Dillon, Peng Wu, Jessica Wong, et al. (2020). "Clustering and superspreading potential of severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2) infections in Hong Kong". In: *PREPRINT (Version 1) available at Research Square* (cit. on p. 123).

Adiga, Aniruddha, Devdatt Dubhashi, Bryan Lewis, et al. (2020). "Mathematical models for covid-19 pandemic: a comparative analysis". In: *Journal of the Indian Institute of Science*, pp. 1–15 (cit. on p. 126).

Afshordi, Niayesh, Benjamin Holder, Mohammad Bahrami, and Daniel Lichtblau (2020). "Diverse local epidemics reveal the distinct effects of population density, demographics, climate, depletion of susceptibles, and intervention in the first wave of COVID-19 in the United States". In: *arXiv preprint arXiv:2007.00159* (cit. on p. 127).

Aleta, Alberto, Guilherme Ferraz de Arruda, and Yamir Moreno (2020). "Data-driven contact structures: from homogeneous mixing to multilayer networks". In: *arXiv preprint arXiv:2003.06946* (cit. on p. 130).

Allen, Arnold O (1990). *Probability, statistics, and queueing theory*. Gulf Professional Publishing (cit. on pp. 30, 45).

Allen, George Edward and Calvin Dytham (2009). "An efficient method for stochastic simulation of biological populations in continuous time". In: *Biosystems* 98.1, pp. 37–42 (cit. on p. 168).

Allen, Linda JS (2015). "Stochastic population and epidemic models". In: *Mathematical biosciences lecture series, stochastics in biological systems* (cit. on p. 128).

Althouse, Benjamin M, Edward A Wenger, Joel C Miller, et al. (2020). "Superspreading events in the transmission dynamics of SARS-CoV-2: Opportunities for interventions and control". In: *PLoS biology* 18.11, e3000897 (cit. on p. 122).

Amini, Hamed, Rama Cont, and Andreea Minca (2016). "Resilience to contagion in financial networks". In: *Mathematical finance* 26.2, pp. 329–365 (cit. on p. 190).

Anderson, Roy M, B Anderson, and Robert M May (1992). *Infectious diseases of humans: dynamics and control*. Oxford university press (cit. on p. 126).

Anderson, William J (2012). *Continuous-time Markov chains: An applications-oriented approach*. Springer Science & Business Media (cit. on p. 33).

Backenköhler, Michael and Gerrit Großmann (2020). "Poster: Birth-Death Processes Reproduce the Infection Footprint of Complex Networks". In: (cit. on p. 14).

Baier, Christel, Boudewijn Haverkort, Holger Hermanns, and J-P Katoen (2003). "Model-checking algorithms for continuous-time Markov chains". In: *IEEE Transactions on software engineering* 29.6, pp. 524–541 (cit. on p. 34).

Baier, Christel, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen (2000). "Model checking continuous-time Markov chains by transient analysis". In: *International Conference on Computer Aided Verification*. Springer, pp. 358–372 (cit. on p. 34).

Bailey, Norman TJ (1964). "Some stochastic models for small epidemics in large populations". In: *Applied Statistics*, pp. 9–19 (cit. on p. 9).

Bak, Per, Kan Chen, and Chao Tang (1990). "A forest-fire model and some thoughts on turbulence". In: *Physics letters A* 147.5-6, pp. 297–300 (cit. on p. 205).

Ball, Frank, David Sirl, and Pieter Trapman (2010). "Analysis of a stochastic SIR epidemic on a random network incorporating household structure". In: *Mathematical Biosciences* 224.2, pp. 53–73 (cit. on p. 138).

Barabasi, Albert-Laszlo (2005). "The origin of bursts and heavy tails in human dynamics". In: *Nature* 435.7039, p. 207 (cit. on p. 69).

Barabási, Albert-László (2013). "Network science". In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371.1987, p. 20120375 (cit. on pp. 9, 17, 20).

Barbarossa, Maria Vittoria, Jan Fuhrmann, Jan H Meinke, et al. (2020). "Modeling the spread of COVID-19 in Germany: Early assessment and possible scenarios". In: *Plos one* 15.9, e0238559 (cit. on p. 127).

Barman, Shohag and Yung-Keun Kwon (2018). "A Boolean network inference from time-series gene expression data using a genetic algorithm". In: *Bioinformatics* 34.17, pp. i927–i933 (cit. on p. 214).

Bartlett, MS (1960). "Monte Carlo studies in ecology and epidemiology". In: *Proc. Fowth Berkeley Symp. Math. Statist. Prob.* Vol. 4, pp. 39–56 (cit. on p. 9).

Benayoun, Marc, Jack D Cowan, Wim van Drongelen, and Edward Wallace (2010). "Avalanches in a stochastic model of spiking neurons". In: *PLoS computational biology* 6.7 (cit. on pp. 8, 99).

Bengio, Emmanuel, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio (2021). "Flow Network based Generative Models for Non-Iterative Diverse Candidate Generation". In: *Advances in Neural Information Processing Systems* 34 (cit. on p. 214).

Bertozzi, Andrea L, Elisa Franco, George Mohler, Martin B Short, and Daniel Sledge (2020). "The challenges of modeling and forecasting the spread of COVID-19". In: *arXiv preprint arXiv:2004.04741* (cit. on p. 128).

Biswas, Kathakali, Abdul Khaleque, and Parongama Sen (2020). "Covid-19 spread: Reproduction of data and prediction using a SIR model on Euclidean network". In: *arXiv preprint arXiv:2003.07063* (cit. on p. 130).

Blythe, SP and RM Anderson (1988). "Variable infectiousness in HFV transmission models". In: *Mathematical Medicine and Biology: A Journal of the IMA* 5.3, pp. 181–200 (cit. on p. 69).

Boguná, Marian, Luis F Lafuerza, Raúl Toral, and M Ángeles Serrano (2014). "Simulating non-Markovian stochastic processes". In: *Physical Review E* 90.4, p. 042108 (cit. on pp. 69, 84).

Boltzmann, Ludwig (2012). *Lectures on gas theory*. Courier Corporation (cit. on p. 9).

Bortolussi, Luca (2016). “Hybrid behaviour of Markov population models”. In: *Information and Computation* 247, pp. 37–86 (cit. on p. 168).

Bortolussi, Luca and Jane Hillston (2012). “Fluid model checking”. In: *International Conference on Concurrency Theory*. Springer, pp. 333–347 (cit. on p. 34).

Bortolussi, Luca, Jane Hillston, Diego Latella, and Mieke Massink (2013). “Continuous approximation of collective system behaviour: A tutorial”. In: *Performance Evaluation* 70.5, pp. 317–349 (cit. on p. 184).

Boudrioua, Mohamed Samir and Abderrahmane Boudrioua (2020). “Predicting the COVID-19 epidemic in Algeria using the SIR model”. In: *medRxiv* (cit. on p. 127).

Bovenkamp, R. van de and P. Van Mieghem (2014). “Time to metastable state in SIS epidemics on graphs”. In: *Signal-Image Technology and Internet-Based Systems*. IEEE, pp. 347–354 (cit. on p. 154).

Brauer, Fred, PV den Driessche, and Jianhong Wu (2008). “Lecture notes in mathematical epidemiology”. In: *Berlin, Germany. Springer* 75.1, pp. 3–22 (cit. on p. 126).

Brauner, Jan M., Sören Mindermann, Mrinank Sharma, et al. (2020). “Inferring the effectiveness of government interventions against COVID-19”. In: *Science*. eprint: `https://science.sciencemag.org/content/early/2020/12/15/science.abd9338.full.pdf` (cit. on p. 121).

Buchanan, Colin R, Cyril R Pernet, Krzysztof J Gorgolewski, Amos J Storkey, and Mark E Bastin (2014). “Test–retest reliability of structural brain networks from diffusion MRI”. In: *Neuroimage* 86, pp. 231–243 (cit. on p. 8).

Buchholz, Peter (1994). “Exact and ordinary lumpability in finite Markov chains”. In: *Journal of applied probability* 31.1, pp. 59–75 (cit. on p. 169).

Bui, Quoctrung, Josh Katz, Alicia Parlapiano, and Margot Sanger-Katz (2020). “What 5 coronavirus models say the next month will look like”. In: *New York Times* (cit. on p. 122).

Campan, Alina, Alfredo Cuzzocrea, and Traian Marius Truta (2017). “Fighting fake news spread in online social networks: Actual trends and future research directions”. In: *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, pp. 4453–4457 (cit. on p. 5).

Campbell, Angus, Gerald Gurin, and Warren E Miller (1954). “The voter decides.” In: (cit. on p. 204).

Cao, Yang, Daniel T Gillespie, and Linda R Petzold (2006). “Efficient step size selection for the tau-leaping simulation method”. In: *The Journal of chemical physics* 124.4, p. 044109 (cit. on p. 168).

Cardelli, Luca, Mirco Tribastone, Max Tschaikowski, and Andrea Vandin (2017). “ERODE: a tool for the evaluation and reduction of ordinary differential equations”. In: *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, pp. 310–328 (cit. on p. 168).

Casadiego, Jose, Mor Nitzan, Sarah Hallerberg, and Marc Timme (2017). “Model-free inference of direct network interactions from nonlinear collective dynamics”. In: *Nature communications* 8.1, pp. 1–10 (cit. on p. 213).

Castellano, Claudio, Santo Fortunato, and Vittorio Loreto (2009). “Statistical physics of social dynamics”. In: *Reviews of modern physics* 81.2, p. 591 (cit. on p. 9).

Castro, Mario, Saúl Ares, José A Cuesta, and Susanna Manrubia (2020). “The turning point and end of an expanding epidemic cannot be precisely forecast”. In: *Proceedings of the National Academy of Sciences* 117.42, pp. 26190–26196 (cit. on p. 128).

Cave, Emma (2020). “COVID-19 super-spreaders: Definitional quandaries and implications”. In: *Asian Bioethics Review*, p. 1 (cit. on p. 123).

Cevik, Muge, Julia Marcus, Caroline Buckee, and Tara Smith (2020). “SARS-CoV-2 transmission dynamics should inform policy”. In: *Available at SSRN 3692807* (cit. on p. 123).

Chan, Thalia E, Michael PH Stumpf, and Ann C Babtie (2017). “Gene regulatory network inference from single-cell data using multivariate information measures”. In: *Cell systems* 5.3, pp. 251–267 (cit. on p. 190).

Chen, Jun, Ming Hu, Boyang Li, and Mohamed Elhoseiny (2022). “Efficient Self-supervised Vision Pretraining with Local Masked Reconstruction”. In: *arXiv preprint arXiv:2206.00790* (cit. on p. 214).

