Hybrid algorithm for efficient simulation of spreading processes on adaptive networks

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften (Dr. rer. nat.)

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Nadezhda Malysheva

Berlin 2024

Erstgutachter/in: Prof. Dr. Max von Kleist

Zweitgutachter/in: Prof. Dr. Susanna Röblitz

Tag der Disputation: 11.11.2024

Declaration of authorship

Name: Malysheva

First name: Nadezhda

I declare to the Freie Universität Berlin that I have completed the submitted dissertation independently and without the use of sources and aids other than those indicated. The present thesis is free of plagiarism. I have marked as such all statements that are taken literally or in content from other writings. This dissertation has not been submitted in the same or similar form in any previous doctoral procedure.

I agree to have my thesis examined by a plagiarism examination software.

Date: 11.06.2024

Acknowledgements

First and foremost, I would like to thank my advisor, Max, who was always available whenever I needed advice or guidance and for believing in me, even when challenges came up. Your support has been invaluable throughout my academic journey.

I would also like to extend my heartfelt thanks to the Max-Planck Society for allowing me to be a part of it, and particularly to Kirsten and Anne-Dominique, for their brilliant guidance through a multitude of administrative tasks and their invaluable assistance.

I am also profoundly grateful to my family for their continuous support. To Philipp, whose efforts to organize time and space for me, even in the most challenging environments, have made this thesis possible. To Charlotte, whose presence has been a powerful motivator that I never knew I needed. This inspiration has given me the strength and determination to persevere and complete this journey.

To my friends, I owe a special thank you to Artyom for making this journey possible to begin. To Maureen, who also helped me to proofread my thesis, Alisher, Insa and Evgenia, who have provided me with some of the greatest times and fun I could have ever had. Your friendship has been a source of joy during this challenging journey.

Thank you all for your support and for believing in me.

Zusammenfassung

Die Übertragung verschiedener Phänomene erfolgt, wenn Individuen in einer für die jeweilige Übertragungsroute relevanten Weise interagieren. Der Begriff "Kontakt" ist hierbei je nach Phänomen unterschiedlich definiert – von sexuellen Begegnungen für STIs bis hin zur Nähe für luftübertragene Infektionen. Zur Verbesserung der Simulationsgenauigkeit wurden in früheren Studien verschiedene Netzwerkmodelle entwickelt, die die Dynamik der Krankheitsübertragung und deren Zusammenhang mit Kontaktmustern erfassen.

Wenn sich das Kontakt-Netzwerk langsam verändert und während eines Ausbruchs weitgehend stabil bleibt, kann es als statisch betrachtet werden. In manchen Fällen jedoch müssen sowohl die Dynamik des Netzwerks als auch die Phänomenaussbreitung berücksichtigt werden, besonders wenn beide Prozesse auf ähnlichen Zeitskalen ablaufen und Netzwerkveränderungen die Ausbreitung beeinflussen. Diese Arbeit untersucht ein Szenario, in dem Kontaktänderungen und Phänomenaussbreitung eng verbunden sind. Dabei beeinflussen Ausbreitungsereignisse die Netzwerkstruktur, was die weitere Ausbreitung prägt. Dieses adaptive Verhalten ermöglicht eine realistischere Modellierung und Erfassung des Zusammenspiels von Kontaktmustern und Ausbreitung.

Die Notwendigkeit dieser Arbeit ergibt sich aus dem erforderlichen Bedarf, Präzision und Laufzeit bei der Simulation von Ausbreitungsprozessen auf adaptiven, zeitlich veränderlichen Netzwerken auszugleichen. Die stochastische Natur sowohl der Ausbreitungs- als auch der Kontaktprozesse erschwert die Wahl eines geeigneten Simulationsalgorithmus. Näherungsweise Algorithmen bieten schnelle Berechnungen, können jedoch an Genauigkeit verlieren, während exakte Algorithmen präzise, aber oft ineffizient sind. Besonders die Integration adaptiver Netzwerkreaktionen ist bisher ein kritischer und oft vernachlässigter Aspekt des dynamischen und reaktiven Modellierens in komplexen Systemen.

Abstract

The transmission of different phenomena takes place when individuals interact in ways relevant to the specific transmission route. The definition of "contact" varies depending on the modelled phenomenon. For instance, sexual encounters are relevant for the transmission of sexually transmitted diseases (STDs), while close proximity is significant for airborne infections. To improve the accuracy of the simulation of the spread of phenomena of different origin, researchers have advanced various network models in previous studies. These models aim to better capture the dynamics of disease transmissions and their relationship to contact patterns.

In cases where the contact dynamics occur at a much slower pace than the spreading dynamics, leading to transmission whenever contact is made, it is sufficient to focus exclusively on the contact dynamics. If the network undergoes gradual changes and remains mostly unaffected throughout an outbreak, approximating it as a static network would be suitable. However, there are specific circumstances in which it becomes vital to consider both the dynamic nature of the contact network and the spread of the phenomena. Such consideration becomes particularly relevant when these two processes unfold at comparable timescales, and network modifications can shape the trajectory of the spread. In such circumstances, it is vital to include these changes in the analysis to obtain a comprehensive understanding of the dynamics of the spread. This research work addresses a specific scenario in which the temporal processes of the contact changes and the spreading process are closely interconnected. In this scenario, the occurrence of spreading events directly influences the structure of the network, subsequently influencing the subsequent spread of the disease. This adaptive behaviour enables a more realistic representation of behavioural changes that arise when individuals become aware of their infection and make choices such as self-isolation . This approach allows to capture the interplay between contact patterns and the progression of the spreading process, providing valuable insights into how different phenomena propagate within a population.

The necessity of this work arises from the critical need to balance accuracy and computational efficiency in simulating spreading processes on adaptive, time-evolving networks. The challenge in simulating transmission models on time-evolving adaptive networks stems

from the stochastic nature of both spreading processes and contact behaviour. Choosing an appropriate stochastic simulation algorithm is challenging due to this dual stochasticity. While a range of stochastic simulation algorithms exists, selecting a suitable method is not straightforward. Approximate algorithms offer rapid computation but may compromise simulation accuracy and predictive reliability. Conversely, exact simulation algorithms yield accurate predictions but can suffer from computational inefficiency and protracted simulation times. This often hampers research progress and limits the generation of a sufficient amount of simulation trajectories for robust predictions. A distinct category of algorithms is available that allows for the explicit integration of internal dynamics and the evolution of contact network structures within simulations. However, these algorithms predominantly lack the capability to incorporate adaptive responses, a critical aspect of dynamic and responsive modelling in complex systems.

This thesis presents the development and validation of a novel hybrid algorithm, bridging gaps in current methodologies by combining the exact simulation of spreading dynamics with faster, either exact or approximate, simulations of contact dynamics. This methodology focuses on accurately simulating and predicting spreading dynamics while maintaining reliable statistics of contact behaviour, significantly enhancing computational performance for real-world scenarios.

Overview of the thesis structure

The first chapter of this work focuses on continuous-time Markov processes in modelling the spread of infectious diseases, information, and innovations. This analysis reveals the strength of Markov models in capturing the probabilistic nature of these processes and their ability to account for the temporal dynamics of spread. However, the research also identifies critical gaps in traditional compartmental models, particularly their inability to reflect the heterogeneity of individual behaviour and network structures.

To address these shortcomings, the thesis advocates for the utilisation of agent-based models. These models allow for the simulation of individual actions and interactions within a network, providing insights into how individual behaviours and network characteristics influence the overall dynamics of spreading processes. The second chapter of this thesis examines various approaches to modelling the network of contacts relevant to spreading phenomena and emphasises the importance of incorporating the temporal nature of contact and the adaptive change in contact behaviour into the models.

The third chapter of the thesis conducts an in-depth analysis of the currently available simulation algorithms for stochastic models of different complexity. It evaluates their capacity to accurately simulate the spreading dynamics on complex adaptive contact networks. This

evaluation discusses the practical applicability of these algorithms in modelling real-world scenarios. The chapter provides a critical examination of the limitations and strengths of existing methods. By doing so, it not only highlights the areas where these algorithms excel but also identifies the gaps and challenges that need to be addressed for improved simulation performance. This analysis contributes to the development of more effective and efficient approaches to simulate spreading processes within time-evolving adaptive networks.

The fourth chapter of the thesis introduces the design of a novel algorithm, focusing on its foundational principle: the separation of contact and spreading dynamics. This is achieved by identifying distinct boundaries where the contact dynamics can be independently simulated. Consequently, this division allows for an approximate simulation of contact dynamics, leading to substantial improvements in simulation runtime. Additionally, the chapter provides a theoretical analysis of the algorithm's correctness and its anticipated computational complexity, offering insights into its efficiency and reliability in simulating complex dynamics.

Chapter five presents an application-based evaluation of the algorithm, using a specific example for demonstration. In this evaluation, the results are benchmarked against the stochastic simulation algorithm (SSA), which serves as the ground truth. This comparison reiterates the accuracy of the simulations produced by the novel algorithm. Additionally, a comparison of the runtime of both algorithms highlights the significant computational efficiency gained with the new approach.

Chapter six introduces a hybrid Python/C++ package that implements the algorithm. This chapter provides a detailed description of the main classes and the primary Application Programming Interface (API), offering insights into how users can interact with and utilize the software. Furthermore, it discusses the limitations of the current implementation and explores potential future directions for the software's development.

In chapter seven, the thesis discusses the main contributions and limitations of the study. This final chapter also outlines several potential directions for future research, suggesting avenues through which the work can be expanded or refined. This chapter serves as a bridge connecting the current study to forthcoming inquiries in the field, aiming to inspire and guide subsequent investigations.

Table of contents

List of figures

List of Algorithms

Nomenclature

$\exp(\cdot)$	exponential function
inf	the infimum
$\lambda(x)$	Markov process: jump rate associated with the state $x \in \mathscr{S}$
$\left\langle \dot{-}^k \right\rangle$	k-th central moment
$\langle \cdot \rangle$	average
$\left\langle \cdot^k \right\rangle$	k-th raw moment
$\mathbb{E}$	expectation
$\mathbb{P}$	Probability density function; state density function
$\mathbb{S}$	Markov process: stoichiometry matrix
$\mathbb{V}$	variance
$\mathbf{G}$	Markov process: generator matrix
$\mathbf{I}$	Identity matrix
$\mathbf{P_t}$	Markov process: transition probability matrix
$\mathbf{p_t}$	probability vector
$\mathbf{P}$	Markov process: semigroup
$\mathbf{Q}$	One step transition probability matrix
$\mathscr{B}(a,b)$	Binomial distribution
$\mathscr{N}(\alpha,...)$	Normal distribution

$\mathscr{P}(\alpha)$ Poisson distribution

$\mathscr{S}$ State space

$\mathscr{U}(a,b)$ Uniform distribution

$\mathfrak{P}$ population

$\mu(x,y)$ Markov process: transition rate

ν Markov process: jump size

$\tau(t)$ Markov process: holding time

T_L Look-ahead horizon

Ξ Markov process: Markov propagator

a_0 propensities total

a_k reaction propensity

$B(T_L)$ Upper bound

$Exp(\alpha)$ Exponential distribution

$G(t)$ network

$M_{(\cdot)}$ total number of reactions

$O()$ Complexity

$Pr(\cdot)$ Probability function

R^+ contact dynamics reactions – edge addition reactions

R^- contact dynamics reactions – edge deletion reactions

R^c contact dynamics reactions

R^s populational dynamics reactions

R_k reaction

t time index

T_F Simulation end time

T_i Markov process: Stopping time

Y_i Markov process: Embedded Markov chain

ABM Agent-based model

Extrande Extra Reaction Algorithm for Networks in Dynamic Environments

MJP Markov jump process

N number of nodes/individuals in the population

SIR Susceptible-Infected-Recovered model

SSA Stochastic simulation algorithm

SSATAN-X Stochastic Simulation Algorithm for effective spreading dynamics simulation on Time-evolving Adaptive Networx

STD /STI Sexualy transmitted disease / infection

Chapter 1

Spreading process

1.1 Introduction

The term "spreading process" refers to how phenomena, such as information, influence, or disease spread through a population. The pattern of the spread can be defined by various factors, including the nature of the spreading agent, the structure of the contact network, and the behaviour of individuals within the population. Examples of the spread of such phenomena include the following:

- *Spread of infectious disease:* The spread of pathogens through a population, which can depend on factors such as the virulence of the disease, the contact patterns of individuals, and the effectiveness of preventive measures such as vaccination.
- *Spread of information:* The spread of information, beliefs, and behaviours through a society, which can depend on factors such as the credibility of the information source, the structure of the social network, the social norms of the community, and the influence of opinion leaders.
- *Spread of innovation:* The spread of new ideas or technologies through a population, which can depend on factors such as the worth value of the innovation, the factors promoting adoption, and the characteristics of the population (e.g., age and occupation).

This work focusses on the spreading processes in an epidemiological context. However, the key findings and results can be rather straightforwardly adapted to other relevant contexts of interest.

Analytical and numerical studies of spreading process dynamics have been widely published in the literature. In the past several decades, this field of research has made significant

headway due to advances in modern networks and epidemiological science. Two widely applied approaches to model the spreading process are *deterministic* and *stochastic*.

The deterministic approach requires the assumption that for large populations, average rates of spread without random fluctuations are known. For instance, if 10,000 individuals each have a 90% chance of surviving one year, it can be reasonably assumed that 9,000 of them will survive. Deterministic models generate consistent results with the same inputs, regardless of how many times the model is recalculated. Mathematical characteristics in this case are fixed and not random, and each problem has a single set of values and solutions. The unknown factors in a deterministic model are external to the model, which emphasizes definitive outcomes and does not account for errors.

Stochastic models are used to describe the evolution of a system over time, where the evolution is random and uncertain. These models include random fluctuations induced by factors such as parameter uncertainties or population sizes being too small to apply average rates. For example, consider a population of 20 individuals, each with a 0.8 probability of surviving another year. The average number of survivors would be $20 \times 0.8 = 16$. However, due to the small size of the population, random variations would appear, and a probabilistic representation of the population at the end of the year may be more appropriate. For the given example, a binomial model of the population can be used, providing the probabilities of having from zero up to 20 survivors at the end of the year. This choice of model provides a probability distribution for the number of survivors rather than just an average number. Stochastic modelling is substantially unpredictable, with unknown components incorporated into the model. The model generates numerous answers, estimates, and outcomes. To estimate the behaviour of the modelled system, the same procedure is then repeated multiple times in different settings.

A hybrid modelling approach combining both stochastic and deterministic elements can comprehensively capture the complexities of many real-world systems, particularly those exhibiting both continuous and discrete behaviour. Combining stochastic and deterministic models, a hybrid approach can capture both the random and predictable aspects of a system and better explain its behaviour. For example, a hybrid model could use stochastic methods to model the spread of a disease within a population while incorporating deterministic models to capture the effects of public health interventions or social distancing measures.

Models, either deterministic or stochastic, can be either continuous or discrete in time. In a continuous model, events can occur at any point in time. For instance, the duration between infection and recovery for the individual could be any positive decimal number. In contrast, in a discrete model events are classified within time intervals. For instance, the number of

infections can be tallied in various time intervals: between 0 and 1, between 1 and 5, between 5 and 10, between 10 and 20, and so on (the interval lengths do not need to be identical).

In this study, the interest lies in the modelling of the *stochastic spreading process*. One modelling technique widely used for stochastic modelling is "Markov models" or "Markov processes". The name comes from the mathematician Andrey Markov, who developed the theory of stochastic processes in the early 20^{th} century. Markov processes are a type of stochastic process that models random behaviour over time. It includes two basic components – a set of states and a set of transitions between states – that are used to depict the system's behaviour. They can model diverse behaviour types observed in natural complex systems. The current study will further focus on a continuous-time Markov process, rather than a discrete Markov chain, because continuous-time models are better suited to capture the continuous and dynamic nature of many real-world processes, such as the spread of infectious diseases, the diffusion of innovations, or the flow of information in social networks.

The next sections present the main theoretical framework of continuous-time Markov processes, alongside its application in modelling epidemiological spread.

1.2 Markov jump process

To introduce a mathematical model describing a random process, it is necessary to define the probability space beforehand.

Definition 1.2.1. Also called *probability triple*, *probability space* $(\Omega, \mathscr{A}, Pr)$ is a mathematical construct consisting of three elements:

1. *Sample space* Ω: The set of all possible outcomes.
2. σ*-algebra* $\mathscr{A}$: The collection of all considered events. An event refers to a collection of outcomes within the sample space.
3. *Probability measure* $Pr : \mathscr{A} \mapsto [0,1]$: A probability function that assigns a probability value between 0 and 1 to each event in the event space.