Chen, Wenbin, Nagiza F Samatova, Matthias F Stallmann, William Hendrix, and Weiqin Ying (2016). "On size-constrained minimum s–t cut problems and size-constrained dense subgraph problems". In: *Theoretical Computer Science* 609, pp. 434–442 (cit. on p. 160).

Cheng, Shin-Ming, Weng Chon Ao, Pin-Yu Chen, and Kwang-Cheng Chen (2010). "On modeling malware propagation in generalized social networks". In: *IEEE Communications Letters* 15.1, pp. 25–27 (cit. on p. 5).

Comunian, Alessandro, Romina Gaburro, and Mauro Giudici (2020). "Inversion of a SIR-based model: A critical analysis about the application to COVID-19 epidemic". In: *Physica D: Nonlinear Phenomena* 413, p. 132674 (cit. on p. 128).

Cooper, Ian, Argha Mondal, and Chris G Antonopoulos (2020). "A SIR model assumption for the spread of COVID-19 in different communities". In: *Chaos, Solitons & Fractals* 139, p. 110057 (cit. on p. 127).

Cota, Wesley and Silvio C Ferreira (2017). "Optimized Gillespie algorithms for the simulation of Markovian epidemic processes on large and heterogeneous networks". In: *Computer Physics Communications* 219, pp. 303–312 (cit. on pp. 43, 45, 47, 56, 62, 91).

Cox, Christopher (Nov. 2020). *The Vulnerable Can Wait. Vaccinate the Super-Spreaders First Who gets priority when Covid-19 shots are in short supply? Network theorists have a counterintuitive answer: Start with the social butterflies.* `https://www.wired.com/story/covid-19-vaccine-super-spreaders/`. Online; accessed May 24, 2022 (cit. on p. 119).

Cox, David Roxbee (1962). "Renewal theory". In: (cit. on p. 76).

D'Angelo, Gianlorenzo, Lorenzo Severini, and Yllka Velaj (2016). "Influence Maximization in the Independent Cascade Model." In: *ICTCS*, pp. 269–274 (cit. on p. 74).

Daley, Daryl J and D Vere Jones (2003). *An Introduction to the Theory of Point Processes: Elementary Theory of Point Processes*. Springer (cit. on pp. 76, 80).

Dassios, Angelos, Hongbiao Zhao, et al. (2013). "Exact simulation of Hawkes process with exponentially decaying intensity". In: *Electronic Communications* 18 (cit. on p. 99).

Davidson, James, François Bouchart, Stephen Cavill, and Paul Jowitt (2005). "Real-time connectivity modeling of water distribution networks to predict contamination spread". In: *Journal of Computing in Civil Engineering* 19.4, pp. 377–386 (cit. on p. 5).

De Visscher, Alex (2020). "The COVID-19 pandemic: model-based evaluation of non-pharmaceutical interventions and prognoses". In: *Nonlinear dynamics* 101.3, pp. 1871–1887 (cit. on p. 127).

Dehning, Jonas, Johannes Zierenberg, F Paul Spitzner, et al. (2020). "Inferring COVID-19 spreading rates and potential change points for case number forecasts". In: *arXiv preprint arXiv:2004.01105* (cit. on p. 127).

Derisavi, Salem, Holger Hermanns, and William H Sanders (2003). "Optimal state-space lumping in Markov chains". In: *Information processing letters* 87.6, pp. 309–315 (cit. on p. 34).

Desikan, Rahul S, Florent Ségonne, Bruce Fischl, et al. (2006). "An automated labeling system for subdividing the human cerebral cortex on MRI scans into gyral based regions of interest". In: *Neuroimage* 31.3, pp. 968–980 (cit. on p. 7).

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (cit. on p. 214).

Devriendt, Karel and Piet Van Mieghem (2019). "Tighter spectral bounds for the cut size, based on Laplacian eigenvectors". In: *Linear Algebra and its Applications* 572, pp. 68–91 (cit. on p. 160).

– (2017). "Unified mean-field framework for susceptible-infected-susceptible epidemics on networks, based on graph partitioning and the isoperimetric inequality". In: *Physical Review E* 96.5, p. 052314 (cit. on pp. 166, 169, 184, 187).

Di Lauro, F, J-C Croix, M Dashti, L Berthouze, and IZ Kiss (2020). "Network inference from population-level observation of epidemics". In: *Scientific Reports* 10.1, pp. 1–14 (cit. on pp. 166, 213).

Dolbeault, Jean and Gabriel Turinici (2020). "Heterogeneous social interactions and the COVID-19 lockdown outcome in a multi-group SEIR model". In: *arXiv preprint arXiv:2005.00049* (cit. on p. 127).

Doob, Joseph L (1945). “Markoff chains–denumerable case”. In: *Transactions of the American Mathematical Society* 58.3, pp. 455–473 (cit. on pp. 10, 43).

Ellison, Glenn (2020). *Implications of heterogeneous SIR models for analyses of COVID-19*. Tech. rep. National Bureau of Economic Research (cit. on p. 127).

Endo, Akira, Sam Abbott, Adam J Kucharski, Sebastian Funk, et al. (2020). “Estimating the overdispersion in COVID-19 transmission using outbreak sizes outside China”. In: *Wellcome Open Research* 5.67, p. 67 (cit. on pp. 123, 129).

Erdős, Paul, Alfréd Rényi, et al. (1960). “On the evolution of random graphs”. In: *Publ. Math. Inst. Hung. Acad. Sci* 5.1, pp. 17–60 (cit. on p. 10).

Estrada, Ernesto and Philip A Knight (2015). *A first course in network theory*. Oxford University Press, USA (cit. on pp. 9, 10).

Fagiolo, Giorgio (2020). “Assessing the Impact of Social Network Structure on the Diffusion of Coronavirus Disease (COVID-19): A Generalized Spatial SEIRD Model”. In: *arXiv preprint arXiv:2010.11212* (cit. on p. 130).

Farajtabar, Mehrdad, Yichen Wang, Manuel Gomez Rodriguez, et al. (2015). “Coevolve: A joint point process model for information diffusion and network co-evolution”. In: *Advances in Neural Information Processing Systems*, pp. 1954–1962 (cit. on pp. 94, 98, 99).

Farrington, CP, MN Kanaan, and NJ Gay (2003). “Branching process models for surveillance of infectious diseases controlled by mass vaccination”. In: *Biostatistics* 4.2, pp. 279–295 (cit. on p. 128).

Feng, Z and HR Thieme (2000). “Endemic models for the spread of infectious diseases with arbitrarily distributed disease stages I: General theory”. In: *SIAM J. Appl. Math* 61.3, pp. 803–833 (cit. on p. 69).

Fennell, Peter G (2015). “Stochastic processes on complex networks: techniques and explorations”. In: (cit. on p. 9).

Fennell, Peter G and James P Gleeson (2019). “Multistate dynamical processes on networks: analysis through degree-based approximation frameworks”. In: *SIAM Review* 61.1, pp. 92–118 (cit. on p. 27).

Fiedler, Miroslav (1973). “Algebraic connectivity of graphs”. In: *Czechoslovak mathematical journal* 23.2, pp. 298–305 (cit. on p. 163).

Fingelkurts, Andrew A, Alexander A Fingelkurts, and Seppo Kähkönen (2005). “Functional connectivity in the brain—is it an elusive concept?” In: *Neuroscience & Biobehavioral Reviews* 28.8, pp. 827–836 (cit. on p. 8).

Finn, Kelly R, Matthew J Silk, Mason A Porter, and Noa Pinter-Wollman (2019). “The use of multilayer network analysis in animal behaviour”. In: *Animal behaviour* 149, pp. 7–22 (cit. on p. 190).

Fornito, Alex, Andrew Zalesky, and Michael Breakspear (2015). “The connectomics of brain disorders”. In: *Nature Reviews Neuroscience* 16.3, pp. 159–172 (cit. on p. 190).

Fornito, Alex, Andrew Zalesky, and Edward Bullmore (2016). *Fundamentals of brain network analysis*. Academic Press (cit. on p. 7).

Fosdick, Bailey K, Daniel B Larremore, Joel Nishimura, and Johan Ugander (2018). “Configuring random graph models with fixed degree sequences”. In: *SIAM Review* 60.2, pp. 315–355 (cit. on pp. 61, 95).

Frauenthal, James C (2012). *Mathematical modeling in epidemiology*. Springer Science & Business Media (cit. on p. 126).

Friedman, Jerome, Trevor Hastie, and Robert Tibshirani (2008). “Sparse inverse covariance estimation with the graphical lasso”. In: *Biostatistics* 9.3, pp. 432–441 (cit. on p. 213).

Fu, Tianfan, Cao Xiao, Xinhao Li, Lucas M Glass, and Jimeng Sun (2020). “Mimosa: Multi-constraint molecule sampling for molecule optimization”. In: *arXiv preprint arXiv:2010.02318* (cit. on p. 214).

Ganguly, Arnab, Tatjana Petrov, and Heinz Koeppl (2014). “Markov chain aggregation and its applications to combinatorial reaction networks”. In: *Journal of mathematical biology* 69.3, pp. 767–797 (cit. on p. 168).

Garcia, P, A Parravano, MG Cosenza, J Jiménez, and A Marcano (2002). “Coupled map networks as communication schemes”. In: *Physical Review E* 65.4, p. 045201 (cit. on p. 205).

Gawrychowski, Paweł, Shay Mozes, and Oren Weimann (2019). “Minimum Cut in O(m2n) Time”. In: *arXiv preprint arXiv:1911.01145* (cit. on p. 160).

Gehring, Ronette, Phillip Schumm, Mina Youssef, and Caterina Scoglio (2010). “A network-based approach for resistance transmission in bacterial populations”. In: *Journal of theoretical biology* 262.1, pp. 97–106 (cit. on p. 5).

Gerhard, Felipe and Wulfram Gerstner (2010). “Rescaling, thinning or complementing? On goodness-of-fit procedures for point process models and Generalized Linear Models”. In: *Advances in neural information processing systems*, pp. 703–711 (cit. on pp. 80, 86, 92).

Gillespie, Daniel T (1977). “Exact stochastic simulation of coupled chemical reactions”. In: *The journal of physical chemistry* 81.25, pp. 2340–2361 (cit. on pp. 10, 43).

Girvan, Michelle and Mark EJ Newman (2002). “Community structure in social and biological networks”. In: *Proceedings of the national academy of sciences* 99.12, pp. 7821–7826 (cit. on pp. 174, 184).

Gomes, M Gabriela M, Ricardo Aguas, Rodrigo M Corder, et al. (2020). “Individual variation in susceptibility or exposure to SARS-CoV-2 lowers the herd immunity threshold”. In: *medRxiv* (cit. on p. 128).

Gomory, Ralph E and Tien Chung Hu (1961). “Multi-terminal network flows”. In: *Journal of the Society for Industrial and Applied Mathematics* 9.4, pp. 551–570 (cit. on p. 160).

Goutsias, John and Garrett Jenkinson (2013). “Markovian dynamics on complex reaction networks”. In: *Physics reports* 529.2, pp. 199–264 (cit. on p. 10).

Goyal, Ashish, Daniel B Reeves, E Fabian Cardozo-Ojeda, Joshua T Schiffer, and Bryan T Mayer (2020). “Wrong person, place and time: viral load and contact network structure predict SARS-CoV-2 transmission and super-spreading events”. In: *Medrxiv* (cit. on pp. 123, 129).

Grima, R. (2010). “An effective rate equation approach to reaction kinetics in small volumes: Theory and application to biochemical reactions in nonequilibrium steady-state conditions”. In: *The Journal of Chemical Physics* 133.3, p. 035101 (cit. on p. 168).

Grima, Ramon (2012). “A study of the accuracy of moment-closure approximations for stochastic chemical kinetics”. In: *The Journal of chemical physics* 136.15, 04B616 (cit. on pp. 168, 187).

Großmann, Gerrit and Michael Backenköhler (2022). “Birth-Death Processes Reproduce the Epidemic Footprint”. In: *International Conference on Complex Networks and Their Applications (Book of Abstracts)*. Springer (cit. on p. 14).

Großmann, Gerrit, Michael Backenköhler, Jonas Klesen, and Verena Wolf (2020). “Learning Vaccine Allocation from Simulations”. In: *International Conference on Complex Networks and Their Applications*. Springer, pp. 432–443 (cit. on pp. 13, 190).

Großmann, Gerrit, Michael Backenköhler, and Verena Wolf (2021a). “Epidemic overdispersion strengthens the effectiveness of mobility restrictions”. In: *Proceedings of the 24th International Conference on Hybrid Systems: Computation and Control*, pp. 1–2 (cit. on p. 149).

– (2021b). “Heterogeneity matters: Contact structure and individual variation shape epidemic dynamics”. In: *Plos one* 16.7, e0250050 (cit. on p. 14).

– (2020). “Importance of interaction structure and stochasticity for epidemic spreading: A COVID-19 case study”. In: *International Conference on Quantitative Evaluation of Systems*. Springer, pp. 211–229 (cit. on p. 13).

Großmann, Gerrit and Luca Bortolussi (2019). “Reducing spreading processes on networks to Markov population models”. In: *International Conference on Quantitative Evaluation of Systems*. Springer, pp. 292–309 (cit. on p. 15).

Großmann, Gerrit, Luca Bortolussi, and Verena Wolf (2020). “Efficient simulation of non-Markovian dynamics on complex networks”. In: *Plos one* 15.10, e0241394 (cit. on p. 12).

– (2019). “Rejection-based simulation of non-Markovian agents on complex networks”. In: *international conference on complex networks and their applications*. Springer, pp. 349–361 (cit. on p. 12).

Großmann, Gerrit, Charalampos Kyriakopoulos, Luca Bortolussi, and Verena Wolf (2018). “Lumping the approximate master equation for multistate processes on complex networks”. In: *International Conference on Quantitative Evaluation of Systems*. Springer, pp. 157–172 (cit. on p. 169).

Großmann, Gerrit and Verena Wolf (2019). “Rejection-based simulation of stochastic spreading processes on complex networks”. In: *International Workshop on Hybrid Systems Biology*. Springer, pp. 63–79 (cit. on p. 12).

Großmann, Gerrit, Julian Zimmerlin, Michael Backenköhler, and Verena Wolf (2021). "GINA: Neural Relational Inference From Independent Snapshots". In: *arXiv preprint arXiv:2105.14329* (cit. on p. 15).

Gu, Shi, Fabio Pasqualetti, Matthew Cieslak, et al. (2015). "Controllability of structural brain networks". In: *Nature communications* 6.1, pp. 1–10 (cit. on p. 190).

Hagberg, Aric and Daniel A Schult (2008). "Rewiring networks for synchronization". In: *Chaos: An interdisciplinary journal of nonlinear science* 18.3, p. 037105 (cit. on p. 190).

Hagberg, Aric, Pieter Swart, and Daniel S Chult (2008). *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States) (cit. on pp. 176, 208, 268).

Harris, Theodore Edward et al. (1963). *The theory of branching processes*. Vol. 6. Springer Berlin (cit. on p. 128).

Hartle, Harrison, Brennan Klein, Stefan McCabe, et al. (2020). "Network comparison and the within-ensemble graph distance". In: *Proceedings of the Royal Society A* 476.2243, p. 20190744 (cit. on p. 209).

Hasan, Agus, Hadi Susanto, Muhammad Kasim, et al. (2020). "Superspreading in Early Transmissions of COVID-19 in Indonesia". In: *medRxiv* (cit. on p. 123).

Hashemifar, Somaye, Behnam Neyshabur, Aly A Khan, and Jinbo Xu (2018). "Predicting protein–protein interactions through sequence-based deep learning". In: *Bioinformatics* 34.17, pp. i802–i810 (cit. on p. 213).

Hayward, Ryan and Colin McDiarmid (1991). "Average case analysis of heap building by repeated insertion". In: *J. Algorithms* 12.1, pp. 126–153 (cit. on p. 56).

Hébert-Dufresne, Laurent, Benjamin M Althouse, Samuel V Scarpino, and Antoine Allard (2020). "Beyond R0: Heterogeneity in secondary infections and probabilistic epidemic forecasting". In: *medRxiv* (cit. on p. 123).

Held, Leonhard, Niel Hens, Philip D O'Neill, and Jacco Wallinga (2019). *Handbook of infectious disease data analysis*. CRC Press (cit. on p. 9).

Henzinger, Thomas A, Maria Mateescu, and Verena Wolf (2009). "Sliding window abstraction for infinite Markov chains". In: *International Conference on Computer Aided Verification*. Springer, pp. 337–352 (cit. on p. 168).

Hollingsworth, T D., R. M Anderson, and C. Fraser (2008). "HIV-1 transmission, by stage of infection". In: *The Journal of infectious diseases* 198.5, pp. 687–693 (cit. on p. 69).

Holmdahl, Inga and Caroline Buckee (2020). "Wrong but useful—what covid-19 epidemiologic models can and cannot tell us". In: *New England Journal of Medicine* (cit. on p. 122).

Holme, Petter (2015a). "Modern temporal network theory: a colloquium". In: *The European Physical Journal B* 88.9, p. 234 (cit. on p. 59).

– (2015b). "Shadows of the susceptible-infectious-susceptible immortality transition in small networks". In: *Physical Review E* 92.1, p. 012804 (cit. on p. 181).

Holme, Petter and Jari Saramäki (2012). "Temporal networks". In: *Physics reports* 519.3, pp. 97–125 (cit. on p. 59).

Horstmeyer, Leonhard, Christian Kuehn, and Stefan Thurner (2020). "Balancing quarantine and self-distancing measures in adaptive epidemic networks". In: *arXiv preprint arXiv:2010.10516* (cit. on p. 130).

Hosseini, Soodeh and Mohammad Abdollahi Azgomi (2016). "A model for malware propagation in scale-free networks based on rumor spreading process". In: *Computer Networks* 108, pp. 97–107 (cit. on p. 5).

Hu, Weihua, Bowen Liu, Joseph Gomes, et al. (2019). "Strategies for pre-training graph neural networks". In: *arXiv preprint arXiv:1905.12265* (cit. on p. 214).

Huang, Keke, Shuo Li, Penglin Dai, Zhen Wang, and Zhaofei Yu (2020). "SDARE: A stacked denoising autoencoder method for game dynamics network structure reconstruction". In: *Neural Networks* 126, pp. 143–152 (cit. on p. 213).

Huepe, Cristián, Gerd Zschaler, Anne-Ly Do, and Thilo Gross (2011). "Adaptive-network models of swarm dynamics". In: *New Journal of Physics* 13.7, p. 073022 (cit. on p. 5).

Humphries, Rory, Mary Spillane, Kieran Mulchrone, et al. (2020). “A metapopulation network model for the spreading of SARS-CoV-2: Case study for Ireland”. In: *medRxiv* (cit. on p. 127).

Ioannidis, John PA (2020). “Coronavirus disease 2019: the harms of exaggerated information and non-evidence-based measures”. In: *European journal of clinical investigation* (cit. on p. 126).

Jackson, James R (1957). “Networks of waiting lines”. In: *Operations research* 5.4, pp. 518–521 (cit. on p. 10).

Jeh, Glen and Jennifer Widom (2003). “Scaling personalized web search”. In: *Proceedings of the 12th international conference on World Wide Web*, pp. 271–279 (cit. on p. 116).

Ji, Xiaoyun (2004). *Graph partition problems with minimum size constraints*. Rensselaer Polytechnic Institute (cit. on p. 160).

Jo, Hang-Hyun, Byoung-Hwa Lee, Takayuki Hiraoka, and Woo-Sung Jung (2019). “Copula-based algorithm for generating bursty time series”. In: *arXiv preprint arXiv:1904.08795* (cit. on p. 101).

Jo, Hang-Hyun, Juan I Perotti, Kimmo Kaski, and János Kertész (2014). “Analytically solvable model of spreading dynamics with non-Poissonian processes”. In: *Physical Review X* 4.1, p. 011041 (cit. on p. 84).

Jones, Terry C, Barbara Mühlemann, Talitha Veith, et al. (2020). “An analysis of SARS-CoV-2 viral load by patient age”. In: *medRxiv* (cit. on p. 123).

Kaneko, Kunihiko (1992). “Overview of coupled map lattices”. In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 2.3, pp. 279–282 (cit. on pp. 205, 268).

Karaivanov, Alexander (2020). “A Social Network Model of COVID-19”. In: *Available at SSRN 3584895* (cit. on p. 130).

Karp, Richard M (1972). “Reducibility among combinatorial problems”. In: *Complexity of computer computations*. Springer, pp. 85–103 (cit. on p. 160).

Katoen, Joost-Pieter, Marta Kwiatkowska, Gethin Norman, and David Parker (2001). “Faster and symbolic CTMC model checking”. In: *Joint International Workshop von Process Algebra and Probabilistic Methods, Performance Modeling and Verification*. Springer, pp. 23–38 (cit. on p. 34).

Keeler, Paul (Mar. 2019). *Simulating an inhomogeneous Poisson point process*. `https://hpaulkeeler.com/simulating-an-inhomogeneous-poisson-point-process/`. Online; accessed 19-May-2020 (cit. on p. 80).