Definition 1.2.2. For a given probability space $(\Omega, \mathscr{A}, Pr)$, a *random variable* X is a measurable function assigning a value from a measurable space $\mathscr{S}$ to each possible outcome of a random event or a process:

$$X : \Omega \mapsto \mathscr{S}.$$

The random variable allows for the probability of each event to be expressed as the probability of the corresponding set of values in the space $\mathscr{S}$:

$$Pr(\omega) = Pr(X(\omega)), \quad \omega \in \Omega, \quad X(\omega) \in \mathscr{S}.$$

Definition 1.2.3. For a given probability space $(\Omega, \mathscr{A}, Pr)$ and a measurable space $\mathscr{S}$, a collection of random variables $X_t = \{X(t) : \Omega \mapsto \mathscr{S}\}$ is called a *stochastic process* with the time-index $t \in [0, \infty)$ and a state space $\mathscr{S}$.
For a particular $\omega \in \Omega$, the set $\{X(t, \omega)\}$, $t \in [0, \infty]$ is called a *trajectory* or *realisation* of the stochastic process X_t.

In other words, the state space is defined using elements reflecting the different values the stochastic process can take, and X_t maps sample space Ω to this state space $\mathscr{S}$. Considering a dynamical system evolving through time, the notion $X(t)$ refers to the state of the system X_t at the time t.

Definition 1.2.4. The vector of probabilities of the process X_t being in each state $x \in \mathscr{S}$ at the initial time $t_0 = 0$,

$$p_0 = \{Pr(X(0) = x)\}, \forall x \in \mathscr{S},$$

is called an *initial distribution*. If, for any particular x, the probability $Pr(X(0) = x) = 1$, then x is called an *initial state*.

It is often assumed that the state of the system X_t at initial time $t_0 = 0$ is known as $X(0) = x_0$, however, states $X(t)$ for $t > 0$ are still random variables representing values observed at time t and could be predicted only probabilistically.

To analyse the evolution of X_t over time, the state of the system at successive time points $t_1 \leq t_2 \leq t_3 \leq ... \leq t_n$ can be investigated. Any realisation of a stochastic process obeys the joint probability:

$$\begin{aligned} &\mathbb{P}(x_n, t_n; x_{n-1}, t_{n-1}; \ldots; x_1, t_1 \mid x_0, t_0) \mathrm{d}x_n \mathrm{d}x_{n-1} \ldots \mathrm{d}x_1 \equiv \\ &Pr(X(t_i) \in [x_i, x_i + \mathrm{d}x_i), x_i \in S, \forall i = \overline{1,n} \mid X(t_0) = x_0). \end{aligned} \tag{1.1}$$

Because $\mathbb{P}$ defined above is a joint probability function, it can be extended:

$$\begin{aligned} &\mathbb{P}(x_1, t_1, x_2, t_2, \ldots, x_n, t_n \mid x_0, t_0) = \mathbb{P}(x_1, t_1 \mid x_0, t_0) \times \\ &\times \mathbb{P}(x_2, t_2 \mid x_1, t_1; x_0, t_0) \times \mathbb{P}(x_3, t_3 \mid x_2, t_2; x_1, t_1; x_0, t_0) \times \ldots \\ &\times \mathbb{P}(x_n, t_n \mid x_{n-1}, t_{n-1}; x_2, t_2; x_1, t_1; x_0, t_0), x_i \in S, \forall i = \overline{1,n}. \end{aligned} \tag{1.2}$$

Equation (1.2) defines a joint probability density function for n random variables. By integrating the function $\mathbb{P}$ over any chosen x_i the function for $n-1$ variables can be achieved; however, in most cases, it is impossible to derive a probability density function for $n+1$ random variables. This complexity makes further comprehensive analysis significantly more intricate and tangled.

In the study of spreading processes, however, a specific subclass of stochastic processes is of particular interest.

Definition 1.2.5. If the state density function $\mathbb{P}$ of the stochastic process X_t satisfies the property

$$\mathbb{P}(x_n,t_n \mid x_{n-1},t_{n-1};x_2,t_2;x_1,t_1;x_0,t_0) = \mathbb{P}\left(x_n,t_n \mid x_{n-1},t_{n-1}\right), \quad \forall t \in [0,\infty), \tag{1.3}$$

then X_t is called the *Markov process*. Equation (1.3) is called the *Markov property*.

The Markov property is also known as the "past-forgetting" or "memoryless" property, which implies that the next value of the process depends only on its current, most recent state, and not on any of its previous states or events or the history of the process.

The joint probability function (1.2) for the Markov process is simplified to the following:

$$\begin{aligned}\mathbb{P}\left(x_1,t_1,x_2,t_2,\ldots,x_n,t_n \mid x_0,t_0\right) &= \mathbb{P}(x_n,t_n \mid x_{n-1},t_{n-1})\times \\ \times\mathbb{P}(x_{n-1},t_{n-1} \mid x_{n-2},t_{n-2}) \times \ldots \times \mathbb{P}(x_2,t_2 \mid x_1,t_1) &\times \mathbb{P}(x_1,t_1 \mid x_0,t_0).\end{aligned}$$

Definition 1.2.6. Markov processes X_t is *time-homogeneous* or *temporally homogeneous* if the right part of equation (1.3) does not explicitly depend on t_i but on the time-increment $t_i - t_{i-1}$

Most Markov processes observed in reality are temporally homogeneous. This trend, however, does not imply that X_t and its density function $\mathbb{P}$ are completely time-independent, yet this dependency can be somewhat simplified:

$$\mathbb{P}(x,t \mid x_0,t_0) = \mathbb{P}(x,t-t_0 \mid x_0,0), \quad x_0,x \in \mathscr{S}.$$

Without the loss of generality, it could be posited that $t_0 = 0$, and the state density function then takes the form

$$\mathbb{P}(x,t \mid x_0) = Pr(X(t) = x \mid X(0) = x_0), \quad x_0,x \in \mathscr{S}. \tag{1.4}$$

Equation (1.4) expresses the probability of jumping from the state x_0 to the state x within a period of time t.

Because temporally homogeneous Markov processes are the central focus of this chapter and this work, the term "Markov process" will further refer to a temporally homogeneous Markov process unless otherwise specified.

Realistically speaking, natural systems and models representing spreading processes do not evolve gradually, but rather in sudden bursts. The process is in state x at time point t and will remain there until time $t' > t$, when it jumps to another state $x' \neq x$. This behaviour motivates the name ***Markov jump process*** (MJP).

Although the general stochastic processes and Markov process definitions provided above assume continuous state space, the main interest of this work lies in a Markov jump process with discrete countable state space S. Moreover, as covered in section 1.4.7, the specific case of the Markov models for the spreading process takes values on non-negative integers, so from now on it will be assumed that $\mathscr{S} = \mathbb{N}^+$.

Markov state density function $\mathbb{P}$ for the stochastic Process X_t with the discrete-valued state space is defined as follows:

$$\mathbb{P}(y,t \mid x) = \mathbb{P}(X(t) = y \mid X(0) = x), \quad x,y \in \mathscr{S}, \quad t \geq t_0.$$

As $\mathbb{P}$ is a probability density function, this function must satisfy two relations:

$$\begin{aligned} &\mathbb{P}(y,t \mid x) \geq 0, \qquad x,y \in \mathscr{S}, \quad t \geq t_0, \\ &\sum_{\forall y \in \mathscr{S}} \mathbb{P}(y,t \mid x) = 1. \end{aligned} \tag{1.5}$$

Moreover, to reassure that no transition happens in zero time, the function $\mathbb{P}$ should also satisfy the following condition:

$$\mathbb{P}(y,0 \mid x) = \delta(y,x), \quad x \in \mathscr{S}, \tag{1.6}$$

where $\delta(y,x)$ is a Kronecker-Delta function and takes value of unity if $x = y$ and zero otherwise.

The Markov Property (1.3) also holds for the discrete state space $\mathscr{S}$.

The state density function $\mathbb{P}$ is also required to satisfy the Chapman–Kolmogorov equation, which implies a consistency condition of $\mathbb{P}$ for the Markov process X_t:

$$\mathbb{P}(x_2,t_2 \mid x_0) = \sum_{\forall x_1 \in S} \mathbb{P}(x_2,t_2 - t_1 \mid x_1)\mathbb{P}(x_1,t_1 \mid x_0), \quad 0 \leq t_1 \leq t_2; \quad x_0,x_1,x_2 \in \mathscr{S}.$$

The Chapman–Kolmogorov equation reads that the probability of transitioning from one state to another over a period of time can be expressed as the sum of probabilities of transitioning through all possible intermediate states.

1.3 Components of the Markov jump process

Although the definition of the Markov process already provides two essential components – state space $\mathscr{S}$ and time index t – two further aspects are of particular interest for a jump Markov process:

- *When* will the system leave its current state? This is described by residual or holding times.
- *What* would be the next state? This is described by the embedded Markov Chain – a discrete-time Markov chain that tells what transitions are made.

Definition 1.3.1. Given that X_t is a Markov jump process on $\mathscr{S}$, then

$$\tau(t) = \inf\{h > 0 : X(t+h) \neq X(t)\}$$

is called *holding times* or *residual lifetimes*. In other words, it refers to the length of time spent in the state $X(t)$.

To preserve the Markov memoryless property, holding times must have an exponential distribution [1, 2]. The conditional distribution of $\tau(t)$ given $X(t) = x$ is exponential with parameter $\lambda(x) \in [0, \infty]$:

$$Pr(\tau(t) > h \mid X(t) = x) = \exp(-\lambda(x)h), \quad h > 0.$$

$\lambda(x)$ is called a ***jump rate*** associated with the state $x \in \mathscr{S}$.

Notably, $\tau(t)$ defines a stopping time, indicating a specific behaviour of interest exhibited by the MJP at that moment – the next jump occurs at the time $t + \tau(t)$.

Given that X_t is in the state x at the time t and given the exponential parameter $\lambda(x)$, the probability that the process will jump away from x to some other state during time interval dt can be expressed as follows:

$$Pr(X(t+dt) \neq x \mid X(t) = x) = \lambda(x)dt.$$

Definition 1.3.2. Consider Markov process X_t being at the state $x \in \mathscr{S}$

1. If $\lambda(x) = 0$, then $Pr(\tau = \infty \mid X(t) = x) = 1$, and x is an *absorbing state*. Hence, the Markov process never leaves the state.

2. If $\lambda(x) \in (0, \infty)$, then $Pr(0 < \tau < \infty \mid X(t) = x) = 1$, and x is a *stable state*. Hence, the Markov process leaves the state after a reasonable amount of time.

3. If $\lambda(x) = \infty$, then $Pr(\tau = 0 \mid X(t) = x) = 1$, and x is a ***instantaneous state***. Hence, the Markov process exists in the state as soon as it enters it.

If absorbing states are quite common in practical applications – for example, in Fig.1.1 absorbing states of the SIR model lack infectious individuals, and neither further infection nor further recovery is possible; in Fig. 1.2 the state $\varnothing$, depicting deceased individuals, is absorbing – instantaneous states are associated with undesirable behaviour and are generally avoided in most scenarios. To ensure this avoidance, it is assumed in this work that the Markov jump processes under consideration are *right continuous*:

$$\lim_{t \to 0^+} \mathbb{P}(y, t \mid x) = \delta(x, y), \quad x, y \in \mathscr{S}. \tag{1.7}$$

As instantaneous states are excluded from consideration, the sequence of stopping times $\{T_i : i \in \mathbb{N}\}$ can be defined as follows:

$$T_0 = t_0; \quad T_i = \begin{cases} \inf\{t \in (T_{i-1}, \infty) : X(t) \neq X(T_{i-1})\}, & i \in \mathbb{N}^+, \quad X(T_{i-1}) \text{ stable}, \\ T_{i-1} = \infty, & X(T_{i-1}) \text{ absorbing}. \end{cases}$$

By construction, this sequence is strictly increasing unless the absorbing state is entered, in which case it becomes $T_i = T_{i+1} = T_{i+2} = \ldots = \infty$. Denote $M = \sup\{i \in \mathbb{N} : T_i < \infty\}$ the number of jumps made by MJP X_t. If $M < \infty$, then the MJP lands at the absorbing state at some time T_M. Time increments $T_{i+1} - T_i$, $i < M$ define holding times. This allows for the initiating of an embedded discrete-time Markov chain.

Definition 1.3.3. Let X_t be the Markov jump process on state space S. Let $\{T_i : i \in \mathbb{N}\}$ be the set of stopping times of X_t introduced above. For $i \in \mathbb{N}$ let $Y_i = X(T_i)$ for $i \leq M$ and $Y_i = X(T_M)$ otherwise. A homogenous discrete-time Markov chain on $\mathscr{S}$ is defined as $Y = \{Y_i : i \in \mathbb{N}\}$ is known as *embedded Markov chain* or *jump chain* of X_t.

For the discrete-time Markov chain Y,

$$Pr(Y_{i+1} = y \mid Y_i = x) = Pr(X(T_{i+1}) = y \mid X(T_i) = x).$$

The one-step transition probability can be conveniently represented with the matrix $\mathbf{Q}$, such as follows:

$$\mathbf{Q}[x,y] = \begin{cases} Pr(Y_1 = y \mid Y_0 = x), & x \text{ is stable,} \\ \mathbf{I} & x \text{ is absorbing,} \end{cases} \qquad x,y \in \mathscr{S}.$$

$\mathbf{I}$ denotes an identity matrix. From construction, $\mathbf{Q}[x,x] = 0$ if the state x is stable, and $\mathbf{Q}[x,x] = 1$ if x is absorbing. Because $\mathbf{Q}$ is a probability matrix, it possesses the following properties:

$$\begin{aligned} &\mathbf{Q}[x,y] \geq 0, \\ &\sum_{y \in \mathscr{S}} \mathbf{Q}[x,y] = 1, \quad \forall x,y \in \mathscr{S}. \end{aligned} \tag{1.8}$$

Proposition 1. *Given the initial state $X(0) = x$, holding time and the next state are independent, and for $x,y \in \mathscr{S}$ and $t \in [0,\infty)$ then holds*

$$Pr(Y_1 = y, \tau_1 > t \mid Y_0 = x) = \mathbf{Q}(x,y)\exp(-\lambda(x)t)$$

Proof. By applying the Bayes Rule, the following can be derived:

$$\begin{aligned} Pr(Y_1 = y, \tau_1 > t \mid Y_0 = x) =& Pr(X(T_1) = y, \tau_1 > t \mid X(0) = x) = \\ =& Pr(X(T_1) = y \mid \tau_1 > t, X(0) = x) \cdot Pr(\tau_1 > t \mid X(0) = x). \end{aligned}$$

For $t < \tau_1$, given $X(0) = x$, it is true that also $X(t) = x$. Utilising the Markov property, the chain starts over at the time t with $X(t) = x$, independent of $\tau_1 > t$ and $X(0) = x$. As such,

$$\begin{aligned} Pr(X(T_1) = y \mid \tau_1 > t, X(0) = x) =& Pr(X(T_1) = y \mid X(t) = x \tau_1 > t, X(0) = x) = \\ =& Pr(X(T_1) = y \mid X(0) = x) = \mathbf{Q}[x,y]. \end{aligned}$$

If x is a stable state, then $\lambda(x) \in (0,\infty)$, and

$$Pr(\tau_1 > t \mid X(0) = x) = \exp(-\lambda(x)t).$$

If x is an absorbing state, then $\lambda(x) = 0$ and $\mathbb{P}(\tau_1 = \infty \mid X(0) = x) = 1$ as well as $\mathbb{P}(Y_1 = x \mid Y_0 = x) = 1$. Therefore,

$$Pr(Y_1 = y, \tau_1 > t \mid Y_0 = x) = \mathbf{Q}[x,y]\exp(-\lambda(x)t) = \mathbf{I}[x,y].$$

□

Proposition 1 can be generalised for set of the stable states $\{x_0, x_1, ..., x_n\}$, $x_i \in \mathscr{S}$, and sequence $(t_1, ...t_n)$ on the interval $[0, \infty)$:

$$Pr(Y_1 = x_1, \tau_1 > t_1, Y_2 = x_2, \tau_2 > t_2, ..., Y_n = x_n, \tau_n > t_n \mid Y_0 = x) =$$
$$\mathbf{Q}[x_0, x_1] \exp(-\lambda(x_0)t_1) \cdot \mathbf{Q}[x_1, x_2] \exp(-\lambda(x_1)t_2) \cdot ... \cdot \mathbf{Q}[x_n, x_{n-1}] \exp(-\lambda(x_n)t_n).$$

Definition 1.3.4. For the sequence of stopping times $\{T_i : i \in \mathbb{N}\}$ of the jump process X_t, the limit

$$T_\infty = \lim_{i \to \infty} T_i$$

exists on $[0, \infty)$ and is called *explosion time*. Even though the holding time spent in a state is positive, it remains feasible that $Pr(T_\infty < \infty) > 0$. In this case, the Markov jump process X_t is called *explosive*. Otherwise, MJP is called *regular* or *non-explosive*.

An explosion, then, indicates that the process becomes arbitrarily large in a finite time. The next two propositions specify conditions under which the Markov jump process is regular.

Proposition 2. *If λ is bounded, then X_t is regular.*

Proposition 3. *A necessary and sufficient condition for a homogeneous Markov jump process to be regular is that*

$$Pr\left(\sum_{x \in \mathscr{S}^+} \frac{1}{\lambda(x)} = \infty \right) = 1,$$

where exponential parameter function $\lambda : \mathscr{S} \mapsto [0, \infty)$ and $\mathscr{S}^+ = \{x \in \mathscr{S} : \lambda(x) > 0\}$.

Proofs for both propositions can be found, for instance, in [3]. As a corollary, if state space $\mathscr{S}$ is finite, then λ is bounded, and a continuous-time MJP on a finite state space is regular.

Definition 1.3.5. Let X_t be a regular MJP on $\mathscr{S}$ with exponential parameter function λ and transition matrix $\mathbf{Q}$, parameter $\mu(x, y) = \lambda(x)\mathbf{Q}[x, y])$ for pair $x, y \in \mathscr{S}$ is called a *transition rate*.

Parameter $\mu(x, y)$ determines the parameters $\lambda(x)$, $x \in \mathscr{S}$, and transition probabilities $\mathbf{Q}[x, y]$ for stable state $x \in \mathscr{S}$ and state $y \in \mathscr{S}$:

$$\lambda(x) = \sum_{y \in \mathscr{S}} \mu(x, y), \qquad x \in \mathscr{S},$$
$$\mathbf{Q}[x, y] = \frac{\mu(x, y)}{\lambda(x)}, \qquad x, y \in \mathscr{S}, \quad x \text{ - stable}.$$

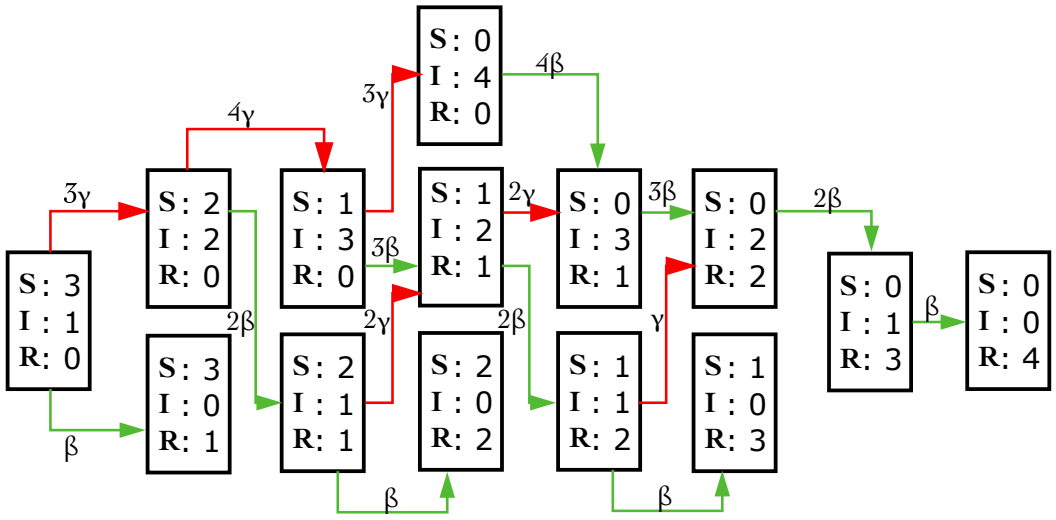


Fig. 1.1. State graph for the Markov jump process of the SIR compartmental model. The model obey the following rules: $\mathrm{S}^{+}\mathrm{I} \xrightarrow{\gamma} 2\mathrm{I}$; $\mathrm{I} \xrightarrow{\beta} \mathrm{R}$. It starts with population size $N = 4$, where 3 individuals are susceptible and 1 infected. Red arrows depict the jumps corresponding to the infection events, and green arrows – the recovery events. Arrows are labelled with the corresponding jump rate. There are 4 absorbing states: $\{S:3,I:0,R:1\}$, $\{S:2,I:0,R:2\}$, $\{S:1,I:0,R:3\}$, $\{S:0,I:0,R:4\}$, where the number of infected is 0, so no further transmission or recovery is possible.

For the absorbing state x holds $\lambda(x) = 0$ and $\mathbf{Q}[x,x] = 1$.

Given X_t being in the state x at the time t and the exponential parameter $\mu(x,y)$, the probability that the process will jump away from x to some other state during time interval dt and land at some state y can be expressed as follows:

$$\mathbb{P}(y,t+dt \mid x,t) = Pr(X(t+dt) = y \mid X(t) = x) = \mu(x,y)dt.$$

Continuous-time MJP on a finite state space $\mathscr{S}$ can be represented with a *state graph* graph: the vertex set of this graph is state space $\mathscr{S}$, and the edges $E = \{(x,y) \in \mathscr{S} \times \mathscr{S} : \mathbf{Q}[x,y] > 0\}$. Each edge $(x,y) \in E$ is labelled with transition rate $\mu(x,y)$. For example, Figure (1.1) depicts the transition between possible states of the SIR model for the population of size 4.

One can consider the Markov jump process as a set of timers, assigned to every pair of states $(x,y) \in \mathscr{S} \times \mathscr{S}$. Each timer is set to some random value Z_{xy} distributed exponentially with parameter $\mu(x,y)$. As soon as the process enters state x, all adjacent timers on (x,y), $y \in \mathscr{S}$ start synchronously. As soon as the first timer fires, the process jumps immediately to the corresponding state y, and then the process repeats. If x is absorbing, then all alarms $Z_{xy} = \infty$, and none of the timers fires ever.

1.4 Analysis of the Markov jump process

1.4.1 Connectivity and recurrence

Definition 1.4.1. Given a regular Markov jump process X_t on a state space $\mathscr{S}$. Then for two states $x, y \in \mathscr{S}$, it can be said that

- x *leads* to y or $x \to y$ if there exists some $t > 0$, such as: $\mathbb{P}(X(t) = y \mid X(0) = x)$,
- x and y *communicate*, or $x \leftrightarrow y$ if x leads to y and y leads to x, and
- the MJP is *irreducible* if all state pairs $(x, y) \in \mathscr{S} \times \mathscr{S}$, $x \neq y$ communicate.

The equivalence relation $\leftrightarrow$ divides state space $\mathscr{S}$ into *communicating classes*. If the Markov jump process is irreducible, all its states are in the same class.

Definition 1.4.2. The stopping time

$$T_H(x) = \inf\{t \geq 0 : X(t) = x \mid X(0) \neq x\}$$

is called a *first hitting time* for the state $x \in \mathscr{S}$.
The stopping time

$$T_E(x) = \inf\{t \geq 0 : X(t) \neq x \mid X(0) = x\}$$

is called a *first escape time* from the state $x \in \mathscr{S}$.
The stopping time

$$T_R(x) = \inf\{t > T_E(x) : X(t) = x\}$$

is called a *first return time* to the state $x \in \mathscr{S}$.

Definition 1.4.3. State $x \in \mathscr{S}$ is called *recurrent* if $Pr(T_R(x) < \infty) = 1$ or if x is absorbing. Otherwise, it is considered *transient*.

In a communicating class either all states are transient or all states are recurrent. Therefore, the MJP can be referred to as the "transient MJP" or the "recurrent MJP" respectively. If the MJP is irreducible and recurrent, then it is also regular.

1.4.2 Transition semigroup and infinitesimal generator

Definition 1.4.4. For the regular Markov jump process X_t on a state space $\mathscr{S}$, matrix $\mathbf{P_t}$, such that

$$\mathbf{P_t}[x, y] = Pr(X(t) = y \mid X(0) = x) = \mathbb{P}(y, t \mid x), \qquad x, y \in \mathscr{S}, \quad t \in [0, \infty)$$

is called *transition probability matrix*. By definition, $\mathbf{P_0} = \mathbf{I}$.

The probability density function of X_t with initial condition $X(0) = x_0$ is given by the mapping $y \mapsto \mathbf{P_t}[x,y]$. Consequently, $\mathbf{P_t}$ can be identified as a probability matrix satisfying the following requirements:

$$\begin{aligned} &\mathbf{P_t}[x,y] \geq 0, \\ &\sum_{y \in \mathscr{S}} \mathbf{P_t}[x,y] = 1, \quad \forall x,y \in \mathscr{S}. \end{aligned}$$

The transition matrix also satisfies the Chapman–Kolmogorov equation:

$$\mathbf{P_{t+t'}} = \mathbf{P_t}\mathbf{P_{t'}},$$

which can be expressed in the following element-wise form:

$$\mathbf{P_{t+t'}}[x,z] = \sum_{y \in \mathscr{S}} \mathbf{P_t}[x,y]\mathbf{P_{t'}}[y,z].$$

Definition 1.4.5. The collection of the transition probability matrices $\mathbf{P} = \{\mathbf{P_t}, t \in [0,\infty)\}$ is called a *semigroup*. Because the MJP X_t is right continuous (1.7) and contains no instantaneous states, then the following holds:

$$\lim_{t \to 0^+} \mathbf{P_t}[x,x] = 1, \quad \text{for each } x \in \mathscr{S}.$$

The semigroup satisfying this property is also called *standard*.
The semigroup is called *uniform* or *uniformly continuous* if this property holds uniformly for all $x \in \mathscr{S}$.

The standard semigroup property states that the probability of a Markov process staying in state x at time t approaches 1 as t approaches 0 from the right. This property ensures that the Markov process behaves smoothly and avoids abrupt transitions.

The uniform semigroup property, on the other hand, requires that the convergence of the transition probability matrix to be uniform across all states in the Markov process. Hence, not only does the probability of staying in state x approach 1 as t approaches 0, but it does so uniformly across all states in the process. The uniform semigroup property is a stronger condition than is the standard semigroup property and provides more information about the continuity of the Markov process.

In practical terms, the uniform semigroup property is a desirable criterion for the validity of numerical methods for approximating Markov processes, as it ensures that the method

accurately captures the behaviour of the process for all states. The standard semigroup property is a weaker condition but remains important to ensure the smooth behaviour of the Markov process.

Proposition 4. *If parameter λ of the MJP X is bounded, it implies that the corresponding semigroup is uniform.*

Proof. Assume that λ is bounded. It implies that the supremum norm – such as: $\|\lambda\| = \sup\{|\lambda(x)|, x \in \mathscr{S}\}$ –exists. Consider now $\mathbf{P_t}[x,x]$:

$$\begin{aligned}\mathbf{P_t}[x,x] =& \mathbb{P}(x,t \mid x) = Pr(X(t) = x \mid X(0) = x) \geq \\ \geq& Pr(\tau > t \mid X(0) = x) = \exp(-\lambda(x)t) \geq \exp(-\|\lambda\|t).\end{aligned}$$

The term $\exp(-\|\lambda\|t) \to 1$ as $t \to 0$ uniformly for all x. □

Due to the requirement of no transitions in zero time (1.6) and right-continuity assumption (1.7) in this work, it can be asserted that the transition semigroup $\mathbf{P}$ is uniform, and $\lim_{t\to 0^+} \mathbf{P_t} = \mathbf{I}$. The term "semigroup" will further refer to a uniform semigroup unless otherwise specified.

Connection between regular transition semigroup $\mathbf{P}$, one-step transition matrix $\mathbf{Q}$ and jump parameter λ can be described with the integral equation,

$$\mathbf{P_t}[x,y] = \mathbf{I}[x,y]\exp(-\lambda(x)t) + \int_0^t \lambda(x)\exp(-\lambda(x)s)\mathbf{Q}\mathbf{P_{t-s}}[x,y]\,ds. \tag{1.9}$$

Definition 1.4.6. Consider semigroup $\mathbf{P} = \{\mathbf{P_t} : t \in [0,\infty)\}$ of the Markov jump process X_t.Then,

$$\mathbf{G} = \lim_{t\to 0^+} \frac{\mathbf{P_t} - \mathbf{I}}{t}$$

is an *infinitesimal generator* of X_t, and $\mathbf{G}[x,y] = -\lambda(x)\mathbf{I}[x,y] + \lambda(x)\mathbf{Q}[x,y]$ for all $x, y \in \mathscr{S}$.

Generator matrix $\mathbf{G}$ possesses the following properties:

$$\begin{aligned}-\infty <& \mathbf{G}[x,x] \leq 0, \\ \sum_{y\in\mathscr{S}} & \mathbf{G}[x,y] = 0, \qquad x \in \mathscr{S}.\end{aligned}$$

Jump rate λ and transition matrix $\mathbf{Q}$ are also defined by $\mathbf{G}$, such as:

$$\begin{aligned}&\lambda(x) = -\mathbf{G}[x,x], \quad \forall x \in \mathscr{S}, \\ &\mathbf{Q}[x,y] = -\frac{\mathbf{G}[x,y]}{\mathbf{G}[x,x]}, \quad x \in \mathscr{S} \text{ is stable}, y \in \mathscr{S} \setminus \{x\}.\end{aligned}$$

A comparison of definition 1.4.6 with 1.3.5, reveals

$$\begin{aligned}\mathbf{G}[x,y] &= \mu(x,y), \quad x \neq y, \quad x,y \in \mathscr{S} \\ \mathbf{G}[x,x] &= -\sum_{y\in\mathscr{S},y\neq x} \mathbf{G}[x,y] = -\sum_{y\in\mathscr{S},y\neq x} \mu(x,y).\end{aligned} \tag{1.10}$$

Also, now the probability of the process X_t being in a state x and jumping to the state y during time interval dt can be expressed using the generator matrix $\mathbf{G}$:

$$\mathbb{P}(y,t+dt \mid x,t) = \mathbf{G}[x,y]dt.$$

1.4.3 Backward, forward, and master equations

Consider MJP X_t with transition semigroup $\mathbf{P}$ and generator matrix $\mathbf{G}$. Then, the matrix function $t \mapsto \mathbf{P_t}$ is differentiable on $t \in [0,\infty)$ and satisfies the *Kolmogorov backward equation*:

$$\frac{\mathrm{d}\mathbf{P_t}}{\mathrm{d}t} = \mathbf{G}\mathbf{P_t}. \tag{1.11}$$

The *uniform* semigroup $\mathbf{P}$ also satisfies the *Kolmogorov forward equation*:

$$\frac{\mathrm{d}\mathbf{P_t}}{\mathrm{d}t} = \mathbf{P_t}\mathbf{G} \tag{1.12}$$

A comparison (1.11) and (1.12) indicates that for the uniform semigroup $\mathbf{P}$ holds $\mathbf{P_t}\mathbf{G} = \mathbf{G}\mathbf{P_t}$.

For the finite state space $\mathscr{S}$, the solution of the equation(1.12) can be given in the form of the matrix exponential:

$$\mathbf{P_t} = \exp(t\mathbf{G}) = \sum_{n=0}^{\infty} \frac{t^n}{n!}\mathbf{G}^n, t \in [0,\infty).$$

Notably, in the case of an infinite state space, the transition probabilities may not be well-defined for all pairs of states. Therefore, it may not be possible to define the infinitesimal generator straightforwardly for an infinite state space Markov process. Hence, in some cases, it may become necessary to solve coupled ordinary differential equations (ODE) systems of the component-wise forms of the Kolmogorov backward and forward equations:
Kolmogorov backward equation:

$$\frac{\mathrm{d}\mathbf{P_t}[x,y]}{\mathrm{d}t} = -\lambda(x)\mathbf{P_t}[x,y] + \sum_{\forall z\in\mathscr{S},z\neq x} \lambda(x)\mathbf{Q}[x,z]\mathbf{P_t}[z,y], \quad (x,y) \in \mathscr{S}\times\mathscr{S}, \tag{1.13}$$

Kolmogorov forward equation:

$$\frac{\mathrm{d}\mathbf{P_t}[x,y]}{\mathrm{d}t} = -\lambda(y)\mathbf{P_t}[x,y] + \sum_{\forall z\in\mathscr{S}\, z\neq y} \mathbf{P_t}[x,z]\lambda(z)\mathbf{Q}[z,y], \quad (x,y)\in\mathscr{S}\times\mathscr{S}. \tag{1.14}$$

As compared to the forward equation, the backward equation has wider scope, requiring only the transition semigroup to be standard, whereas the forward equation requires a uniform semigroup. Sometimes, however, the forward equation is easier to solve, so the assumption is often made that λ is bound and, therefore, $\mathbf{P}$ is uniform.