Kermack, William Ogilvy and Anderson G McKendrick (1927). "A contribution to the mathematical theory of epidemics". In: *Proceedings of the royal society of london. Series A, Containing papers of a mathematical and physical character* 115.772, pp. 700–721 (cit. on p. 9).

Kerr, Cliff C, Robyn M Stuart, Dina Mistry, et al. (2020). "Covasim: an agent-based model of COVID-19 dynamics and interventions". In: *medRxiv* (cit. on p. 130).

Khailaie, Sahamoddin, Tanmay Mitra, Arnab Bandyopadhyay, et al. (2020). "Estimate of the development of the epidemic reproduction number Rt from Coronavirus SARS-CoV-2 case data and implications for political measures based on prognostics". In: *medRxiv* (cit. on p. 127).

KhudaBukhsh, Wasiur R, Arnab Auddy, Yann Disser, and Heinz Koeppl (2019). "Approximate lumpability for Markovian agent-based models using local symmetries". In: *Journal of Applied Probability* 56.3, pp. 647–671 (cit. on p. 169).

Kingman, John FC (1969). "Markov population processes". In: *Journal of Applied Probability* 6.1, pp. 1–18 (cit. on p. 10).

Kipf, Thomas, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel (2018). "Neural relational inference for interacting systems". In: *International Conference on Machine Learning*. PMLR, pp. 2688–2697 (cit. on pp. 190, 191, 213).

Kishan, KC, Rui Li, Feng Cui, Qi Yu, and Anne R Haake (2019). "GNE: a deep learning framework for gene network inference by aggregating biological information". In: *BMC systems biology* 13.2, p. 38 (cit. on p. 213).

Kiss, I. Z, G. Röst, and Z. Vizi (2015). "Generalization of pairwise models to non-Markovian epidemics on networks". In: *Physical review letters* 115.7, p. 078701 (cit. on p. 84).

Kiss, István Z, Joel C Miller, Péter L Simon, et al. (2017). "Mathematics of epidemics on networks". In: *Cham: Springer* 598 (cit. on pp. 17, 43, 48, 49, 84, 169).

Klepac, Petra, Adam J Kucharski, Andrew JK Conlan, et al. (2020). "Contacts in context: large-scale setting-specific social mixing matrices from the BBC Pandemic project". In: *medRxiv*. eprint: `https://www.medrxiv.org/content/early/2020/03/05/2020.02.16.20023754.full.pdf` (cit. on p. 127).

Koç, Yakup, Martijn Warnier, Piet Van Mieghem, Robert E Kooij, and Frances MT Brazier (2014). "The impact of the topology on cascading failures in a power grid model". In: *Physica A: Statistical Mechanics and its Applications* 402, pp. 169–179 (cit. on p. 5).

Kuhl, Ellen (2020). "Data-driven modeling of COVID-19—Lessons learned". In: *Extreme Mechanics Letters*, p. 100921 (cit. on pp. 122, 126).

Kyriakopoulos, Charalampos, Gerrit Grossmann, Verena Wolf, and Luca Bortolussi (2018). "Lumping of degree-based mean-field and pair-approximation equations for multistate contact processes". In: *Physical Review E* 97.1, p. 012301 (cit. on pp. 169, 176).

Langer, Nicolas, Andreas Pedroni, Lorena RR Gianotti, et al. (2012). "Functional brain network efficiency predicts intelligence". In: *Human brain mapping* 33.6, pp. 1393–1406 (cit. on p. 7).

Levesque, Jérôme, David W Maybury, and RHA David Shaw (2020). "A model of COVID-19 propagation based on a gamma subordinated negative binomial branching process". In: *Journal of Theoretical Biology*, p. 110536 (cit. on p. 129).

Li, Genyuan and Herschel Rabitz (1990). "A general analysis of approximate lumping in chemical kinetics". In: *Chemical engineering science* 45.4, pp. 977–1002 (cit. on p. 169).

Liang, Xia, Jinhui Wang, Chaogan Yan, et al. (2012). "Effects of different correlation metrics and preprocessing factors on small-world brain functional networks: a resting-state functional MRI study". In: *PloS one* 7.3, e32766 (cit. on p. 8).

Liu, Congying, Xiaoqun Wu, Riuwu Niu, Xiuqi Wu, and Ruguo Fan (2020). "A new SAIR model on complex networks for analysing the 2019 novel coronavirus (COVID-19)". In: *Nonlinear Dynamics* 101.3, pp. 1777–1787 (cit. on p. 130).

Lloyd, Alun L (2001). "Realistic distributions of infectious periods in epidemic models: changing patterns of persistence and dynamics". In: *Theoretical population biology* 60.1, pp. 59–71 (cit. on p. 69).

Lloyd-Smith, James O (2007). “Maximum likelihood estimation of the negative binomial dispersion parameter for highly overdispersed data, with applications to infectious diseases”. In: *PloS one* 2.2, e180 (cit. on pp. 123, 125).

Lloyd-Smith, James O, Sebastian J Schreiber, P Ekkehard Kopp, and Wayne M Getz (2005). “Superspreading and the effect of individual variation on disease emergence”. In: *Nature* 438.7066, pp. 355–359 (cit. on pp. 123, 128, 148).

Lotka, Alfred James (1925). *Elements of physical biology*. Williams & Wilkins (cit. on p. 9).

Lourenço, José, Robert Paton, Mahan Ghafari, et al. (2020). “Fundamental principles of epidemic spread highlight the immediate need for large-scale serological surveys to assess the stage of the SARS-CoV-2 epidemic”. In: *medRxiv* (cit. on p. 127).

Lu, Jian and Dengtian Lu (2022). “Modelling traffic congestion propagation based on data analysis method”. In: *Sixth International Conference on Electromechanical Control Technology and Transportation (ICECTT 2021)*. Vol. 12081. SPIE, pp. 959–964 (cit. on p. 5).

Lynn, Christopher W and Danielle S Bassett (2019). “The physics of brain network structure, function and control”. In: *Nature Reviews Physics* 1.5, pp. 318–332 (cit. on p. 7).

Ma, Dan (July 2011). *Applied Probability and Statistics - The hazard rate function*. `http://statisticalmodeling.wordpress.com/tag/non-homogeneous-poisson-process/`. Online; accessed 10-February-2020 (cit. on p. 76).

Ma, Stefan and Yingcun Xia (2009). *Mathematical understanding of infectious disease dynamics*. Vol. 16. World Scientific (cit. on p. 126).

Mancastroppa, Marco, Raffaella Burioni, Vittoria Colizza, and Alessandro Vezzani (2020). “Active and inactive quarantine in epidemic spreading on adaptive activity-driven networks”. In: *arXiv preprint arXiv:2004.07902* (cit. on p. 130).

Martinez, J Arjona, Olmo Cerri, Maria Spiropulu, JR Vlimant, and M Pierini (2019). “Pileup mitigation at the Large Hadron Collider with graph neural networks”. In: *The European Physical Journal Plus* 134.7, p. 333 (cit. on p. 190).

Masuda, Naoki and Petter Holme (2017). *Temporal Network Epidemiology*. Springer (cit. on p. 59).

Masuda, Naoki and Norio Konno (2006). "Multi-state epidemic processes on complex networks". In: *Journal of Theoretical Biology* 243.1, pp. 64–75 (cit. on p. 58).

Masuda, Naoki and Luis EC Rocha (2018). "A Gillespie algorithm for non-Markovian stochastic processes". In: *SIAM Review* 60.1, pp. 95–115 (cit. on pp. 69, 85, 98, 101).

Mateescu, M, V Wolf, F Didier, and TA Henzinger (2010). "Fast adaptive uniformisation of the chemical master equation". In: *IET systems biology* 4.6, pp. 441–452 (cit. on p. 168).

May, Robert M (2004). "Simple mathematical models with very complicated dynamics". In: *The Theory of Chaotic Attractors*, pp. 85–93 (cit. on p. 268).

Meier, Jil, X Zhou, Arjan Hillebrand, et al. (2017). "The epidemic spreading model and the direction of information flow in brain networks". In: *NeuroImage* 152, pp. 639–646 (cit. on p. 8).

Mello, Isys F, Lucas Squillante, Gabriel O Gomes, Antonio C Seridonio, and Mariano de Souza (2021). "Epidemics, the Ising-model and percolation theory: a comprehensive review focussed on Covid-19". In: *Physica A: Statistical Mechanics and its Applications*, p. 125963 (cit. on p. 127).

Memmesheimer, Raoul-Martin and Marc Timme (2006). "Designing complex networks". In: *Physica D: Nonlinear Phenomena* 224.1-2, pp. 182–201 (cit. on p. 190).

Millán, Ana P, Elisabeth CW van Straaten, Cornelis J Stam, et al. (2022). "Epidemic models characterize seizure propagation and the effects of epilepsy surgery in individualized brain networks based on MEG and invasive EEG recordings". In: *Scientific reports* 12.1, pp. 1–20 (cit. on p. 8).

Mishra, Pushkar, Aleksandra Piktus, Gerard Goossen, and Fabrizio Silvestri (2020). "Node masking: Making graph neural networks generalize and scale better". In: *arXiv preprint arXiv:2001.07524* (cit. on pp. 192, 214).

Moslonka-Lefebvre, Mathieu, Marco Pautasso, and Mike J Jeger (2009). "Disease spread in small-size directed networks: epidemic threshold, correlation between links to and from nodes, and clustering". In: *Journal of theoretical biology* 260.3, pp. 402–411 (cit. on p. 181).

Moussaid, Mehdi, Simon Garnier, Guy Theraulaz, and Dirk Helbing (2009). "Collective information processing and pattern formation in swarms, flocks, and crowds". In: *Topics in Cognitive Science* 1.3, pp. 469–497 (cit. on p. 5).

Müller, Johannes and Volker Hösel (2020). "Contact Tracing & Super-Spreaders in the Branching-Process Model". In: *arXiv preprint arXiv:2010.04942* (cit. on pp. 123, 128).

Munday, James D, Katharine Sherratt, Sophie Meakin, et al. (2020). "Implications of the school-household network structure on SARS-CoV-2 transmission under different school reopening strategies in England". In: *medRxiv* (cit. on p. 130).

Murphy, Charles, Edward Laurence, and Antoine Allard (2021). "Deep learning of contagion dynamics on complex networks". In: *Nature Communications* 12.1, pp. 1–11 (cit. on p. 8).

Nande, Anjalika, Ben Adlam, Justin Sheen, Michael Z Levy, and Alison L Hill (2021). "Dynamics of COVID-19 under social distancing measures are driven by transmission network structure". In: *PLOS Computational Biology* 17.2. See also for the computation of R0: `alhill.shinyapps.io/COVID19seir/`, e1008684 (cit. on pp. 130–132, 136).

Neipel, Jonas, Jonathan Bauermann, Stefano Bo, Tyler Harmon, and Frank Jülicher (2020). "Power-law population heterogeneity governs epidemic waves". In: *PloS one* 15.10, e0239678 (cit. on p. 127).

Nelson, Kenrad E and Carolyn Masters Williams (2014). *Infectious disease epidemiology: theory and practice*. Jones & Bartlett Publishers (cit. on p. 126).

Netrapalli, Praneeth, Siddhartha Banerjee, Sujay Sanghavi, and Sanjay Shakkottai (2010). "Greedy learning of Markov network structure". In: *2010 48th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. IEEE, pp. 1295–1302 (cit. on p. 214).

Newman, Mark (2018). *Networks*. Oxford university press (cit. on p. 17).

Newman, Mark EJ (2018). "Estimating network structure from unreliable measurements". In: *Physical Review E* 98.6, p. 062321 (cit. on p. 213).

Nielsen, Bjarke Frost and Kim Sneppen (2020). "COVID-19 superspreading suggests mitigation by social network modulation". In: *medRxiv* (cit. on p. 130).

Nowzari, Cameron, Victor M Preciado, and George J Pappas (2016). “Analysis and control of epidemics: A survey of spreading processes on complex networks”. In: *IEEE Control Systems Magazine* 36.1, pp. 26–46 (cit. on p. 105).

Omranian, Nooshin, Jeanne MO Eloundou-Mbebi, Bernd Mueller-Roeber, and Zoran Nikoloski (2016). “Gene regulatory network inference using fused LASSO on multiple data sets”. In: *Scientific reports* 6.1, pp. 1–14 (cit. on p. 213).

St-Onge, Guillaume, Jean-Gabriel Young, Laurent Hébert-Dufresne, and Louis J Dubé (2018). “Efficient sampling of spreading processes on complex networks using a composition and rejection algorithm”. In: *arXiv preprint arXiv:1808.05859* (cit. on p. 47).

Orlin, James B (2013). “Max flows in O (nm) time, or better”. In: *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pp. 765–774 (cit. on p. 160).

Page, Lawrence, Sergey Brin, Rajeev Motwani, and Terry Winograd (Nov. 1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Technical Report 1999-66. Previous number = SIDL-WP-1999-0120. Stanford InfoLab (cit. on p. 110).

Pastor-Satorras, Romualdo, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani (2015). “Epidemic processes in complex networks”. In: *Reviews of modern physics* 87.3, p. 925 (cit. on p. 84).

Pastor-Satorras, Romualdo and Alessandro Vespignani (2001). “Epidemic spreading in scale-free networks”. In: *Physical review letters* 86.14, p. 3200 (cit. on p. 130).

Pasupathy, Raghu (2011). “Generating nonhomogeneous poisson processes”. In: (cit. on pp. 80, 83).

Paszke, Adam, Sam Gross, Francisco Massa, et al. (2019). “Pytorch: An imperative style, high-performance deep learning library”. In: *arXiv preprint arXiv:1912.01703* (cit. on p. 204).

Paulus, Max, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison (2020). “Gradient estimation with stochastic softmax tricks”. In: *Advances in Neural Information Processing Systems* 33, pp. 5691–5704 (cit. on p. 214).

Pautasso, Marco, Mathieu Moslonka-Lefebvre, and Michael J Jeger (2010). "The number of links to and from the starting node as a predictor of epidemic size in small-size directed networks". In: *Ecological Complexity* 7.4, pp. 424–432 (cit. on p. 181).

Pellis, Lorenzo, Thomas House, and Matt J Keeling (2015). "Exact and approximate moment closures for non-Markovian network epidemics". In: *Journal of theoretical biology* 382, pp. 160–177 (cit. on p. 84).

Petrov, Tatjana and Stefano Tognazzi (2021). "Lumping Reductions for Multispread in Multi-Layer Networks". In: *International Conference on Complex Networks and Their Applications*. Springer, pp. 289–300 (cit. on p. 169).

Plateau, Brigitte and William J Stewart (2000). "Stochastic automata networks". In: *Computational Probability*. Springer, pp. 113–151 (cit. on p. 27).

Porter, Thomas and Istvan Simon (1975). "Random insertion into a priority queue structure". In: *IEEE Transactions on Software Engineering* 3, pp. 292–298 (cit. on p. 56).

Prakash, B Aditya, Deepayan Chakrabarti, Nicholas C Valler, Michalis Faloutsos, and Christos Faloutsos (2012). "Threshold conditions for arbitrary cascade models on arbitrary networks". In: *Knowledge and information systems* 33.3, pp. 549–575 (cit. on pp. 10, 148, 154).

Prakash, B Aditya, Hanghang Tong, Nicholas Valler, Michalis Faloutsos, and Christos Faloutsos (2010). "Virus propagation on time-varying networks: Theory and immunization algorithms". In: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 99–114 (cit. on p. 105).

Prakash, B Aditya, Jilles Vreeken, and Christos Faloutsos (2012). "Spotting culprits in epidemics: How many and which ones?" In: *2012 IEEE 12th International Conference on Data Mining*. IEEE, pp. 11–20 (cit. on p. 190).

Prasse, Bastian and Piet Van Mieghem (2018). "Maximum-likelihood network reconstruction for SIS processes is NP-hard". In: *arXiv preprint arXiv:1807.08630* (cit. on p. 198).

– (2020a). "Network reconstruction and prediction of epidemic outbreaks for general group-based compartmental epidemic models". In: *IEEE Transactions on Network Science and Engineering* (cit. on p. 213).

Prasse, Bastian and Piet Van Mieghem (2020b). "Predicting dynamics on networks hardly depends on the topology". In: *arXiv preprint arXiv:2005.14575* (cit. on p. 191).

Prem, Kiesha, Yang Liu, Timothy W Russell, et al. (2020). "The effect of control strategies to reduce social mixing on outcomes of the COVID-19 epidemic in Wuhan, China: a modelling study". In: *The Lancet Public Health* (cit. on p. 127).

Pujari, Bhalchandra S and Snehal M Shekatkar (2020). "Multi-city modeling of epidemics using spatial networks: Application to 2019-nCov (COVID-19) coronavirus in India". In: *medRxiv* (cit. on p. 130).

Raponi, Simone, Zeinab Khalifa, Gabriele Oligeri, and Roberto Di Pietro (2022). "Fake News Propagation: A Review of Epidemic Models, Datasets, and Insights". In: *ACM Transactions on the Web (TWEB)* (cit. on p. 5).

Reich, Ofir, Guy Shalev, and Tom Kalvari (2020). "Modeling COVID-19 on a network: super-spreaders, testing and containment". In: *medRxiv* (cit. on p. 130).

Riou, Julien and Christian L Althaus (2020). "Pattern of early human-to-human transmission of Wuhan 2019 novel coronavirus (2019-nCoV), December 2019 to January 2020". In: *Eurosurveillance* 25.4 (cit. on p. 123).

Roda, Weston C, Marie B Varughese, Donglin Han, and Michael Y Li (2020). "Why is it difficult to accurately predict the COVID-19 epidemic?" In: *Infectious Disease Modelling* (cit. on p. 128).

Rossini, PM, Riccardo Di Iorio, M Bentivoglio, et al. (2019). "Methods for analysis of brain connectivity: An IFCN-sponsored review". In: *Clinical Neurophysiology* 130.10, pp. 1833–1858 (cit. on p. 190).

Röst, Gergely, Zsolt Vizi, and István Z Kiss (2016). "Impact of non-Markovian recovery on network epidemics". In: *BIOMAT 2015: International Symposium on Mathematical and Computational Biology*. World Scientific, pp. 40–53 (cit. on p. 97).

Sahneh, Faryad Darabi, Aram Vajdi, Heman Shakeri, Futing Fan, and Caterina Scoglio (2017). "GEMFsim: a stochastic simulator for the generalized epidemic modeling framework". In: *Journal of computational science* 22, pp. 36–44 (cit. on p. 48).

Sambaturu, Prathyush and Anil Vullikanti (2019). “Designing Robust Interventions to Control Epidemic Outbreaks”. In: *International Conference on Complex Networks and Their Applications*. Springer, pp. 469–480 (cit. on p. 105).

Sanft, Kevin R, Sheng Wu, Min Roh, et al. (2011). “StochKit2: software for discrete stochastic simulation of biochemical systems with events”. In: *Bioinformatics* 27.17, pp. 2457–2458 (cit. on p. 168).

Sanguinetti, Guido (2020). “Systematic errors in estimates of R_t from symptomatic cases in the presence of observation bias”. In: *arXiv preprint arXiv:2012.02105* (cit. on p. 126).

Sarraf, Saman and Jian Sun (2016). “Advances in functional brain imaging: a comprehensive survey for engineers and physical scientists”. In: *International Journal of Advanced Research* 4.8, pp. 640–660 (cit. on p. 191).

Schneider, Christian M, Tamara Mihaljev, Shlomo Havlin, and Hans J Herrmann (2011). “Suppressing epidemics with a limited amount of immunization units”. In: *Physical Review E* 84.6, p. 061911 (cit. on p. 105).

Schnoerr, David, Guido Sanguinetti, and Ramon Grima (2018). “Approximation and inference methods for stochastic biochemical kinetics - a tutorial review”. In: *Journal of Physics A* 51, p. 169501 (cit. on p. 184).

Shen, Chen, Nassim Nicholas Taleb, and Yaneer Bar-Yam (2020). “Review of Ferguson et al “Impact of nonpharmaceutical interventions..”” In: *New England Complex Systems Institute* (cit. on p. 123).

Sherborne, N, JC Miller, KB Blyuss, and IZ Kiss (2016). “Mean-field models for non-Markovian epidemics on networks: from edge-based compartmental to pairwise models”. In: *arXiv preprint arXiv:1611.04030* (cit. on p. 84).

Silva, Cristiana J, Guillaume Cantin, Carla Cruz, et al. (2020). “Complex network model for COVID-19: human behavior, pseudo-periodic solutions and multiple epidemic waves”. In: *arXiv preprint arXiv:2010.02368* (cit. on p. 130).

Simon, Péter L, Michael Taylor, and Istvan Z Kiss (2011). “Exact epidemic models on graphs using graph-automorphism driven lumping”. In: *Journal of mathematical biology* 62.4, pp. 479–508 (cit. on p. 169).

Singh, Abhyudai and João P Hespanha (2010). “Stochastic hybrid systems for studying biochemical processes”. In: *Royal Society A* 368.1930, pp. 4995–5011 (cit. on p. 168).