If the initial distribution vector $\mathbf{p_0}$ is known (see definition 1.2.4), then by denoting

$$\mathbf{p_t} = \mathbf{p_0}\mathbf{P_t}, \tag{1.15}$$

where $\mathbf{p_t}$ is a vector describing the probability of the X_t being at state x at the time t, given the initial distribution $\mathbf{p_0}$:

$$\mathbf{p_t}[y] = Pr(X(t) = y \mid x)Pr(X(0) = x) = \mathbb{P}(y,t \mid x)\mathbf{p_0}[x], \quad \forall x,y\in\mathscr{S}.$$

By applying left-side multiplication to the forward equation (1.12), the ***master equation*** is derived:

$$\frac{\mathrm{d}\mathbf{p}(t)}{\mathrm{d}t} = \mathbf{p}(t)\mathbf{G}. \tag{1.16}$$

The general solution of the master equation (1.16) for the finite state space $\mathscr{S}$ is based on the matrix exponential:

$$\mathbf{p}(t) = \mathbf{p_0}\exp(t\mathbf{G}). \tag{1.17}$$

The component-wise form of the master equation is as follows:

$$\frac{\mathrm{d}\mathbf{p_t}[x]}{\mathrm{d}t} = \sum_{\forall y\in\mathscr{S}} \mathbf{p_t}[y]\mathbf{G}[y,x] = -\lambda(x)\mathbf{p_t}[x] + \sum_{\substack{\forall y\in\mathscr{S}-\exp(\\ y\neq x}} \mathbf{p_t}[y]\lambda(y)\mathbf{Q}[y,x]. \tag{1.18}$$

The above equation could also be rewritten as,

$$\begin{aligned}\frac{\mathrm{d}\mathbf{p_t}[x]}{\mathrm{d}t} = \sum_{\forall y\in\mathscr{S}} \mathbf{p_t}[y]\mathbf{G}[y,x] = \sum_{y=x} \mathbf{p_t}[y]\mathbf{G}[y,x] + \sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} \mathbf{p_t}[y]\mathbf{G}[y,x] = \\ = \mathbf{p_t}[x]\mathbf{G}[x,x] + \sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} \mathbf{p_t}[y]\mathbf{G}[y,x].\end{aligned} \tag{1.19}$$

According to (1.10) $\mathbf{G}[x,x] = -\sum_{y\neq x}^{\forall y\in\mathscr{S}} \mathbf{G}[x,y]$, therefore, equation(1.19) becomes,

$$\begin{aligned}\frac{\mathrm{d}\mathbf{p_t}[x]}{\mathrm{d}t} &= -\sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} \mathbf{p_t}[x]\mathbf{G}[x,y] + \sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} \mathbf{p_t}[y]\mathbf{G}[y,x] = \\ &= \sum_{\forall y\in\mathscr{S}} \Big(\mathbf{p_t}[y]\mathbf{G}[y,x] - \mathbf{p_t}[x]\mathbf{G}[x,y]\Big).\end{aligned} \tag{1.20}$$

The first term on the right side of equation (1.20) represents the rate at which the system transitions from any other state y to state x, weighted by the probability of being in state y at time t. The second term represents the rate at which the system transitions from state x to state y, weighted by the probability of being in state x at time t.

Notably, if the *initial state* $X(0) = x_0$ is known (i.e., $\mathbf{p_0}[x_0] = 1$), the master equation (1.20) describes the evolution of the state density function $\mathbb{P}(x,t \mid x_0)$:

$$\frac{\mathrm{d}\mathbb{P}(x,t \mid x_0)}{\mathrm{d}t} = \sum_{\forall y\in\mathscr{S}} \Big(\mathbb{P}(y,t \mid x_0)\mathbf{G}[y,x] - \mathbb{P}(x,t \mid x_0)\mathbf{G}[x,y]\Big). \tag{1.21}$$

In summary, the forward equation is used to predict the future state of a system based on its current state and describe how the probability distribution of the state evolves over time in a forward direction. The backward equation is used to compute transition probabilities backwards in time. It is typically applied in the context of calculating conditional probabilities and expectations when considering the reverse time direction. The master equation is often used to model the time evolution of the state of a system in a discrete state space, understand the long-term behaviour and compute statistical properties of the system, depending on its specific form and context.

The master equation is a powerful tool for describing the time evolution of a system over time, but it suffers from the *curse of dimensionality*. As the number of states or variables increases, the number of terms in the equation grows exponentially. It thus becomes increasingly difficult to solve the equation analytically or numerically.

1.4.4 Long-term behaviour of Markov jump processes

Definition 1.4.7. Function $f : \mathscr{S} \mapsto [0,\infty)$ satisfying:

$$f = f\mathbf{P_t}, \quad \forall t \in [0,\infty)$$

for every $t \in [0,\infty)$ is called *invariant* for the MJP X_t. If also $f(\mathscr{S}) = 1$, then it is a *stationary distribution* of the MJP X_t.

An invariant function is one that remains constant over time, while a stationary distribution is a probability distribution that remains constant over time.

If a function is invariant under the transition probabilities of the MJP, then the expected value of that function remains constant over time, regardless of the initial state of the system. Ergo, the system will eventually reach a steady-state distribution where the probability of being in any particular state is constant over time. Furthermore, an invariant function can be used to determine the limiting behaviour of the MJP as time approaches infinity.

A stationary distribution of a Markov jump process is a probability distribution that remains unchanged over time, even as the system transitions between states. If the stationary distribution f exists, then the Markov state density function $\mathbb{P}$ eventually becomes time independent. However, the MJP itself does not thereby become static, but rather stochastic properties of the MJP eventually become static. In other words, if the Markov jump process starts in any initial state, after many transitions the probability distribution over the states converges to the stationary distribution.

1.4.5 Markov propagator

The Markov process can also be described using the Markov propagator.

Definition 1.4.8. Given Markov process X_t being in the state x at the time t, the state displacement of the X_t during time interval dt

$$\Xi(dt;x,t) = X(t+dt) - X(t)$$

is called the *Markov propagator* of the MJP X_t.

The state displacement $\Xi(dt;x,t)$ is a random variable and is specified by its density function:

$$\Pi(\nu \mid dt;x,t) = Pr(x+\nu, t+dt \mid x,t).$$

However, as the focus of this work is time-homogeneous MJPs, so using the definition of the homogeneity (1.2.6) and probability density function of the time-homogeneous process (1.4), the above equation can be rewritten:

$$\Pi(\nu \mid dt;x) = Pr(x+\nu, dt \mid x). \tag{1.22}$$

Given $X(t) = x$, by the time $t+dt$, the process either will have jumped once with the probability $\lambda(x)dt$ or the jump did not occur with the probability $1-\lambda(x)dt$. If the jump does occur, then the probability that it landed in the state $x+\nu$ is $\mathbf{Q}[x, x+\nu]$. If the jump did

not occur, then the probability of it landing at the state $x+\nu$ is zero if $\nu \neq 0$ and 1 otherwise. Hence, the probability density function $\Pi(\nu \mid dt; x)$ takes the following form:

$$\Pi(\nu \mid dt; x) = \lambda(x)\mathbf{Q}[x, x+\nu]dt + [1-\lambda(x)dt]\,\delta(\nu,0) = \\ = \mathbf{G}[x, x+\nu]dt + \left[1 - \sum_{\substack{\forall \nu\prime: \\ (x+\nu\prime)\in\mathscr{S}}} \mathbf{G}[x, x+\nu\prime]dt\right]\delta(\nu,0),$$

where $\delta(\nu,0)$ denotes Kronecker-Delta function.
The state density function can be expressed using the propagator density function (1.22):

$$\mathbb{P}(x,t \mid x_0) = \sum_{\substack{\forall \nu_1: \\ (x_1+\nu_1)\in\mathscr{S}}} \ldots \sum_{\substack{\forall \nu_{k-1}: \\ (x_{k-1}+\nu_{k-1})\in\mathscr{S}}} \prod_{i=1}^{k} \Pi(\nu_i \mid dt; x_{i-1}).$$

1.4.6 Moments evolution

Often master equations (1.20) or (1.21) of the MJP X_t on a discrete state space S could not be solved directly for p_t or $\mathbb{P}(y,t \mid x)$, respectively. It therefore becomes useful to know how various moments of the process X_t evolve over time. While generating trajectories and computing sample averages can be a viable approach, density function equations are essential to analytically calculate a system's moments.

Definition 1.4.9. The *initially conditioned average* of any univariate function $g(X(t)$,

$$\langle g(X(t) \mid X(0) = x_0)\rangle = \langle g(X(t))\rangle = \sum_{\forall x\in\mathscr{S}} g(x)\mathbb{P}(x,t \mid x_0).$$

Definition 1.4.10. The average

$$\langle X^k(t)\rangle \equiv \sum_{\forall x\in S} x^k \mathbb{P}(x,t \mid x_0), t \in$$

is called the k^{th} *raw moment* of X_t.

The raw moments of the process describe the expected values of the number of times the process is in each state at a given time. With respect to the normalisation condition (1.5), the zeroth moment of X always exists and equals one:

$$\langle X^0(t)\rangle = 1.$$

Deriving moment evolution equations using the Markov propagator

The moment evolution equation can be derived using the definition of the propagator (1.4.8):

$$X(t+dt) = X(t) + \Xi(dt; X(t), t).$$

Rising in power $n \geqslant 1$ and applying binomial formula:

$$\begin{aligned} X^n(t+dt) &= [X(t) + \Xi(dt; X(t), t)]^n = \\ &= \sum_{k=0}^{n} \binom{n}{k} X^{n-k}(t) \Xi^k(dt; X(t), t) = \\ &= X^n(t) + \sum_{k=1}^{n} \binom{n}{k} X^{n-k}(t) \Xi^k(dt; X(t), t). \end{aligned}$$

By applying the average operator $\langle\rangle$ to both sides of the above equation and using its linearity:

$$\langle X^n(t+dt)\rangle = \langle X^n(t)\rangle + \sum_{k=1}^{n} \binom{n}{k} \left\langle X^{n-k}(t) \Xi^k(dt; X(t), t) \right\rangle. \tag{1.23}$$

Utilising the definition of the average (1.4.9) for the $\left\langle X^j(t) \Xi^k(dt; X(t), t) \right\rangle$,

$$\begin{aligned} &\left\langle X^j(t) \Xi^k(dt; X(t), t) \right\rangle = \left\langle X^j(t) B_k(X(t), t) \right\rangle dt + o(dt), \\ &\text{where } B_k(X(t), t) = \sum_{\forall x \in \mathscr{S}} \sum_{\substack{\forall \nu: \\ (x+\nu) \in \mathscr{S}}} \nu^k \mathbf{G}[x, x+\nu]. \end{aligned} \tag{1.24}$$

The term $o(dt)$ denotes a term that goes to zero faster than dt ($o(dt)/dt \to 0$ as $dt \to 0$). The detailed derivation of this property can be found in [4]. Substituting equation (1.24) to the (1.23) yields the following:

$$\langle X^n(t+dt)\rangle = \langle X^n(t)\rangle + \sum_{k=1}^{n} \binom{n}{k} \left\langle X^{n-k}(t) B_k(X(t), t) \right\rangle dt + o(dt).$$

Dividing through dt and taking the limit $dt \to 0$, the set of the equations is derived:

$$\frac{\mathrm{d}\left\langle X^k(t)\right\rangle}{\mathrm{d}t} = \sum_{k=1}^{n} \binom{n}{k} \left\langle X^{n-k}(t) B_k(X(t), t) \right\rangle,$$

where $\left\langle X^{n-k}(t) B_k(X(t), t) \right\rangle$ is initially conditioned average.

Deriving moment evolution equations using the master equation

To derive the time evolution equations for the raw moments of X, the time derivative is taken:

$$\frac{\mathrm{d}\langle X^k(t)\rangle}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t}\sum_{x\in\mathscr{S}} x^k \mathbb{P}(x,t \mid x_0) = \sum_{x\in\mathscr{S}} x^k \frac{\partial \mathbb{P}(x,t \mid x_0)}{\partial t}.$$

Using component-wise form of the master equation (1.21) :

$$\begin{aligned}\frac{\mathrm{d}\langle X^k(t)\rangle}{\mathrm{d}t} &= \sum_{x\in\mathscr{S}} x^k \sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} \Big[\mathbb{P}(y,t \mid x_0)\mathbf{G}[y,x] - \mathbb{P}(x,t \mid x_0)\mathbf{G}[x,y]\Big] = \\ &= \sum_{x\in\mathscr{S}}\sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} x^k \mathbb{P}(y,t \mid x_0)\mathbf{G}[y,x] - \sum_{x\in\mathscr{S}}\sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} x^k \mathbb{P}(x,t \mid x_0)\mathbf{G}[x,y].\end{aligned}$$

By performing expansion of the summation operators and regrouping the elements, the above equation can be rewritten in the form below:

$$\frac{\mathrm{d}\langle X^k(t)\rangle}{\mathrm{d}t} = \sum_{x\in\mathscr{S}}\sum_{\substack{\forall y\in\mathscr{S}\\ y\neq x}} (x^k - y^k)\mathbb{P}(y,t \mid x_0)\mathbf{G}[y,x] \tag{1.25}$$

The next step of the derivation is to note, that for each state $y \in \mathscr{S}$, every other state $x \in \mathscr{S}$ with $x \neq y$, can be represented as $x = y + \nu$, where ν is a size of the jump from y to x. Now, equation (1.25) can be rewritten as:

$$\frac{\mathrm{d}\langle X^k(t)\rangle}{\mathrm{d}t} = \sum_{\forall y\in\mathscr{S}}\sum_{\substack{\forall \nu:\\ (x+\nu)\in\mathscr{S}}} ((y+\nu)^k - y^k)\mathbb{P}(y,t \mid x_0)\mathbf{G}[y,y+\nu].$$

Using the binomial formula to expand $(y+\nu)^k$, the following form of the equation could be derived:

$$\begin{aligned}\frac{\mathrm{d}\langle X^k(t)\rangle}{\mathrm{d}t} &= \sum_{\forall y\in\mathscr{S}}\sum_{\substack{\forall \nu:\\ (x+\nu)\in\mathscr{S}}} \left[\sum_{j=0}^{k}\binom{k}{j} y^{k-j}\nu^j - y^k\right]\mathbb{P}(y,t \mid x_0)\mathbf{G}[y,y+\nu] = \\ &= \sum_{\forall y\in\mathscr{S}}\sum_{\substack{\forall \nu:\\ (x+\nu)\in\mathscr{S}}} \left[\sum_{j=1}^{k}\binom{k}{j} y^{k-j}\nu^j\right]\mathbb{P}(y,t \mid x_0)\mathbf{G}[y,y+\nu] = \\ &= \sum_{j=1}^{k}\binom{k}{j}\sum_{\forall y\in\mathscr{S}} y^{k-j}\mathbb{P}(y,t \mid x_0)\sum_{\substack{\forall \nu:\\ (x+\nu)\in\mathscr{S}}} \nu^j \mathbf{G}[y,y+\nu].\end{aligned}$$

Applying the definition of the *initially conditioned average* 1.4.9, the set of the time evolution equations of the k$^{\text{th}}$ raw moment of the MJP X_t can be obtained:

$$\frac{\mathrm{d}\left\langle X^k(t)\right\rangle}{\mathrm{d}t} = \sum_{j=1}^{k} \binom{k}{j} \left\langle X^{k-j}(t) B_k(X(t)), t \right\rangle, \tag{1.26}$$

where $B_k(X(t))$is given by formula (1.24).

These equations for the k$^{\text{th}}$ raw moment are to be solved with initial conditions:

$$\left\langle X^k(t_0)\right\rangle = x_0^k$$

Important to note, that set of equations (1.26) depends on the first $k-1$ moments of X_t, being multiplied by the polynomial function $B_k(X(t))$. If this function is a polynomial in y of degree $\leq k$, then the hierarchy of the moment evolution equations will be closed and could have a closed solution. If, however, this condition is not satisfied, then in some cases, an approximate solution could be obtained by applying, for instance, the approximation procedure described in [4]. The most used raw moment is the first one, which is also an expectation of the process X_t and denoted as $\mathbb{E}[X]$. The time evolution of the expectation of the X_t is described with the following equation:

$$\frac{\mathrm{d}\mathbb{E}[X](t)}{\mathrm{d}t} = \frac{\mathrm{d}\left\langle X(t)\right\rangle}{\mathrm{d}t} = \sum_{\forall x \in \mathscr{S}} \sum_{\substack{\forall \nu: \\ (x+\nu)\in\mathscr{S}}} \nu G[x, x+\nu] = \sum_{\forall x\in\mathscr{S}} \sum_{\forall y \neq x} (x-y) G[y,x], \tag{1.27}$$

with initial condition being:

$$\mathbb{E}[X](t_0) = \langle X(t_0)\rangle = x_0$$

Definition 1.4.11. The average

$$\left\langle \bar{X}^k(t)\right\rangle \equiv \sum_{\forall x\in\mathscr{S}} (x - \langle X(t)\rangle)^k \mathbb{P}(x,t), t \in \tag{1.28}$$

is called the *kth central moment* of X_t. $\langle X\rangle$ denotes the expectation or the first raw moment of the X_t.