Singh, Rajesh and Ronojoy Adhikari (2020). “Age-structured impact of social distancing on the COVID-19 epidemic in India”. In: *arXiv preprint arXiv:2003.12055* (cit. on p. 127).

Slavtchova-Bojkova, M. (2020). “Branching processes modelling for coronavirus (COVID’19) pandemic”. In: *13th International Conference on Information Systems and Grid Technologies, ISGT 2020* 2656 (cit. on p. 129).

Softky, William R and Christof Koch (1993). “The highly irregular firing of cortical cells is inconsistent with temporal integration of random EPSPs”. In: *Journal of Neuroscience* 13.1, pp. 334–350 (cit. on p. 69).

Soltani, Mohammad, Cesar Augusto Vargas-Garcia, and Abhyudai Singh (2015). “Conditional moment closure schemes for studying stochastic dynamics of genetic circuits”. In: *IEEE transactions on biomedical circuits and systems* 9.4, pp. 518–526 (cit. on pp. 168, 187).

Song, Chonggang, Wynne Hsu, and Mong Li Lee (2015). “Node immunization over infectious period”. In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 831–840 (cit. on pp. 105, 115).

Sporns, Olaf (2013). “Structure and function of complex brain networks”. In: *Dialogues in clinical neuroscience* 15.3, p. 247 (cit. on p. 7).

Starnini, Michele, James P Gleeson, and Marián Boguñá (2017). “Equivalence between non-Markovian and Markovian dynamics in epidemic spreading processes”. In: *Physical review letters* 118.12, p. 128301 (cit. on p. 84).

Stewart, GW and JH Miller (1975). “Methods of simultaneous iteration for calculating eigenvectors of matrices”. In: *Topics in Numerical Analysis II*, pp. 169–185 (cit. on p. 111).

Stutt, Richard OJH, Renata Retkute, Michael Bradley, Christopher A Gilligan, and John Colvin (2020). “A modelling framework to assess the likely effectiveness of facemasks in combination with ‘lock-down’in managing the COVID-19 pandemic”. In: *Proceedings of the Royal Society A* 476.2238, p. 20200376 (cit. on p. 127).

Sun, Kaiyuan, Wei Wang, Lidong Gao, et al. (2020). “Transmission heterogeneities, kinetics, and controllability of SARS-CoV-2”. In: *Science*. eprint: `https://science.sciencemag.org/content/early/2020/11/23/science.abe2424.full.pdf` (cit. on pp. 123, 148).

Szabó, György and Gabor Fath (2007). “Evolutionary games on graphs”. In: *Physics reports* 446.4-6, pp. 97–216 (cit. on p. 205).

Tang, Lu, Yiwang Zhou, Lili Wang, et al. (2020). “A Review of Multi-Compartment Infectious Disease Models”. In: *International Statistical Review* 88.2, pp. 462–513 (cit. on p. 127).

Tariq, Amna, Yiseul Lee, Kimberlyn Roosa, et al. (2020). “Real-time monitoring the transmission potential of COVID-19 in Singapore, March 2020”. In: *BMC Medicine* 18, pp. 1–14 (cit. on p. 123).

Tibshirani, Robert (1996). “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1, pp. 267–288 (cit. on p. 213).

Tong, Hanghang, B Aditya Prakash, Charalampos Tsourakakis, et al. (2010). “On the vulnerability of large graphs”. In: *2010 IEEE International Conference on Data Mining*. IEEE, pp. 1091–1096 (cit. on p. 105).

Trivedi, Kishor S (2008). *Probability & statistics with reliability, queuing and computer science applications*. John Wiley & Sons (cit. on p. 37).

Truccolo, Wilson (2010). “Stochastic models for multivariate neural point processes: Collective dynamics and neural decoding”. In: *Analysis of parallel spike trains*. Springer, pp. 321–341 (cit. on p. 98).

Tuite, Ashleigh R and David N Fisman (2020). “Reporting, epidemic growth, and reproduction numbers for the 2019 novel coronavirus (2019-nCoV) epidemic”. In: *Annals of Internal Medicine* 172.8, pp. 567–568 (cit. on p. 129).

Van Kampen, Nicolaas Godfried (1992). *Stochastic processes in physics and chemistry*. Vol. 1. Elsevier (cit. on p. 168).

Van Mieghem, Piet (2010). *Graph spectra for complex networks*. Cambridge University Press (cit. on pp. 10, 162).

– (2016). “Universality of the SIS prevalence in networks”. In: *arXiv preprint arXiv:1612.01386* (cit. on p. 6).

Van Mieghem, Piet, Jasmina Omic, and Robert Kooij (2008). “Virus spread in networks”. In: *IEEE/ACM Transactions On Networking* 17.1, pp. 1–14 (cit. on p. 33).

Vardi, Moshe Y (1985). “Automatic verification of probabilistic concurrent finite state programs”. In: *26th Annual Symposium on Foundations of Computer Science (SFCS 1985)*. IEEE, pp. 327–338 (cit. on p. 34).

Vázquez, Alexei, Joao Gama Oliveira, Zoltán Dezsö, et al. (2006). “Modeling bursts and heavy tails in human dynamics”. In: *Physical Review E* 73.3, p. 036127 (cit. on p. 69).

Vestergaard, Christian L and Mathieu Génois (2015). “Temporal gillespie algorithm: Fast simulation of contagion processes on time-varying networks”. In: *PLoS computational biology* 11.10, e1004579 (cit. on p. 59).

Walker, Ann Sarah, Emma Pritchard, Thomas House, et al. (2020). “Viral load in community SARS-CoV-2 cases varies widely and temporally”. In: *medRxiv* (cit. on p. 123).

Walsh, Kieran A, Karen Jordan, Barbara Clyne, et al. (2020). “SARS-CoV-2 detection, viral load and infectivity over the course of an infection: SARS-CoV-2 detection, viral load and infectivity”. In: *Journal of Infection* (cit. on p. 123).

Ward, Jonathan A and John Evans (2018). “A General Model of Dynamics on Networks with Graph Automorphism Lumping”. In: *International Workshop on Complex Networks and their Applications*. Springer, pp. 445–456 (cit. on pp. 169, 181).

Watts, Duncan J, Roby Muhamad, Daniel C Medina, and Peter S Dodds (2005). “Multiscale, resurgent epidemics in a hierarchical metapopulation model”. In: *Proceedings of the National Academy of Sciences* 102.32, pp. 11157–11162 (cit. on p. 127).

Wei, James and James CW Kuo (1969). “Lumping analysis in monomolecular reaction systems. Analysis of the exactly lumpable system”. In: *Industrial & Engineering chemistry fundamentals* 8.1, pp. 114–123 (cit. on p. 168).

Welton, Thomas, Daniel A Kent, Dorothee P Auer, and Robert A Dineen (2015). “Reproducibility of graph-theoretic brain network metrics: a systematic review”. In: *Brain connectivity* 5.4, pp. 193–202 (cit. on p. 8).

Whitt, Ward (2006). “Continuous-time Markov chains”. In: *Dept. of Industrial Engineering and Operations Research, Columbia University, New York* (cit. on p. 164).

Wijayanto, Arie Wahyu and Tsuyoshi Murata (2019). “Effective and scalable methods for graph protection strategies against epidemics on dynamic networks”. In: *Applied Network Science* 4.1, p. 18 (cit. on p. 105).

– (2017). “Flow-aware vertex protection strategy on large social networks”. In: *2017 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*. IEEE, pp. 58–63 (cit. on p. 105).

– (2018). “Learning adaptive graph protection strategy on dynamic networks via reinforcement learning”. In: *2018 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*. IEEE, pp. 534–539 (cit. on p. 105).

Wilson, Nick, Lucy Telfar Barnard, Amanda Kvalsvig, and Michael Baker (2020). “Potential Health Impacts from the COVID-19 Pandemic for New Zealand if Eradication Fails: Report to the NZ Ministry of Health”. In: *Government Report* (cit. on p. 127).

Wolfram, Christopher (2020). “An agent-based model of Covid-19”. In: *Complex Syst* 29, pp. 87–105 (cit. on p. 130).

Wu, Kai, Jing Liu, and Dan Chen (2019). “Network reconstruction based on time series via memetic algorithm”. In: *Knowledge-Based Systems* 164, pp. 404–425 (cit. on p. 214).

Wu, Weichang, Huanxi Liu, Xiaohu Zhang, Yu Liu, and Hongyuan Zha (2019). “Modeling Event Propagation via Graph Biased Temporal Point Process”. In: *arXiv preprint arXiv:1908.01623* (cit. on p. 98).

Yan, Gang, Petra E Vértes, Emma K Towlson, et al. (2017). “Network control principles predict neuron function in the Caenorhabditis elegans connectome”. In: *Nature* 550.7677, pp. 519–523 (cit. on p. 7).

Yanev, Nikolay M, Vessela K Stoimenova, and Dimitar V Atanasov (2020). “Branching stochastic processes as models of Covid-19 epidemic development”. In: *arXiv preprint arXiv:2004.14838* (cit. on p. 129).

Yang, GL (1972). “Empirical study of a non-Markovian epidemic model”. In: *Mathematical Biosciences* 14.1-2, pp. 65–84 (cit. on p. 69).

Young, George F, Luca Scardovi, Andrea Cavagna, Irene Giardina, and Naomi E Leonard (2013). “Starling flock networks manage uncertainty in consensus at low cost”. In: *PLoS computational biology* 9.1, e1002894 (cit. on p. 5).

Zhang, Hai-Feng, Fang Xu, Zhong-Kui Bao, and Chuang Ma (2018). “Reconstructing of networks with binary-state dynamics via generalized statistical inference”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 66.4, pp. 1608–1619 (cit. on p. 191).

Zhang, Yan, Yu Guo, Zhang Zhang, et al. (2021). “Automated Discovery of Interactions and Dynamics for Large Networked Dynamical Systems”. In: *arXiv preprint arXiv:2101.00179* (cit. on pp. 210, 213).

Zhang, Yao and B Aditya Prakash (2015). “Data-aware vaccine allocation over large networks”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 10.2, pp. 1–32 (cit. on pp. 105, 108, 116).

Zhang, Yunjun, Yuying Li, Lu Wang, Mingyuan Li, and Xiaohua Zhou (2020). “Evaluating transmission heterogeneity and super-spreading event of COVID-19 in a metropolis of China”. In: *International Journal of Environmental Research and Public Health* 17.10, p. 3705 (cit. on p. 129).

Zhang, Zhang, Yi Zhao, Jing Liu, et al. (2019). “A general deep learning framework for network reconstruction and dynamics learning”. In: *Applied Network Science* 4.1, pp. 1–17 (cit. on pp. 190, 205, 210, 213, 268).