Central moments could be expressed in terms of the raw moments. By applying the binomial formula to the definition of the central moment (1.4.11) :

$$\begin{aligned}\left\langle \bar{X}^k(t)\right\rangle &= \sum_{\forall x\in\mathscr{S}}\left[\sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}x^{k-j}\left\langle X(t)\right\rangle^k\right]\mathbb{P}(x,t) = \\ &= \sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\left\langle X(t)\right\rangle^k\sum_{\forall x\in\mathscr{S}}x^{k-j}\mathbb{P}(x,t) = \\ &= \sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\left\langle X(t)\right\rangle^k\left\langle X^{k-j}(t)\right\rangle = \\ &= \left\langle X(t)\right\rangle^k\sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\left\langle X^{k-j}(t)\right\rangle.\end{aligned}$$

Time evolution equations respectively take a form:

$$\begin{aligned}\frac{\mathrm{d}\left\langle \bar{X}^k(t)\right\rangle}{\mathrm{d}t} =& \frac{\mathrm{d}}{\mathrm{d}t}\left[\langle X(t)\rangle^k\sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\left\langle X^{k-j}(t)\right\rangle\right] = \\ =& \left[\frac{\mathrm{d}}{\mathrm{d}t}\langle X(t)\rangle^k\right]\sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\left\langle X^{k-j}(t)\right\rangle + \\ &+\langle X(t)\rangle^k\left[\frac{\mathrm{d}}{\mathrm{d}t}\sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\left\langle X^{k-j}(t)\right\rangle\right] = \\ =& k\langle X(t)\rangle^{k-1}\left[\frac{\mathrm{d}}{\mathrm{d}t}\langle X(t)\rangle\right]\sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\left\langle X^{k-j}(t)\right\rangle + \\ &+\langle X(t)\rangle^k\sum_{j=0}^{k}\binom{k}{j}(-1)^{k-j}\frac{\mathrm{d}}{\mathrm{d}t}\left\langle X^{k-j}(t)\right\rangle.\end{aligned}$$

This set of equations, again, depends on a $k-1$ first raw moments of X_t, and is therefore subject to similar restrictions applied to the evolution equations of the raw moments. The most used central moment, *variance*, is easy to derive from their definition.
The variance of X_t is the second central moment:

$$\mathbb{V}[X](t) = \sum_{\forall x\in\mathscr{S}}\Big(x-\langle X(t)\rangle\Big)^2\mathbb{P}(x,t) = \left\langle X^2(t)\right\rangle - \langle X(t)\rangle^2. \tag{1.29}$$

Taking time derivative,

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{V}[X](t) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\left\langle X^2(t)\right\rangle - \langle X(t)\rangle^2\right] = \frac{\mathrm{d}}{\mathrm{d}t}\left\langle X^2(t)\right\rangle - 2\langle X(t)\rangle\frac{\mathrm{d}}{\mathrm{d}t}\langle X(t)\rangle. \tag{1.30}$$

Substituting equations (1.26) and (1.27),

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{V}[X](t) = 2\left[\Big\langle X(t)B_1(X(t))\Big\rangle - \Big\langle X(t)\Big\rangle\Big\langle B_1(X(t))\Big\rangle\right] + \Big\langle B_2(X(t))\Big\rangle, \tag{1.31}$$

where $B_k(X(t))$ is given by formula (1.24).
The initial condition for the variance is,

$$\mathbb{V}[X](0) = 0. \tag{1.32}$$

1.4.7 Application to the spreading process

Compartmental modelling

Although the approach described in this section was originally developed for the chemical species to describe the time evolution of the chemical species concentration [5, 6], the formalism of the reaction-diffusion process can be applied to model the spread of infectious diseases in populations [7]. The basic principles of reaction kinetics can be adapted to epidemiological modelling by interpreting the individuals in the population as chemical species and the disease transmission events as chemical reactions.

The population is divided into different compartments based on epidemiological status: for example, susceptible (S), infected (I) and recovered (R), in the SIR models [8]. For the compartmental approach to be feasible, the assumption should be made that the population is well-stirred, meaning that all individuals are equally likely to interact with each other.

Individuals are assumed to interact through "reaction channels" R_k with the form of

$$\mathrm{Z_A} + \mathrm{Z_B} \longrightarrow \mathrm{Z_C} + \ldots$$

This form specifies the transition of two individuals from compartments A and B into compartments C and beyond, with those in compartments A and B referred to as *reactants*, and those in compartments C considered as *product* species. The goal is to apply the tools of Markov process theory to analyse these reactions R_k.

For the particular combination of reactant individuals, there exists some constant c_k – probability per unit of time that they will react, called the *specific probability rate constant*. Because the system is well-stirred and the selection of reactant individuals is arbitrary, this probability per unit of time is the same for any other combination of reactant individuals. Then $c_k dt$ is a probability that the chosen combination of individuals will react according to R_k in the next time interval $[t, t+dt)$. This is a probability per randomly chosen reactants of R_k.

For spontaneous (0^{th} order) and unimolecular (1^{st}) order reactions, the constant c_k is always independent of the population size N. For bimolecular reactions, however, c_k will be proportional to population size. This dependency follows from the fact that in a well-stirred population, the probability of the individual contacting any other is $1/(N-1)$.

It is assumed that c_k are all known (are constants), the propensity rates are time-independent, and the fluctuations in the system arise from intrinsic noise. If c_k is a time-dependent function, it corresponds to time-dependent propensity rates and the fluctuations in the system arise from variability in external factors (extrinsic noise) [9].

The total number of potential combinations of reactant individuals is $N_A N_B$, where $N.$ refers to the number of individuals in the corresponding compartment. Assuming a brief time interval dt, during which the likelihood of multiple reactions is insignificant, the probability that one of these combinations will react is $cN_A N_B dt$. The term $a_k = cN_A N_B$ represents the *reaction propensity*, indicating the probability of a reaction taking place per unit of time. As can be seen, the reaction propensities a_k for the reactions order ≥ 1 depend on the current state of the system $X(t)$ – namely on the current number of the reactant individuals. Reaction propensities of the typical reactions up to the 3^{rd} order are presented in Table 1.1. The main interest of the model is for the value $X(t) = X_1(t), X_2(t), \ldots$ – the number of individuals in each compartment of the population at the time t.

Importantly, however, *reaction propensity* differs from the deterministic *reaction rate*. The general form of the reaction rates of elementary reactions is given by the law of mass action, which states that the rate of an elementary reaction R_k is proportional to the product of the concentration of all involved reactants raised to the power of their stoichiometric coefficients with the factor of proportionality known as reaction rate constant.

Reaction rate defines a number of reactive events per unit of time, usually expressed as an equivalent concentration change per unit of time. Propensity, on the other hand, expresses the probability of a reactive event per unit of time. Generally speaking, deterministic formalism is an approximation of stochastic formalism, generally accurate only if the system is sufficiently large. Therefore, despite a very close connection between propensity functions and conventional deterministic reaction rates, the latter, being an approximate special case of the former, cannot be used to derive the former [10].

Stochastic process X_t, governing described reaction kinetics, is a well-defined temporarily homogeneous regular MJP on non-negative integers $\mathbb{N}^N$. Its value changes only when one of the reactions R_k occurs. Moreover, its value changes to a particular size, defined by the reaction. An example of the reactions and stoichiometric matrix for a compartmental model is presented in Fig.1.2.

Order	Reaction	propensity
0	$\varnothing \xrightarrow{c} \dots$	c
1	$Z_A \xrightarrow{c} \dots$	cN_A
2	$Z_A + Z_B \xrightarrow{c} \dots$	cN_AN_B
2	$Z_A + Z_A \xrightarrow{c} \dots$	$cN_A(N_A-1)/2$
3	$Z_A + Z_B + Z_C \xrightarrow{c} \dots$	$cN_AN_BN_C$
3	$Z_A + Z_A + Z_B \xrightarrow{c} \dots$	$cN_A(N_A-1)N_B$
3	$Z_A + Z_A + Z_A \xrightarrow{c} \dots$	$cN_A(N_A-1)(N_A-2)/6$

Table 1.1 Reaction propensities for the typical reactions. Adapted from [5].

$Z_{(\cdot)}$ represents reactant individuals, $N_{(\cdot)}$ represents the number of the individuals in the respective compartment.

The alterations in the number of individuals in each compartment are defined by the *stoichiometric matrix*, denoted by $\mathbb{S}$. This matrix encloses information regarding all the changes caused by the reactions R_k. It comprises the stoichiometric coefficients from each reaction that depict the spreading process. The stoichiometric coefficients are integer numbers. The organisation of these coefficients forms the matrix, such that the columns correspond to chemical reactions, while the rows identify the compounds of the population.

For the finite state space, the propensity functions define the corresponding infinitesimal generator matrix:

$$\begin{aligned} \mathbf{G}[x, x+\nu_k] &= \mathbf{G}[x, x+\nu_k] = a_k(x), \\ \mathbf{G}[x,x] &= - \sum_{\substack{\forall \nu_k \neq 0, \\ (x+\nu_k) \in \mathscr{S}}} \mathbf{G}[x, x+\nu_k], \end{aligned} \tag{1.33}$$

where $\nu_k = \mathbb{S}[\cdot, k]$, the change vector corresponding to the reaction R_k.
Now, the master equation (1.20) takes the following form:

$$\frac{\mathrm{d}\mathbf{p_t}[x]}{\mathrm{d}t} = \sum_{k=1}^{M} \mathbf{p_t}[x-\nu_k] a_k(x-\nu_k) - \mathbf{p_t}[x] a_k(x). \tag{1.34}$$

Equations (1.34) are also called *chemical master equations* or *CME*. As discussed in section 1.4.3, solving CME often seems impossible due to limitations as a curse of dimensionality.

Among the moment and moment evolution equations, the most interesting are those for mean (first raw moment) and variance (second central moment).

The time evolution equation for the expectation is,

$$\frac{\mathrm{d}\langle X(t)\rangle}{\mathrm{d}t} = \sum_{k=1}^{M} \nu_k \langle a_k(X(t))\rangle \,. \tag{1.35}$$

If any of the reaction channels R_k involves two and more reactants, then the equation (1.35) will have at least one quadratic moment in the form of $\langle X_i(t)X_j(t)\rangle$ on its right-hand side, such that none of the moment evolution equations will be closed [11].

If no fluctuations are assumed, however, meaning $X(t)$ is a deterministic process, then $\mathbb{V}(X(t)) = 0$, $\langle X(t)\rangle = X(t)$ and equation (1.35) simplify to the following:

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = \sum_{k=1}^{M} \nu_k a_k(X(t)). \tag{1.36}$$

A set of coupled ODEs (1.36) is called the *reaction rate equations*. It can be also expressed in the matrix form:

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = \mathbb{S} \cdot a(X(t)) \tag{1.37}$$

where $\mathbb{S}$ is a stoichiometric matrix and $a = \{a_k(X(t))\}^T$ is a reaction propensity vector.
The reaction rate equations would be applicable in the absence of all fluctuations, however, this assumption necessitates a justified rationale. Overlooking fluctuations without solid reasoning can oversimplify the model, potentially leading to inaccuracies. Therefore, it is important to consider when and why such an approach is taken, as the assumption to disregard fluctuations may not always yield accurate results.

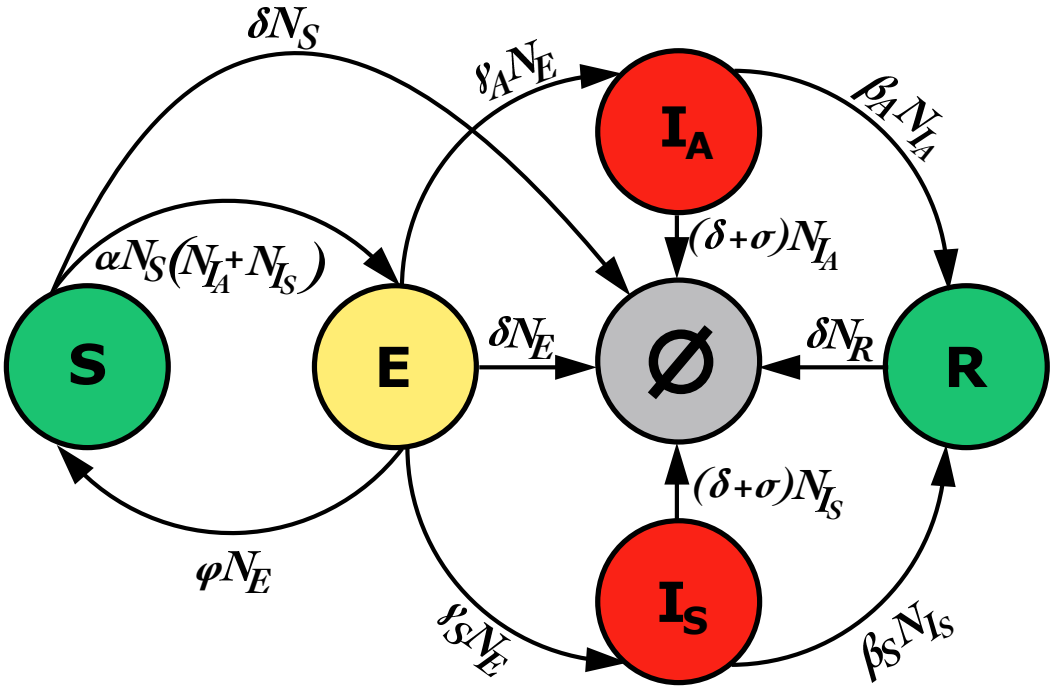
The specific case of the moment evolution of the first-order reaction system was studied and described in detail in [12].

Due to the difficulty of solving CME (1.34) and evolution equations for the mean and variance, it is often more practical to generate simulated trajectories of $X(t)$ over time instead. The numerical methods used to generate the stochastic trajectories are further discussed in Chapter 3.

Agent-based approach

Although the well-stirred approach of compartmental models is a popular tool to study spreading processes of different origins, its limitations make it less suitable in certain cases. That is, it does not include factors of population heterogeneity, but in reality, individuals differ in their characteristics, behaviour, and susceptibility to diseases and other phenomena.

Population heterogeneity can be crucial to understand disease transmission dynamics and to evaluate interventions. The centrality of population heterogeneity can be expressed,



Reactions

$r_1 : S \xrightarrow{\delta} \varnothing$ $\quad r_5 : R \xrightarrow{\delta} \varnothing$ $\quad r_9 : E \xrightarrow{\gamma_A} I_A$

$r_2 : E \xrightarrow{\delta} \varnothing$ $\quad r_6 : S + I_A \xrightarrow{\alpha} E + I_A$ $\quad r_{10} : E \xrightarrow{\gamma_S} I_S$

$r_3 : I_A \xrightarrow{\delta+\sigma} \varnothing$ $\quad r_7 : S + I_S \xrightarrow{\alpha} E + I_S$ $\quad r_{11} : I_A \xrightarrow{\beta_A} R$

$r_4 : I_S \xrightarrow{\delta+\sigma} \varnothing$ $\quad r_8 : E \xrightarrow{\phi} S$ $\quad r_{12} : I_S \xrightarrow{\beta_S} R$

Stoichiometry matrix:

$$\mathbb{S} = \begin{array}{c} \\ S \\ E \\ I_A \\ I_S \\ R \end{array} \begin{array}{c} \begin{array}{cccccccccccc} r_1 & r_2 & r_3 & r_4 & r_5 & r_6 & r_7 & r_8 & r_9 & r_{10} & r_{11} & r_{12} \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & -1 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 1 & -1 & -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array}$$

Fig. 1.2. Reaction-diffusion SEIR model of the pathogen spread. Inspired by [13], in this model the population consists of susceptible individuals $\boldsymbol{S}$, exposed individuals $\boldsymbol{E}$, symptomatic $\boldsymbol{I_S}$ and asymptomatic $\boldsymbol{I_A}$ infected and recovered part of the population $\boldsymbol{R}$. The state $\boldsymbol{\varnothing}$ denotes deceased individuals. Susceptible individuals, who have been in close contact with the infected, whether or not those infected are displaying symptoms, are at risk of being exposed to the pathogen and, therefore, move to the exposed compartment. Some exposed individuals do not become infected and can transition back to the susceptible population, while the remainder move to either the asymptomatic or symptomatic infectious population. Most infectious individuals recover and move to the recovered compartment. All individuals are affected by natural death, with the infected being at a higher risk because of the disease. The state graph depicts a Markov jump process describing transitions between population compartments; its edges are labelled with the transition rates which are at the same time reaction propensities for the depicted reactions. Population changes for all compartments are given by the stoichiometry matrix $\mathbb{S}$.

for example, in the different transition rates for each individual in the compartment. If the modelled system is subject to any kind of spatial dynamics, it is not always possible to incorporate it into the compartmental models, but it could significantly influence the pathogen's spread.

In many infectious diseases, the network of contacts between individuals and behavioural responses is crucial to disease transmission. The well-stirred compartmental model assumes that everyone is equally likely to interact with everyone else, but in reality, individuals exhibit different contact patterns and behavioural changes in response to disease outbreaks and interventions.

As a result of these limitations, an *agent-based approach* may be necessary to capture the dynamics of the system accurately [14–16]. This modelling technique focusses on individual agents x_i and their interactions with each other and their environment. In these models, each agent is in a specific state, and the transitioning from one state to another depends on the current state of the system and the behaviour of individual agents. Reaction channels R_k in this case are defined for each individual transition (i.e., transitions of each individual agent). The Markov process in this case describes the state of the whole system $X = x_i$. For example, consider a model of disease transmission in a population of individuals. Each individual is in one of three states: susceptible, infected, or recovered. The probability of an individual transitioning from the susceptible state to the infected state depends on the contact patterns and behaviour of the individual, as well as the current state of the population as a whole. The transition probabilities for the Markov process can be calculated based on the behaviour of individual agents and their interactions with each other.

1.5 Summary

Markov models can powerfully model the spreading process because they allow to analyse and predict the behaviour of a system over time, based on its current state. They are particularly useful for modelling the spread of infectious diseases, because they can incorporate important factors such as the transmission rate and the probability of recovery or death. The framework of the Markov models allows representation using several mathematical and computational techniques, including transition matrices, differential equations, and Monte Carlo simulations. They also allow for the incorporation of uncertainty and variability in the modelling process, crucial to accurately capture the behaviour of real-world systems. Overall, while Markov models have many advantages and are widely used in many fields, it is important to carefully consider their limitations and ensure that the assumptions and limitations of the model are appropriate for the specific application. Although this work

focusses on Markov processes, the findings may be adapted to non-Markovian processes to some extent, as discussed in Chapter 7.

Chapter 2

Current approaches in contact network modelling

2.1 Introduction

Many diseases are transmitted through physical contact, droplets, or contaminated surfaces. When an infected person comes into contact with a susceptible person, the infectious agent can be transmitted from one person to the other. For example, respiratory diseases such as COVID-19 or influenza can spread through respiratory droplets expelled when an infected person coughs or sneezes. These droplets can land on the mouth or nose of another nearby person, allowing the virus to enter their body and potentially cause an infection. Other diseases, such as sexually transmitted infections (STIs), require sexual contact between individuals for transmission to occur. For instance, HIV is transmitted through the exchange of bodily fluids during sexual activity or through the sharing of contaminated needles. In some cases, diseases may also be transmitted indirectly through contaminated surfaces or objects, such as in the case of foodborne illnesses. In these cases, contact with the contaminated surface or object is necessary for transmission to occur. Overall, the mode of transmission and the infectiousness of the disease will determine the importance of contact in the disease's spread.

Contacts between individuals within the studied population can be formalised as graphs where the nodes represent individuals or entities, and the edges represent the connections or interactions between them. This representation is called a *contact network*. The contact network topology and its change over time are important in understanding the spreading process of infectious diseases. They determine how easily and quickly the disease can propagate through the population [17–20]. For example, if the contact network is highly

clustered, with individuals tending to interact primarily with their immediate neighbours, then infectious diseases may spread quickly within tightly knit communities, but may not spread easily beyond those communities [21–23]. In addition, the presence of hubs, or highly connected individuals in the contact network, can greatly enhance the spread of disease. Hubs can act as "super-spreaders", transmitting the disease to numerous individuals with whom they are in contact [24, 19].

In reality, the contact network is not fixed and may change over time due to factors such as changes in social behaviour, migration, or interventions such as vaccination or social distancing. These changes can have important consequences for the spread of infectious diseases [25–28].

Obtaining an accurate contact network model requires having information about all individuals in a population and all instances of disease transmission through contact, such as sneezing or coughing for airborne diseases or sexual contact for sexually transmitted diseases (STDs). However, this task is often shown impractical, even for populations of relatively small sizes, so different approximation techniques are usually used to represent relevant population connectivity and, if necessary, its evolution over time.

This chapter synopsises the various methods used to incorporate population connectivity into models of disease spread. The first approach discussed in section 2.2 is the classical homogeneous mean-field approximation, which assumes a uniform population and does not consider network structure. The model is refined to incorporate the static network structure, which has been effective in many cases. However, this model lacks temporal information essential to understanding the causal paths that underlie the spreading process.

To address this limitation, section 2.3 introduces the temporal network model, which captures both the static and temporal aspects of network structure, enabling the modelling of dynamic spreading processes. Section 2.4 presents an important extension that models spreading on contact networks by including adaptive behaviour, leading to the class of adaptive networks.

2.2 Static network models

2.2.1 Mean-field approximation

In the context of epidemiological modelling, mean-field approximation is used to simplify the analysis of disease transmission dynamics by applying a number of assumptions to the modelled population. It presupposes population homogeneity, meaning that all individuals in a population have identical interaction patterns and susceptibilities to the disease (as briefly

discussed in section 1.4.7). It also assumes a random mixing, meaning that interactions between individuals are random and occur with equal probability, regardless of their location in the contact network [29]. Additionally, the approximation assumes that the population size is large enough that the effects of individual interactions on disease transmission can be treated as continuous variables. Finally, it requires a well-mixed population, meaning that the probability of disease transmission between any two individuals is independent of the presence or absence of other individuals in the population. The contact network of the population in this case may be represented as a complete graph, where nodes represent individuals and edges between them represent contacts [30].

Stochastic models with the mean-field approximation may be approximated with classic deterministic compartmental models, such as SIS, SIR, or other compartmental ODEs [8, 31], assuming the fluctuation is minimal and can be neglected [29, 32]. Deterministic models tend to overestimate the true stochastic process [30].

2.2.2 Random graph models

In the context of disease transmission, it is unlikely, however, that individuals are in constant contact with all other individuals, particularly in large populations or for diseases transmitted through sexual contact. Empirical evidence suggests that contact networks for STIs are rather sparse [33–35].

To account for this sparsity, a random network model, first proposed by Erdös and Rényi [36], may be considered. This model constructs a random graph by starting with N disconnected nodes and then connecting each pair of nodes with some arbitrary probability p, or not connected with the complementary probability $1-p$. Since each edge in this graph has the same probability, the algorithm results in a homogeneous graph. The average degree (i.e., the number of contacts) of each node $\langle k \rangle = p(N-1)$ is determined by this probability and, for a large population number, converges to a constant, meaning that every node in the network has exactly $\langle k \rangle$ contacts.

Since this is a Bernoulli process, degree distribution is given in the binomial form, and with the large N and fixed p can be approximated by a Poisson distribution:

$$Pr(k) = \binom{N}{k} p^k (1-p)^{N-k} \approx \frac{\langle k \rangle^k \exp(-\langle k \rangle)}{k!}.$$

For fixed probability p, this equation defines the collection of graphs $\{G_{N,p}\}$, where graph with exactly m edges is constructed with the probability $p^m(1-p)^{M-m}$, $M = N(N-1)/2$ being the maximum number of edges.

A series of works [37–40] has shown that for large graph sizes, several properties of random graphs, such as clustering coefficient and the expected size of the giant component, can be determined exactly.

Bollobás [41] has proposed an extension of the homogeneous random graph model to an inhomogeneous one, based on a prescribed kernel. Rather than relying on an arbitrary probability value p, the connection probability between any two nodes is determined by a non-negative, bounded kernel function $\kappa(x,y)$ that is symmetric with respect to its arguments (x,y).

Another approach to constructing a homogeneous random graph is the Waxman model [42]. In this model, N nodes are uniformly placed in a rectangular domain, and the connection probability between any two nodes v_i, v_j depends on their Euclidian distance d_{ij}:

$$Pr(i,j) = \beta \exp\left(-\frac{d_{ij}}{\alpha L}\right),$$

where L is the maximum distance between two nodes, and $\alpha, \beta \in [0,1)$ are model parameters governing the edge density.

The Waxman model generates a homogeneous network with an exponential degree distribution, indicating that the probability of a node having a degree different than the mean degree $\langle k \rangle$ decays exponentially.

The random graph models exhibit the *small world effect*, meaning that the average shortest path length $\langle l \rangle$ scales logarithmically $\langle l \rangle \approx \log N / \log \langle k \rangle$ or slower with network size for fixed mean degree [43, 39]. However, it fails to capture important features of real-world networks, such as high clustering coefficient, non-Poisson degree distribution, and community structure [44].

Although random graph models may not capture the full complexity of real systems, they serve as useful tools for analysing and evaluating dynamic processes in networks. The homogeneity of the model simplifies mathematical modelling, making it easier to understand and interpret results. The presence of correlations also makes it a valuable structural model for assessing the accuracy of dynamic models. Despite its limitations, the random graph model provides valuable insights into the behaviour of dynamic processes in networks.

2.2.3 Small-world network

The small-world network model was first proposed by Watts and Strogatz [45]. It involves an ordered lattice, such as a ring of N vertices, where each vertex is connected to its neighbour h or fewer lattice spacings away. The model then introduces a rewiring process, where

each edge is reconnected to another vertex with a probability of p, except for self-loops and multiple edges.

The rewiring process transforms a regular lattice into a structure similar to, though not identical to, a random graph. When $p = 0$, the graph remains a regular lattice with a high clustering coefficient that tends to $3/4$ for large values of h. However, the average distances between vertices are also high, tending to $N/4h$ for large N [39]. On the other hand, when $p = 1$, all edges are rewired, and the resulting graph becomes almost random with typical distances on the order of $\log N/\log h$, but with a relatively low clustering coefficient of $\approx 2h/N$. Watts and Strogatz demonstrated numerically that the small-world model exhibits low path lengths and high clustering for intermediate values of p.

To address some of the limitations of the original model, Monasson [46] and Newman and Watts [47] proposed a modified version where pairs of vertices are randomly selected and connected with a probability of p, but no rewiring is involved. This modification significantly reduces the average shortest path length while still maintaining a large clustering coefficient for moderate values of p.

The small-world model produces homogeneous networks where each metric has a typical value shared by all nodes with minimal variations [32].

2.2.4 Degree sequence models

Empirical findings from various research fields indicate that many real-world networks exhibit a *scale-free* structure, characterised by a heavy-tailed degree distribution that often follows a power-law function of the form $P(k) \sim k^{-\alpha}$, where the value of parameter α is typically in the range $2 < \alpha < 3$ [48]. In contrast to homogeneous graphs, where the degree distribution centres on the average degree $\langle k \rangle$, heterogeneous networks display a diverse range of node degrees.

To construct networks with non-Poissonian degree distributions, the *configuration model* [49] can be used. This model allows one to construct a graph with a given degree distribution in two steps:(i) based on a distribution $P(k)$, each vertex is assigned a degree (number of outgoing edges) k_i (ii) the pairs of outgoing edges are chosen randomly from the network and connected together. However, for power-law degree distributions with $\alpha \leq 3$, the original configuration model leads to the formation of networks with loops and multiple edges. This problem can be addressed by imposing a structural cut-off, such as setting the maximum degree to $k_{max} \sim \sqrt{N}$, which limits the average degree to $\langle k \rangle = 1/\sqrt{N}$ to prevent the formation of loops and multiple edges in the network [50, 51]. Newman [52] has proposed an extension of the configuration model allowing to incorporate a clustering structure in the modelled network.

The deterministic Havel-Hakimi algorithm, published by Havel [53], and later by Hakimi [54], constructs a simple graph for a given degree sequence $K = \{k_1, k_2, \ldots, k_n\}$ by successively connecting the node of the highest degree to other nodes of the highest degree, resorting remaining nodes by remaining degree and repeating the process. Nodes are labelled with integers that correspond to the index of their expected degree in the input sequence.

Chung and Lu [55] used a random model enabling the construction of graphs with a given sequence of expected degrees, denoted by $K = \{k_1, k_2, \ldots, k_n\}$. The model constructs these graphs by assigning an edge between nodes v_i and v_j with a probability proportional to the product of their expected degrees (i.e., $k_i k_j / \sum_s k_s$). The nodes in the graph are labelled with integers corresponding to the index of their expected degree in the input sequence. The model accounts for the possibility of self-loop edges. However, for finite graphs, this model does not precisely generate the given expected degree sequence. Instead, the expected degree of node v_j is given by $\mathbb{E}[deg(v_j)] = k_j \left(1 + k_j / \sum_i k_i\right)$.

2.2.5 Growing network models

The model of Barabási and Albert [56] belongs to the class of growing network models. This model recognises that numerous real-life networks do not maintain a constant number of nodes and edges. Instead, they evolve over time, with new nodes and links continually being added. This undirected network model utilises the *preferential attachment* approach, first observed and proposed by Price [57, 39]. The original model assumes that the probability of a newly added vertex v_j connecting to some already existing vertex v_i depends linearly on the degree k_i:

$$Pr(v_i, v_j) = \frac{k_i}{\sum_s k_s}. \tag{2.1}$$

The construction of a network of the desired size obeys the following rules: (i) The network starts with some n_0 randomly connected vertices. (ii) At each following step the new vertex with some $m < n_0$ edges is added. (iii) New edges are connected to the node v_i in the network with probability given by 2.1. The most connected vertices are more likely to receive new connections. Thus, networks generated by this model present a power-law degree distribution $P(k) \sim k^{-\alpha}$ with $\alpha \to 2.9 \pm 0.1$ [56].

The initial growing network model has received great attention in the field. Many variations and extensions of the Barabási–Albert model have been proposed to incorporate diverse properties, such as high clustering and adjustable degree-degree correlations, as well as variations in the degree distribution exponent [58, 39].

The Spatial scale-free (SSF) network model, proposed by Barthélemy [59] constructs a scale-free network of desired size N nodes. First, N nodes are distributed randomly

(uniformly or following the other distribution) in the d dimensional space of linear size L. Then the following algorithm is applied: (i) a random subset of n_0 active nodes is selected, and (ii) randomly inactive node v_i is selected and connected with the "active" node v_j with the probability:

$$Pr(v_i, v_j) \propto \frac{k_j + 1}{\exp(d_{ij}/r_c)}.$$

Here, k_j is a degree of the active node v_j, d_{ij} is a Euclidian distance between them, and r_c is a finite scale parameter, governing the clustering coefficient and the assortativity of the network. (iii) The node v_i is labelled as "active". Steps (ii)–(iii) are then repeated until all nodes are "active". To achieve the average degree $\langle k \rangle = 2m$, steps (ii)–(iii) are performed m times for each node.

The Dorogovtsev–Goltsev–Mendes procedure [60] constructs a deterministic pseudofractal graph by starting with a triangle graph (with three nodes and three edges), adding one vertex at the time and connecting it to two already existing adjacent vertices.

2.2.6 p* models

First proposed by Frank and Strauss [62, 63] and Frank [62] and then formulated by Wasserman and Patisson [64], p* models, also often called exponential random graph models, are a family of network models aiming to model the probability of observing a network structure using nodal and dyadic covariates.

Exponential random graph models (ERGM) have the following form for the probability of observing a graph x:

$$Pr(X = x) = \frac{\exp\left(\theta^T z(x)\right)}{\kappa} = \frac{\exp\left(\theta_1 z_1(x) + \theta_2 z_2(x) + \ldots + \theta_r z_r(x)\right)}{\kappa},$$

where $z(x)$ is the vector of the r of measurable properties of a graph (e.g., number of edges or diads, number of starts or cycles; more detailed review can be found in [65]), θ is a vector of unknown model coefficients that must be estimated, and κ is a normalising constant that ensures that the probabilities sum to unity.