Zitnik, Marinka, Monica Agrawal, and Jure Leskovec (2018). “Modeling polypharmacy side effects with graph convolutional networks”. In: *Bioinformatics* 34.13, pp. i457–i466 (cit. on p. 190).

List of Algorithms

List of Models

List of Figures

Part V

Appendix

Brightening BLUE

A

A.1 Direct MPM Construction

Here, we propose a way to directly derive the lumped MPMs from the contact network without building the original CTMC first. Consider Figure 8.1 (p. 172) again. The goal is to go from (a) (model specification) directly to (c) (lumped MPM) and to skip (b) (full CTMC).

A.1.1 Node-Based Abstraction with General Rate Functions

We start with the node-based counting abstraction. Consider a given contact network, partitioning, and rules. Constructing the reduced state space $\mathcal{Y}$ is straight-forward (cf. Eq. (8.2)). The non-obvious part is the computation of the rates (without iterating over the original CTMC states).

Assume we are interested in the rates of a specific lumped state $\mathbf{y} \in \mathcal{Y}$. Our general strategy is to iterate over the nodes in the contact network and to compute the mean rate attributed to that node over all $\mathbf{x} \in \mathcal{X}$ that belong to $\mathbf{y}$. Therefore, we consider the possible states of each node together with all possible species neighborhoods. The probability of a node v being in state s and having species neighborhood $\mathbf{w}$ is denoted as $\mathrm{P}\big(X(v) = s, W(v) = \mathbf{w}\big)$ (cf. Example A.1).

Example A.1: Computation of Neighborhood Probabilities

Consider again Figure 8.1 (p. 172). Assume $\mathbf{y}$ tells us that two nodes in P_1 and zero nodes in P_2 are infected. We want to compute the infection rate belonging to P_2. Both nodes in P_2 will have exactly one infected (in P_1) and one susceptible neighbor (in P_2), and the probability that the node itself is susceptible is 1.
In contrast, when both partitions contain exactly one infected node, we have four possible network-states that follow this description. When we fix a node v, the node-state and the immediate neighborhood of that node can be described as a probability distribution (over $\mathcal{S}$ and $\mathbf{w}$, respectively), assuming uniformity over the four network-states.

For a specific rule $r = s_1 \xrightarrow{f(\cdot)} s_2$ and partition P, we can then re-write Eq. (8.2) such that the propensity becomes:

$$\alpha_{r,P}(\mathbf{y}) = \sum_{v \in P} \sum_{\mathbf{w} \in \mathcal{W}_v} f(\mathbf{m}_\mathbf{w}) P\big(X(v) = s_1, W(v) = \mathbf{w}\big) \,,$$

where $\mathbf{m}_\mathbf{w}$ is the neighborhood vector induced by $\mathbf{w}$, which we receive by grouping all partitions together[1]. Note that it is not computationally necessary to iterate over all nodes in the partition. Instead, we can group all nodes with the same partition neighborhood together, that is, all nodes $v', v'' \in P$ with $W_{v'} = W_{v''}$ as the probability only depends on $\mathbf{w}$.

The calculation of the probability is the interesting part. We start by establishing that

$$\begin{aligned} &P\Big(X(v) = s_1, V(v) = \mathbf{w}\Big) \\ =&P\Big(X(v) = s_1\Big) \cdot P\Big(V(v) = \mathbf{w} \;\Big|\; X(v) = s_1\Big) \,. \end{aligned}$$

[1] For instance, in Figure 8.1c (p. 172), we would sum up each column.

The first term in the product can be described simply by dividing the number of s_1-nodes in P by the total number of nodes in P.

$$\mathrm{P}\Big(X(v) = s_1\Big) = \frac{\mathbf{y}[s_1, P]}{|P|} \quad \text{where: } v \in P \ .$$

The conditional probability can be calculated for each partition independently. This is because we know the number of nodes in each state in each partition (cf. Example A.2).

We also know that in partition P, the current node is already in state s_1, which we must consider. First, we define $\mathbf{y}_P \in \mathbb{Z}_{\geqslant 0}^{|\mathcal{S}|}$ as the projection from $\mathbf{y}$ to P. Thus, each entry is defined by:

$$\mathbf{y}_P[s] = \mathbf{y}[s, P] \ .$$

Likewise, we define $\mathbf{v}_P \in \mathbb{Z}_{\geqslant 0}^{|\mathcal{S}|}$, such that $\mathbf{v}_P[s] = \mathbf{v}[s, P]$. We also define $W(v)_P \in \mathbb{Z}_{\geqslant 0}^{|\mathcal{S}|}$ to be the number of neighbors of node v in partition P for each state. Finally, we define $\mathbf{y}_P^{s_1^-}$ to be the same vector as $\mathbf{y}_P$ except that the entry corresponding to state s_1 is subtracted by one (and truncated at zero). We can now rewrite the probability as:

$$\begin{aligned}
&\mathrm{P}\Big(W(v) = \mathbf{w} \mid X(v) = s_1\Big) \\
=&\prod_{P' \in \mathcal{P}} \mathrm{P}\Big(W(v)_{P'} = \mathbf{v}_{P'} \;\Big|\; X(v) = s_1\Big) \\
=&\mathrm{P}\Big(W(v)_P = \mathbf{v}_P \;\Big|\; X(v) = s_1\Big) \prod_{P' \in \mathcal{P} \setminus \{P\}} \mathrm{P}\Big(W(v)_{P'} = \mathbf{v}_{P'} \;\Big|\; X(v) = s_1\Big) \\
=&\mathrm{p_h}\Big(\mathbf{v}_P; \mathbf{y}_P^{s_1^-}\Big) \cdot \prod_{P' \in \mathcal{P} \setminus \{P\}} \mathrm{p_h}\Big(\mathbf{v}_{P'}; \mathbf{y}_{P'}\Big) \ .
\end{aligned} \tag{A.1}$$

We use $\mathrm{p_h}(\mathbf{k}; \mathbf{K})$ to denote the probability mass function of the multivariate hypergeometric distribution, where $\mathbf{k}, \mathbf{K}$ denote vectors over

non-negative integers of the same length. That is, if $\mathbf{K}$ denotes the number of nodes in each state in a partition (resp., the number of marbles in an urn with different colors), then $p_h(\mathbf{k}; \mathbf{K})$ denotes the probability of drawing exactly $\mathbf{k}[s]$ nodes (resp. marbles) of each state (resp. color).

Example A.2: Drawing Neighborhood Probabilities

Consider again Figure 8.1 (p. 172). Assume P_1 has one infected and one susceptible node. When we know that a node $v \in P_1$ is infected ($X(v) = \mathtt{S}$), then the other node in P_1 has to be susceptible. However, conditioning on $X(v) = \mathtt{S}$ does not change the probability that a neighboring node in P_2 occupies a particular node-state. This is why we factor out the current partition in Eq. (A.1).

A.1.2 Reaction Networks and Linear Models

A special case of MPMs are biochemical reaction networks, where the species represent different types of molecules. The change vectors and corresponding propensity functions can elegantly be expressed as monomolecular ($\mathtt{A} \to \mathtt{B}$) and bimolecular ($\mathtt{A} + \mathtt{B} \to \mathtt{C} + \mathtt{D}$) reaction rules ($\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D} \in \mathcal{Z}$).

Reduction to Biochemical Reaction Networks

Most classical models in computational epidemiology are composed solely of *node-based* rules (like the curing rule) and *edge-based* rules (like the infection propagation rule). We call these *linear models*. Node-based rules, also known as *spontaneous* or *independent* rules, have a constant rate function, i.e., $f(\mathbf{m}) = \mu$. Edge-based rules, also

referred to as *contact* rules, are linear in exactly one dimension, i.e., they have the form $f(\mathbf{m}) = \lambda\mathbf{m}[s]$.

Linear models are special because it is not the whole neighborhood that is important for the rate of a rule but only the expected number of neighbors in a particular state. This makes the rule very similar to monomolecular and bimolecular reaction rates in MPMs. In fact, we can model the whole dynamics as a set of reactions over the species $\mathcal{Z}$.

Chemical reaction networks are a subclass of MPMs. In a chemical reaction network the state space is given by population vectors over species, and molecular reactions have the form $\mathtt{A} \xrightarrow{a} \mathtt{C}$ or $\mathtt{A}+\mathtt{B} \xrightarrow{b} \mathtt{C}+\mathtt{D}$, where $\mathtt{A},\mathtt{B},\mathtt{C},\mathtt{D}$ denote species and $a, b \in \mathbb{R}_{\geqslant 0}$ are reaction rate constants.

For each node-based rule $s_1 \xrightarrow{\mu} s_2$, we construct the reactions

$$(s_1, P) \xrightarrow{\mu} (s_2, P) \qquad \forall P \in \mathcal{P}$$

For each edge-based rule $s_1 \xrightarrow{f(\cdot)} s_1$, $f(\mathbf{m}) = \lambda\mathbf{m}[s']$, we construct the reactions

$$(s_1, P) + (s', P') \xrightarrow{\lambda w_{P,P'}} (s_2, P) + (s', P') \qquad \forall P, P' \in \mathcal{P}$$

where $w_{P,P'}$ denotes the mean number of edges of a random node in P with nodes in P', that is[2]:

$$w_{P,P'} = \begin{cases} \frac{\epsilon(P,P)}{|P|}\frac{1}{|P|-1} & \text{if } P = P' \\ \frac{\epsilon(P,P')}{|P|}\frac{1}{|P'|} & \text{otherwise} \end{cases} .$$

with

$$\epsilon(P, P') = |\{(n_1, n_2) \in \mathcal{E} \mid n_1 \in P, n_2 \in P'\}| . \tag{A.2}$$

[2]Note that, despite the tuple notation, we only count edges once

A.1.3 Edge-Based Counting Abstraction

For each rule $r = s_1 \xrightarrow{f(\cdot)} s_2$, and each partition $P \in \mathcal{P}$, and each $\mathbf{w} \in \mathcal{W}_P$, we define a propensity function $\alpha_{r,P,\mathbf{w}}(\cdot)$ with:

$$\alpha_{r,P,\mathbf{w}}(\mathbf{y}) = \sum_{v \in P} f(\mathbf{m}_{\mathbf{w}}) P\big(X(v) = s_1, W(v) = \mathbf{w}\big) \,.$$

Again, we use:

$$P\big(X(v) = s_1, W(v) = \mathbf{w}\big) = P\big(X(v) = s_1\big) \cdot P\big(W(v) = \mathbf{w} \mid X(v) = s_1\big)$$

to compute this probability, where we can solve $P\big(X(v) = s_1\big)$ exactly as before.

Since we now have information about the edges, we can derive the probability of neighborhoods more precisely. In fact, we can directly construct the set of candidate neighbors from $\mathbf{y}$. Therefore, we define a vector $\mathbf{y}_{s,P,P'} \in \mathbb{Z}_{\geqslant 0}^{|S|}$, where entry $\mathbf{y}_{s,P,P'}[s']$ specifies the number of neighbors of a random node in state s and partition P, which lie in partition P' and occupy state s'. Formally:

$$\mathbf{y}_{s,P,P'}[s'] = \begin{cases} \mathbf{y}[s, P, s', P'] & \text{if } (s, P) \leqslant (s', P') \\ \mathbf{y}[s', P', s, P] & \text{otherwise} \end{cases} \,.$$

This gives rise to the final approximation of the probability of neighborhood species:

$$P\Big(W(v) = \mathbf{w} \;\Big|\; X(v) = s_1\Big) \approx \prod_{P' \in \mathcal{P}} p_h\big(\mathbf{w}_P; \mathbf{y}_{s_1,P,P'}\big) \quad \text{(where } v \in P\text{.)}$$

Technicalities of TEAL

B

B.1 TEAL's Training

We start with a sharpness parameter $\nu = 5.0$ and increase ν after 50 epochs by 0.3. We train (maximally) for 5000 epochs, but we employ early stopping if the underlying (binarized) graph does not change for 500 epochs (measured each 50 epochs). Moreover, we use Pytorch's Adam optimizer with an initial learning rate of 10^{-3}.

We use a mini-batch size of 100. For an efficient forward pass, we first stack the 100 snapshots horizontally, yielding a $n \times 100|\mathcal{S}|$ matrix. We push it through the graph layer and get another $n \times 100|\mathcal{S}|$ matrix. We then reshape it, yielding a $100n \times |\mathcal{S}|$ matrix, and apply the prediction layer row-wise.

In **Experiment 1**, we use a fixed (pre-defined) binarized adjacency matrix and optimize only the weights of the prediction layer during training.

In contrast to standard GNN software (like Pytorch Geometric), we do not use a sparse representation of the underlying graph because we optimize over all entries in the adjacency matrix. Moreover, our training set consists of snapshots rather than graphs, so we do not use any sort of graph batching as it is common in GNN training.

We did not utilize hyperparameter optimization. Using a validation set to optimize the aforementioned parameters would likely increase the performance of TEAL.

B.2 Dynamical Models

Except for the CML, we use continuous-time stochastic processes with a discrete state space to generate snapshots. Specifically, these models have a CTMC semantics and can be formulated using the multi-state process family (cf. Chapter 2.4.3).

SIS. We test a classical `SIS` model. For all models, we add a small amount of stochastic noise ϵ to the dynamics. The noise not only mimics measurement errors but also prevents the system from getting stuck in *trap state* where no rule is applicable (cf. Section 7.1).

In the sequel, we use the corresponding notation:

$$\mathtt{I} \xrightarrow{\alpha+\epsilon} \mathtt{S} \qquad \mathtt{S} \xrightarrow{\beta \mathbf{m}[\mathtt{I}]+\epsilon} \mathtt{I}\,.$$

The parameterization is $\alpha = 2.0$, $\beta = 1.0$, and $\epsilon = 0.1$.

Inverted Voter. This model describes two competing opinions (`A` and `B`) while nodes always tend to maximize their disagreement with their neighbors.

$$\mathtt{A} \xrightarrow{\mathbf{m}[\mathtt{A}]+\epsilon} \mathtt{B} \qquad \mathtt{B} \xrightarrow{\mathbf{m}[\mathtt{B}]+\epsilon} \mathtt{A}\,.$$

We use $\epsilon = 0.01$.

Game of Life. Nodes represent cells (resp. areas) that are either dead (`D`) (resp. unpopulated) or alive (`A`) (resp. populated). Living conditions are good when roughly half of the neighboring cells are

alive. Otherwise, a cell tends to die due to either over- or underpopulation.

$$\mathtt{A} \xrightarrow{|\mathbf{m}[\mathtt{A}]-\mathbf{m}[\mathtt{D}]|+\epsilon} \mathtt{D} \qquad \mathtt{D} \xrightarrow{k-|\mathbf{m}[\mathtt{A}]-\mathbf{m}[\mathtt{D}]|+\epsilon} \mathtt{A}\,,$$

where $k = \mathbf{m}[\mathtt{A}] + \mathbf{m}[\mathtt{D}]$ is the degree of the node. We use $\epsilon = 0.01$.

Rock Paper Scissors. This model mimics a simple evolutionary process where three species compete and defeat each other in a ring-like relationship.

$$\mathtt{R} \xrightarrow{\mathbf{m}[\mathtt{P}]+\epsilon} \mathtt{P} \qquad \mathtt{P} \xrightarrow{\mathbf{m}[\mathtt{S}]+\epsilon} \mathtt{S} \qquad \mathtt{S} \xrightarrow{\mathbf{m}[\mathtt{R}]+\epsilon} \mathtt{R}\,.$$

We use $\epsilon = 0.01$.

Forest Fire. Spots/nodes are either empty (E), on fire (F), or have a tree on them (T). Trees grow at a growth rate g. Random lightning starts a fire on tree-nodes with rate $f_{\mathtt{start}}$. The fire on a node goes extinct with rate $f_{\mathtt{end}}$, leaving the node empty. Finally, fire spreads to neighboring tree-nodes with rate $f_{\mathtt{spread}}$.

$$\mathtt{T} \xrightarrow{f_{\mathtt{start}}+\mathbf{m}[\mathtt{F}]\,f_{\mathtt{spread}}+\epsilon} \mathtt{F} \qquad \mathtt{F} \xrightarrow{f_{\mathtt{end}}+\epsilon} \mathtt{E} \qquad \mathtt{E} \xrightarrow{g+\epsilon} \mathtt{T}\,.$$

The parameterization is $g = 1.0$, $f_{\mathtt{start}} = 0.1$, $f_{\mathtt{end}} = 2.0$, $f_{\mathtt{spread}} = 2.0$, and $\epsilon = 0.1$.

Coupled Map Lattice. Let x_i be the value of node v_i at time-step i. Each node starts with a random value (uniform in $[0, 1]$). At each time step, all nodes are updated based on a linear combination of

their node-value and the node-values of neighboring nodes [Kaneko, 1992]:

$$x_{i+1} = (1.0 - s)f(x_i) + \frac{s}{k_i} \sum_{j \in N(i)} f(x_j) \ ,$$

where k_i is the degree of v_i, $N(i)$ denotes the set of (indices of) nodes adjacent to v_i, s is the coupling strength and $f(\cdot)$ is the local *map*. Like [Z. Zhang et al., 2019], we use the logistic function [May, 2004]:

$$f(x) = r \cdot x \cdot (1.0 - x) \ .$$

where r modulates the complexity of the dynamics.

We use $s = 0.1$ and $r = 3.57$.

B.3 Random Graphs Generation

We use the Python NetworkX package [Hagberg, Swart, et al., 2008] to generate a single instance (variate) of a random graph model and test TEAL and the baselines on a large number of snapshots generated using this graph. In particular, we use Erdős-Renyi (ER) ($N = 22$, $|E| = 33$) graph model with connection probability 0.15:

```
nx.erdos_renyi_graph(22, 0.15, seed=42).
```

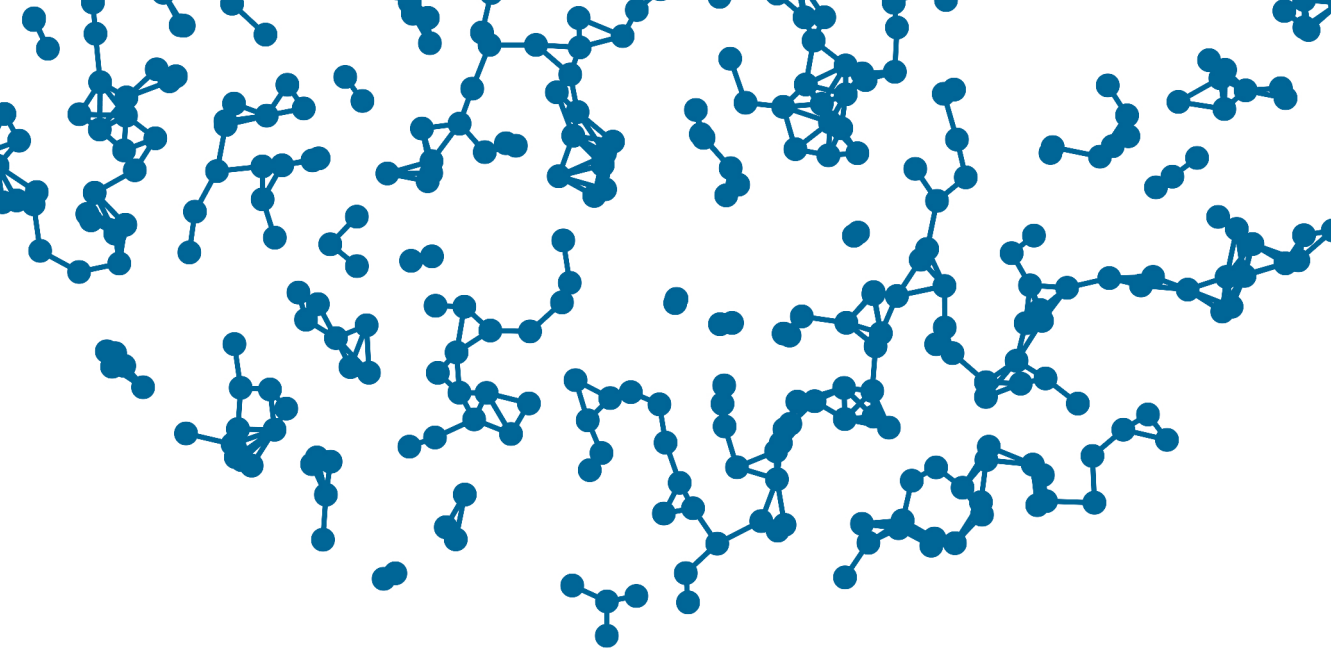
We also use a Geometric graph ($N = 50$, $|E| = 278$):

```
nx.random_geometric_graph(50, 0.3, seed=42)
```

and a Watts–Strogatz graph ($N = 30$, $|E| = 69$) where each node has 4 neighbors and the re-wiring probability is 0.15:

```
nx.newman_watts_strogatz_graph(30, 4, 0.15, seed=42).
```

After generation, the node-ids are randomly shuffled in order to guarantee that they do not leak information about connectivity to the training model.



Gerrit Großmann
Saarbrücken
2022