To construct a network, the following five steps are executed [66]: (i) For a fixed set of N vertices, it is assumed that a contact, or an edge between each pair (i, j) is a random variable X_{ij}. This variable takes a value of 1 if i and j are connected and 0 otherwise. If the constructed network is undirected, then $X_{ij} = X_{ji}$, or directed, implying that $X_{ij} \neq X_{ji}$. (ii) A dependence hypothesis is chosen. For instance, ties may be assumed to be independent of each other, but they may also depend on node-level attributes such as age, gender, or number of already existing connections. (iii) The chosen dependence hypothesis implies a particular

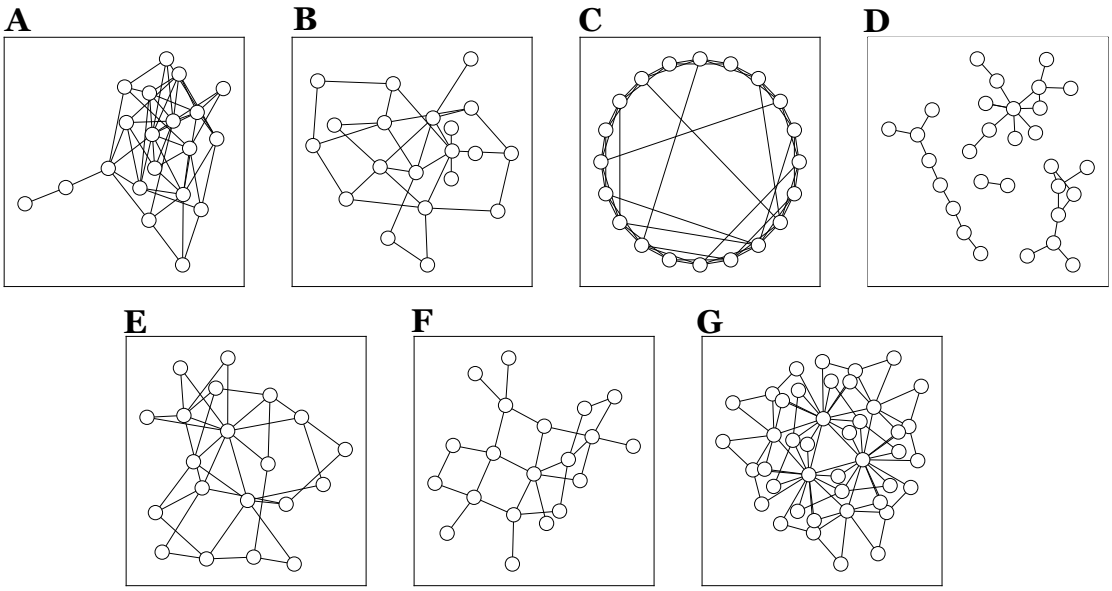


Fig. 2.1. Static network models. **(A)** Erdös-Rényi random graph model of $N = 20$ nodes and the edge probability $p = 0.3$. **(B)** Waxman graph model for $N = 20$ nodes and model parameters $\alpha = 0.5$ and $\beta = 0.5$. **(C)** Newnan's modification of the small-world model (without reconnection of the edges) for $N = 20$ nodes connected to their 4 nearest neighbours, and with the probability $p = 0.2$ for the random edges. **(D)** Configuration model network for $N = 20$ nodes with a power-law degree sequence. **(E)** Barabási – Albert model for $N = 20$ nodes and $m = 2$ edges of each newly appearing node. **(F)** Network constructed using the duplication-divergence model of Isopatov [61] for $N = 20$ nodes and the probability for retaining the edge of the replicated node $p = 0.2$. **(G)** Pseudofractal network constructed using the Dorogovtsev–Goltsev–Mendes procedure, repeated for 4 steps and resulted into $N = 42$ nodes.

form of the model. The model then represents a distribution of random graphs assumed to be "built up" from the localised patterns represented by the configurations. (iv) The parameters are simplified through the introduction of constraints. To reduce the number of parameters, heterogeneity may be introduced. Some parameters may be reformulated in other ways. (v) The model parameters are estimated and interpreted with pseudo-likelihood estimation or Monte Carlo maximum likelihood estimation. A comprehensive review of the dependent hypotheses and parameter estimation methods and particular subclasses of the ERGM, such as bipartite random networks and Markov random graphs, can be found in [67, 65, 68].

2.2.7 Duplicating growths methods

Networks such as biochemical interaction networks have power-law degree distributions but do not fit the preferential attachment model. Kleinberg et al. proposed a hyperlink-induced

topic search (HITS) model [69] to explain the growth of directed networks like the Web, suggesting that they grow by adding vertices and copying edges from existing vertices. The procedure involves the following steps: (i) A pre-existing vertex is selected, and the number of edges m to be added to it is determined. (ii) The destination of these edges is decided by randomly choosing another vertex and replicating the targets of m of its edges. If the selected vertex has fewer than m outgoing edges, then its edges are copied and the process is repeated for another vertex until m edges in total have been duplicated. The degree distribution is a power law with an exponent $\alpha = (2-a)/(1-a)$, where a is the ratio of the number of edges added whose targets are chosen at random to the number whose targets are copied from other vertices [39].

A modified version of the concept of vertex copying is present in the autocatalytic network models proposed by Jain and Krishna [70], as well as in protein interaction network models investigated by Isopatov [61] and random growing graphs investigated by Knudesen and Wulf [71].

2.2.8 General modifications of graph models

While the general definition of network models is typically presented for the case of undirected networks, the specific definition of transmission-relevant contacts and the population being modelled may require a transformation to directed or hybrid networks with both directed and undirected edges [72].

Weighted networks are another important extension, of the general network models, where a weight $\omega_{i,j}$ is assigned to the edge between vertices to represent, for example, the intensity or frequency of contacts [32].

Moreover, in some cases where the mode of transmission or the multiple pathogen-host dynamics require more complex models, multilayer network models are a valuable tool by which to study complex spreading processes. A comprehensive review of multilayer network models can be found in Kinsley et al. (2020) [73].

2.3 Temporal and dynamic network models

Up to this point, the described models have operated under the explicit assumption that the network is stationary, implying that the rate of partner turnover is negligible and has no effect on spreading dynamics. Nonetheless, a critical aspect of numerous real-world networks is the temporary nature of certain connections. To capture the temporal causality of the underlying system, different time-evolving network models have recently been introduced

[26, 74, 75]. If the network's contact frequency changes slowly, its influence on the spread of the disease can be ignored, and one of the earlier described static approximations can be applied. In contrast, if individuals change partners rapidly enough to disregard the possibility of a single edge transmitting twice, the network can be approximated using a time-averaged version. However, when the network and the process evolve at comparable rates, their interaction becomes significant. For instance, an infected individual's connections may undergo substantial changes during their infectious period. In such cases, temporal networks enable one to capture the temporal causality of the spread. A temporal or dynamic network is a network structure that changes over time, with the nature of the change and the notation used for timing varying widely and depending on the data and application.

A *temporal network* is a graph $G(t) = \{V, E(t)\}$, consisting of a set of vertices V and a collection of temporal edges, $E(t)$. Each temporal edge is a trio (v_i, v_j, t), where v_i and v_j are vertices in V and t is a time component, indicating that nodes v_i and v_j are in contact at time t. Time component t can either be a time stamp, meaning that the time is discretised and the network is updated at fixed time intervals, or an interval, implying that the network evolves on a continuous time scale. [76, 77]. This definition assumes that the set of vertices V does not change over time, distinguishing dynamic networks and temporal networks; however, it is possible to incorporate the change in vertices in the model. A static graph for some fixed t is called a snapshot. Figure 2.2 depicts time-varying networks represented as a series of snapshots, one for each time step.

There are several general ways to represent temporal networks graphically, such as with time-node graphs or temporal tensors, and the comprehensive review can be found in [76].

Several books and comprehensive reviews have focussed on the network structure and modelling techniques for temporal networks[20, 26], as well as on the epidemiological and other spreading processes, happening on the dynamic and temporal networks[78, 27].

Temporal network modelling is not a straightforward generalisation of static network models, and relevant analytical findings and techniques cannot be transferred directly to these classes of models, primarily because the contacts in the temporal networks can be defined on a broad spectrum. In the static network, a link between two individuals represents some kind of personal relationship between them, while the contact in the temporal network can represent either just a one-time contact or a relationship lasting for the time period between the first and last contact. In addition, any indirect connection between two nodes is temporal and time-dependent. The connections must happen along time-respecting paths of contacts (with strictly increasing timestamps). However, even if there are some contacts between nodes v_i and v_j and between nodes v_j and v_k, it may still be impossible that the phenomenon spreads from v_i to v_k, because the infection of v_j can happen too late, and there will be no

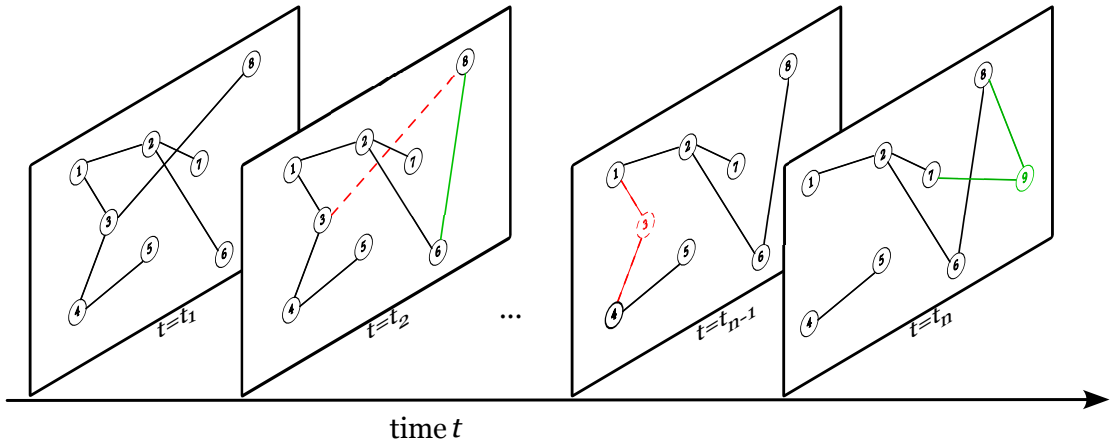


Fig. 2.2. Temporal network model. The temporal network is represented as a sequence of snapshots at time points $\{t_1, t_2, \ldots, t_n\}$. At the time t_2, thr edge between nodes 3 and 8 no longer exists, but the new edge between nodes 6 and 8 appears, though not observed at the time t_1, appears. At the time t_{n-1} node 3 no longer exists, nor its adjacent edges. At the time t_n, a new node 9 appears, with the new adjacent edges, connecting it to nodes 7 and 8.

more contact to v_k. The most common generalisation of distance is *latency* or *temporal distance* – the time it would take to reach v_j from v_i starting at time t and following only time-respecting paths.

There are several modelling techniques for generating a temporal network. A few are described below, and a more comprehensive review can be found, for example, in [26, 20].

A straightforward approach to creating a temporal network involves first constructing a static network using any of the previously described models, and then transforming each edge in the network into a temporal edge by assigning a sequence of contacts over time to it. Holme [79, 80] proposed the following technique: (i) generate some static network of choice, (ii) for every edge randomly generate its activity time interval, (iii) generate a sequence of contact times, and (iv) match the sequence of contacts to the activity intervals. Furthermore, randomisation techniques such as *randomized edges* or *randomly permuted times* are used primarily to analyse contact datasets and the impact of the topology of the network on the spreading process, but can also be used to generate a temporal network [26].

Another approach, used by Perra et al. [81], proposes to (i) clear all edges of the graph at each time step and (ii) consequently activate each node v_i with the probability $\alpha_i \Delta$ for a time window of a given length Δ, connecting it to the m other nodes. Here, α_i is a firing rate of the node v_i.

Past studies have examined spreading processes on time-evolving networks by applying several types of rewiring rules based on averaged group behaviour, but the agent-based approach has recently gained prominence. For example, Vestergaard [82] has proposed

a model where the activation rate of both nodes and links is proportional to their "age" – the time period Δ_i since their last activity. The model operates as follows: (i) Initially, the network comprises N nodes and all links are inactive. (ii) Node v_i uses its last contact involvement time Δ_i as a reference point to activate a new link. The link is chosen from the nodes v_j that are currently not in contact with it, with a probability depending on the Δ_j of these nodes. (iii) An active link is inactivated with a rate that is also dependent on $\Delta_{i,j}$. Starnini [83] used a two-dimensional spatial model where each agent is characterised by their social attractiveness parameter α_i. Agents perform a random walk, and as soon as the distance between two agents is less than some set value d, they start to interact. The more attractive an agent v_i is, the more their interaction partner v_j is affected, who will slow their exploration random walk accordingly. A similar model, but without the attractiveness component was used by Djurdjevac-Conrad [84] to study the spread of innovations spreading in ancient times and by Nadini [75] to study the epidemic spreading and vaccination strategies in an urban-like environment.

It is also possible to combine static and temporal edges in one model, as, for example in [74] or [85] where the links within "household" communities stay unchanged, but the links between agents on a "social" level change with time.

The agent-based models provide a detailed representation of a population of interacting individuals. Agents can represent anything from cells to animals, each following a set of rules controlling their behaviour and interactions. This approach allows for the inclusion of heterogeneity and stochasticity, and environmental processes can be more easily incorporated. Agent-based models are well suited to explain how complex group-level features emerge from individual behaviour, and they capture features that a mere averaging across the population would miss. [86, 16].

The timing of human behaviour, however, often does not follow a Poisson distribution, but can be better described with a heavy-tailed power-law distribution [87, 88], meaning it happens in concentrated bursts, interspersed by long waiting periods. This property is called "burstiness", and has been shown to significantly impact the spreading dynamics [89]. Burstiness imposes certain challenges on simulation techniques that can be used to capture the power-law waiting time distribution. This challenges are touched upon in Chapter 7.

2.4 Adaptive network models

Incorporating temporal network structure and evolution into the study of the spreading process has improved overall prediction accuracy, but the assumption that contact dynamics are independent of what spreads on the network is likely to be invalid in many cases [20].

With the increase in travel activity at both short and long distances, individual behaviour increasingly shapes the spread of epidemic diseases [90, 91]. Furthermore, the high awareness of society due to easy access to disease-related information through mass media and the internet may affect their behaviour in response to an outbreak. For instance, individuals may alter their travel plans, practice self-isolation, avoid contact with infected persons, or seek vaccination. Additionally, the health status of individuals frequently affects their behaviour, limiting their mobility and reducing their likelihood of infecting others [91].

Various methods have been employed to incorporate changes in behaviour into epidemic models [78]. These methods include modifying contact rates based on health status [92], adding new compartments to compartmental models, and appropriately linking models of disease and information propagation [93]. A pivotal step in this direction is the concept of adaptive, or co-evolving, networks, initially proposed by Gross [28, 94]. These networks' structures are dynamically shaped by the disease-spreading process [16].

By integrating the principle of adaptivity into temporal network modelling, researchers can capture the network's ongoing structural evolution in response to disease spread. This approach is instrumental in examining the feedback mechanisms between the spread of the phenomena and network structure. The principal advantage of co-evolving networks lies in their ability to capture the dynamics of disease transmission and the emergence of transmission patterns over time while incorporating temporal modifications in response to the presence of an infectious agent or some kind of intervention, offering critical insights into the efficacy of control measures [78, 29, 22, 95, 16].

Human interaction networks inherently possess a stochastic nature, as they involve numerous interacting individuals whose behaviour can be influenced by many factors, including random events and external stimuli. These factors can lead to unpredictable and random outcomes in social interactions, making it difficult to accurately predict the behaviour of individuals in a social network. Furthermore, the formation and dissolution of relationships, changes in individual behaviour, and external events may also occur randomly or in response to unpredictable events.

Consequently, *stochastic* and *agent-based* modelling techniques are often used to model human interaction networks to capture the inherent uncertainty and variability in social interactions and network dynamics. These methods can help researchers to better understand the complex behaviour of social networks and the individuals within them, as well as to make predictions about their future behaviour and the spread of the phenomenon through the population.

Nevertheless, a primary challenge in the development and implementation of time-evolving adaptive network models in epidemiology and other research fields lies in balancing

model complexity and computational efficiency with the need for the accurate representation of spreading dynamics. Time-evolving adaptive network models can improve the understanding of transmission dynamics and shape effective interventions to mitigate, prevent, or control the spread of phenomena of different origins.

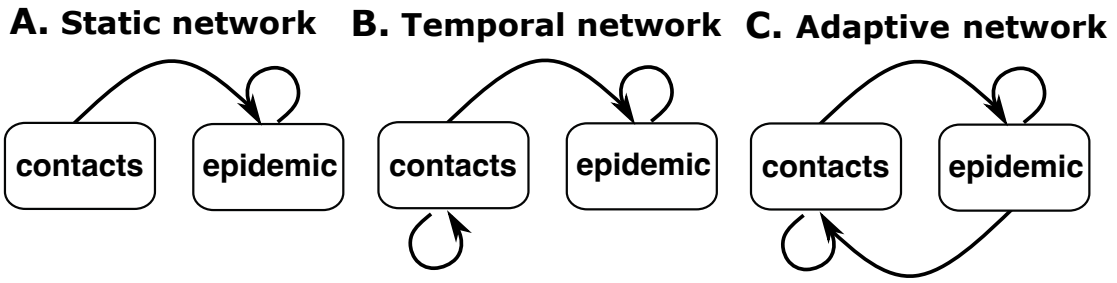


Fig. 2.3. Different approaches for modelling spreading processes on networks. (A) In static network approaches, the network topology, which determines the linkage and thus the spreading process, remains fixed. **(B)** In temporal networks, the network topology evolves according to some *contact dynamics*, which is independent of the *epidemic dynamics*. **(C)** Adaptive networks denote a class of co-evolving stochastic processes, where the contact structure affects the spreading process and where the spreading process alters the contact dynamics.

2.5 Summary

Incorporating contact networks in the model of the phenomena spreading is crucial to the study of the spreading processes of diseases, information, and opinions. Chapter 2 differentiates existing modelling approaches for contact networks, categorising them into static, temporal, and adaptive models. Static models offer insights into network structures, and have their place in certain applications, such as in modelling social networks and analysing historical data. Temporal models introduce the crucial element of time and allow for capturing the dynamic nature of contact networks. However, the most noteworthy are adaptive network models. These models excel in reflecting how networks evolve in response to spreading progression, capturing the dynamic interplay between spread of the phenomena and network adaptation. This adaptivity is especially pivotal for accurate prediction of disease transmission and evaluating the efficiency of medical interventions such as vaccination or treatment, as well as non-medical interventions such as social distancing and introduction of the spreading barriers (for instance, masks during the COVID-19 pandemic or usage of condoms to prevent the spread of STIs).

Chapter 3

Existing computational methods and their application to simulate spreading processes on complex systems

3.1 Introduction

The study of complex systems has been a challenging task for scientists across various fields, as they often involve intricate relationships between multiple entities. Agent-based models (ABMs) have emerged as a powerful tool to model such systems, as they allow for the representation of individual entities and their interactions in a more realistic and nuanced way. The impact of incorporating community structure with its stochasticity was discussed in the previous chapter. However, as the complexity of these models increases, it becomes more challenging to accurately analyse system behaviour.

Stochastic simulation methods address the challenge of simulating community dynamics and spreading processes in a stochastic framework, as they introduce randomness into the model to account for the inherent uncertainty and variability in the system. These methods have been widely used in the field of ABMs, but their application to time-varying adaptive networks has not yet been fully explored.

In ABMs of complex systems with contact networks, the interaction process is often modeled stochastically to account for the heterogeneity of both the agents and the network structure. As such, not only may the spreading events occur randomly, but also contacts between individuals are established and discarded stochastically.

This chapter explores the application of stochastic simulation methods in the context of ABMs with time-varying adaptive networks. Different methods are investigated, including

stochastically exact algorithms, approximate approaches, and hybrid methods that can account for the impact of the dynamic environment. For a comprehensive and detailed review, please refer to the studies [96–99].

3.2 Exact methods

Exact stochastic simulation algorithms are a class of computational methods used to simulate the dynamics of reaction-diffusion Markov models from various scientific fields. These algorithms are designed to exactly simulate the probability distribution of the state of the system over time, based on a given set of reaction rules and initial conditions.

3.2.1 Direct method

The most well-known exact stochastic simulation algorithm is Gillespie's algorithm, also called the SSA or a direct method (DM). It was originally developed to simulate the dynamics of biochemical reaction networks in a stochastic manner [5]. The algorithm works by simulating the occurrence of individual chemical reactions in the system, one at a time, based on their propensity functions.

The key element in generating the numerical realisation of the process X_t is a joint probability density function of the time to the next reaction (τ) and the index of the next reaction (j), given the current state of the process at the time t:

$$Pr(\tau, j \mid X(t),t) = Pr(\tau \mid X(t),t) \cdot Pr(j \mid \tau; X(t),t) = a_j(X(t)) \cdot \exp(-a_0(X(t))\tau), \quad (3.1)$$

$$\text{where } a_0(X(t)) = \sum_j a_j(X(t)) \text{ defines the total of all propensities at the time } t. \quad (3.2)$$

Given the PDF 3.1, it is possible to derive the probability density function of the next jump:

$$Pr(\tau \mid X(t),t) = \sum_j Pr(\tau, j \mid X(t),t) = a_0(X(t)) \cdot \exp(-a_0(X(t))\tau),$$

meaning that the time to the next reaction is an exponential random variable with intensity $a_0(X(t))$. Furthermore, the probability density$Pr(j \mid \tau; X(t),t)$ can also be expressed as follows:

$$Pr(j \mid \tau; X(t),t) = \frac{a_j(X(t))}{a_0(X(t))}. \quad (3.3)$$

Using these theoretical insights, the algorithm proceeds in a series of steps, where at each step, the time of the next reaction is sampled from the exponential probability distribution

with intensity $a_0(X(t))$. The identity of the next reaction to occur is chosen with reference to the probability (3.3). The state of the process $X(t)$ is then updated according to the stoichiometry of the reaction that occurred.

Gillespie's algorithm is exact in the sense that, given enough computational resources, it will simulate the exact time course of the system without any approximation. However, to effectively implement Gillespie's direct method in ABMs to analyse stochastic spreading processes across complex stochastic networks, it is important to account for all potential reactions. For network models, it is often necessary to differentiate between individuals and, occasionally, between various edges. This is because varying neighbour sets and personal characteristics can influence infection probabilities and the overall dynamics of the spread. In these models, some reaction channels are attributed to nodes, while others pertain to the state of the edges, which may toggle between "on" (presence of the edge) and "off" (absence of the edge) [100, 97].

Typically, all reactions R in such models can be categorised into two primary types: state-changing events R^s and interaction or contact events R^c. Given the diverse nature of agents, a single reaction R^c_{jk} often represents only one instance of the formation or dissolution of a connection between two individuals i and j. Similarly, a single reaction R^s_i denotes the state transition of an individual agent i. Consequently, this approach requires the inclusion of up to $N(N-1)/2$ distinct R^c type reactions and $N \times M^s$ individual state change reactions R^s, where M^s denotes the number of possible state change reactions for a single agent. This comprehensive inclusion of reactions can lead to computational overhead, especially as the population size increases, posing a significant challenge in terms of computational efficiency and resource management.

Algorithm 1. SSA.

1: $t = 0$
2: **while** $t < T_F$ **do**
3: Calculate the reaction propensities a_j for each reaction R_j based on the current state of the network $G(t)$ and the agents
4: Compute the total of the reaction propensities $a_0 = \sum_j a_j$
5: Sample $\tau \sim Exp(a_0)$
6: Choose the next reaction, with probability: $\frac{a_j}{a_0}$
7: Execute the chosen reaction
8: $t = t + \tau$

The direct method allows for the seamless integration of adaptive agent behaviour into simulation. This integration can be done by executing corresponding adaptivity step right after a triggering reaction takes place (line 7 in Algorithm 1).

The direct method has an average complexity of $O(n_{sim}M)$, where n_{sim} is the number of simulated steps and M refers to the total number of all reactions, R^s and R^c, together. As such, the complexity has a linear relationship with the number of reaction channels M. However, the number of events occurring per time unit typically also scales linearly with the size of the population N, leading to computational complexity $O(NM)$ or $O(N^2)$ for sparse networks. The evaluation of the DM within real-world biological models indicates that propensity updates generally account for 65% to 85% of the entire simulation cost, reaching even 99% in certain cases [99]. Certain improvements can be made to enhance simulation performance, such as updating only the rates affected by the event instead of all a_j at each time step (lines 3,4 in Algorithm 1). This update can be done in constant time $O(1)$, if the average number of affected reaction channels remains constant, as stated in [97, 99]. This enhancement, however, will not eliminate the overall linear scaling, meaning that for large M the direct method can be quite slow.

When the reaction propensities have only a few distinct values, a technique to decrease the effective number of reaction channels in the DM is feasible, hastening the selection of the next event. The approach involves grouping reactions with identical propensity values and utilising a rounding operation, which can be quickly executed by a computer, to determine a single reaction to be chosen [101].

To enhance the simulation runtime , a binary tree data structure can be employed to store the reaction propensities a_j. This structure allows for the selection of the reaction channel i that will produce the next event (line 6 in Algorithm 1) in $O(\log(M))$ operations instead of $O(M)$ operations and will speed up the whole algorithm. A more detailed explanation can be found, for example, in [97]. Other data structures and search techniques that can be used to improve the performance of the DM are discussed in [99].

Although modern pseudo-random-number generators have significantly improved performance, it remains a good practice to minimise the number of random numbers generated during simulation. Yates and Klingbeil [102] propose a method to recycle a pseudo-random number in the DM. This approach may help to lessen computation time, but may also lower the accuracy of the outcome, although insignificant inaccuracy [97].

3.2.2 First-reaction method

In his publication [5], Gillespie also introduced the concept of the first-reaction method. This approach involves computing the first putative event time $\tau_j \sim Exp(a_j)$ for each reaction channel, which represents when each reaction would fire assuming no other changes occur in the system. After calculating these putative times for all reactions, the algorithm selects the

next reaction to occur R_j by identifying the one with the smallest putative time $\tau = \min_j \tau_j$, where j ranges from 1 to M.

The first-reaction method also has an average complexity of $O(n_{sim}M)$. In the original algorithm, all propensities are updated, and new putative times are generated every time a reaction occurs. To improve efficiency, similar to the direct method, only the propensities are updated, and new putative times are generated for the reaction channels affected by the reaction. Unaffected reaction channels can have their putative times updated by subtracting τ from the previous value: $\tau_j - \tau$ [97].

3.2.3 Next reaction method

Gibson and Bruck [103] have proposed a significant improvement to the first-reaction method with their next reaction method. Instead of storing putative waiting times τ_j for each reaction channel, they stored the absolute times of the next reaction $t_j = t + \tau_j$, where t denotes absolute simulation time. By using absolute time rather than waiting time, the complexity of the reaction propensities update was reduced to $O(1)$, since few reaction channels are affected by each event, on average. To store and retrieve the putative absolute firing times of reactions, the authors introduced an efficient data structure called an *indexed priority queue*, typically implemented as a binary heap.

The authors also introduced a procedure to recycle pseudo-random numbers when generating new putative waiting times τ_j for reaction channels other than i and affected by the event. This procedure involves rescaling the waiting time using the following formula: $\tau_j^{new} = (a_j^{old}\tau_j^{old})/a_j^{new}$.

The next reaction method uses a dependency graph $\mathscr{D}$, a data structure that precisely identifies which reaction propensities to modify when a given reaction occurs. Using the dependency graph, the algorithm can recalculate only the minimum number of a_j values after a reaction occurs.

Similarly to the direct method, the next reaction algorithm also allows one to incorporate the simulation of an agent's adaptive behaviour. This integration can be achieved by executing the adaptivity step right after a triggering reaction takes place (line 7 in Algorithm 2) and correspondingly updating a dependency graph $\mathscr{D}$, if necessary. All of these improvements reduced the average complexity of the algorithm to $O(n_{sim}\log(M))$.

In 2007 Anderson [9] has proposed a modification of the next reaction method incorporating the support of time-dependent reaction rates.

Algorithm 2. Next reaction method.

$t = 0$
Build the reaction dependency graph $\mathscr{D}$
Calculate the reaction propensities a_j for each reaction R_j based on the current state of the network $G(t)$ and the agents.
Compute all putative times τ_j and store times $t_j = t + \tau_j$ in a priority queue Q.
while $t < T_F$ **do**
    Extract the reaction R_μ with the smallest reaction time t_μ from Q
    Execute reaction
    $t = t_\mu$

    for all other reactions R_j dependent on R_μ according to the dependency graph $\mathscr{D}$ **do**
        Compute a_j^{new}
        if $j = \mu$ **then** ▷ *sample new time*
            $\tau_\mu \sim Exp(a_j^{new})$
            $t_\mu = t + \tau_\mu$
        else ▷ *update times*
            $t_j = t + \dfrac{a_j(t_j - t)}{a_j^{new}}$
        $a_j = a_j^{new}$
        Update values t_j in a queue Q

3.2.4 Rejection-based SSA (RSSA)

To increase simulation speed, Thanh et al. [104] aimed to minimise the number of propensity updates for most of the simulation steps by applying the concept of propensity bounds and fluctuation intervals. Instead of recalculating the precise reaction propensities at every time step, the authors defined reaction propensity bounds, $\underline{a_j}$ and $\overline{a_j}$, to determine the next reaction time and identity. The exact propensity is recalculated only if the propensity bounds test fails. Following the acceptance of the reaction, the RSSA assesses whether the new system state falls within the fluctuation interval. If it does, the algorithm can proceed with the next selection step without recomputing the propensity bounds. However, if the new system state lies outside the fluctuation interval $[\underline{X}, \overline{X}]$, a new interval is established and all propensity bounds must be updated.

The average complexity of the algorithm can be estimated as $O(n_{sim}\alpha M)$, where α is the average number of times the search for the next candidate reaction is performed, and is often bounded by a small constant value [99]. Although the method generates more random numbers than direct or next reaction methods, the decrease in the number of propensity updates significantly outweighs the additional computational cost [96].

Several techniques allowing performance increase (e.g., the incorporation of tree-based or table-lookup search) are available for the RSSA and discussed, for example, in [99]. Later authors have also proposed an extension of the method to incorporate extrinsic noise and delays[105].

In the case of the network, however, one must often consider individual reaction channels separately (e.g., if the transmission of the disease can occur through each contact between infected and susceptible individuals with different intensities, each of these possible transmissions is considered to be a separate reaction channel), significantly diminishing the advantages of the RSSA.

3.3 Approximating methods

Exact simulation methods allow for the detailed documentation of each individual reaction event, providing a thorough understanding of the modeled process. However, for systems of practical interest, compiling this extensive record can be time-consuming due to the large number of reaction events that occur in real-world systems. In many cases, it is sufficient to know the frequency of each reaction channel during a particular time interval. This approach forms the basis of approximate simulation methods, allowing for increased simulation efficiency over (controlled) accuracy reduction.

3.3.1 τ- Leaping method

The τ-leaping method has been proposed by Gillespie [106] in 2001. In this approach, simulation time is split into the leap intervals of length τ, redefined adaptively during the simulation. During each time interval τ a group of reactions is fired simultaneously instead of generating individual reaction events for each reaction.

The length of the leap τ chosen to satisfy the leap condition $\left|a_j(X(t)) - a_j(X(t+\tau))\right| \leq \varepsilon a_0(X(t))$. $j = 1 \ldots M$. This condition reassures that the expected changes in the propensity functions in time τ are small enough, so the propensity of each reaction R_j can be approximated as a constant value $a_j(X(t))$ during the time interval. At each simulation step, for each reaction R_j the number of its firings k_j is sampled from the Poisson distribution $\mathscr{P}\left(a_j(X(t))\tau\right)$ with intensity $a_j(X(t))\tau$, and the state of the system is updated according to the stoichiometry and the number k_j of the reaction that occurred: $X((t)+\tau) = X(t) + \sum_j^M k_j \nu_j$, where ν_j is a change vector corresponding to the reaction R_j. If the conditions for a leap are violated to the point where certain population values become negative, the modifications are reversed, and the value of τ is reduced by multiplying

Algorithm 3. $\boldsymbol{\tau}$-leap algorithm.

$t = 0$
while $t < T_F$ **do**
  Compute reaction propensities $R_{(\cdot,\cdot)}$ and $a_{(\cdot)}$ based on the current contact network $G(t)$
  Compute the sum of the reaction propensities $R_0 = \sum R_{jk}; \quad a_0 = \sum a_\ell$
  Calculate $\boldsymbol{\tau}$ satisfying the leap conditions
  accepted = FALSE
  repeat
    if $\boldsymbol{\tau} < \dfrac{l}{a_0 + R_0}$ **then**
      Execute n steps of the SSA(Algorithm 1)
    else
      Generate M random numbers $k^- \sim \mathscr{P}(R_0^- \boldsymbol{\tau})$, $k^+ \sim \mathscr{P}(R_0^+ \boldsymbol{\tau})$ and $u_j \sim \mathscr{P}(a_j \boldsymbol{\tau})$
      if negative values **then**
        reject changes
        set $\boldsymbol{\tau} = \alpha \boldsymbol{\tau}$
      else
        Update network, performing k^- deletions and k^+ additions of the network edges.
        Execute u_j reactions a_j
        $t = t + \boldsymbol{\tau}$
        accepted = TRUE
  until accepted

it by the coefficient α (originally set to $\alpha = 0.5$). Additionally, the $\boldsymbol{\tau}$-leaping technique will switch to the exact SSA if the anticipated number of changes during the leap is deemed relatively low. In such cases, performing n steps of the SSA (typically set to $n = 100$) is more effective. Although the overall computational complexity of this method still scales linearly with the total number of reactions M due to the requisite recalculation of propensities and their total during each leap, the approach enhances runtime performance. This improvement is primarily attributable to a decrement in the number of simulation steps n_{sim} as a consequence of executing multiple reactions within a single leap.

Selecting a suitable value of $\boldsymbol{\tau}$ that satisfies the leap conditions is crucial for the effectiveness of the $\boldsymbol{\tau}$-leaping algorithm. Opting for a value that is too small often results in switching to the SSA, while choosing a value too large can lead to the violation of the leap conditions, which requires repeating the step with a smaller $\boldsymbol{\tau}$ value and rejecting the changes. An inappropriate choice of $\boldsymbol{\tau}$ can lead to inefficient or inaccurate results alongside other

issues. To address these problems, various studies have proposed modified versions of the $\boldsymbol{\tau}$-leaping method. These modifications include the reaction partition method [107], binomial methods [108, 109], implicit methods [110, 111], K/R/S-leaping [112–114], and a post-leap correcting method [115].

As the reactant populations become very large, the $\boldsymbol{\tau}$-leap method approaches an even faster Langevin simulation method [106]. When the expected number of reactions for each channel during the time interval $\boldsymbol{\tau}$ is much greater than one $a_j(X(t))\boldsymbol{\tau} \gg 1$, the Poisson random variable $\mathscr{P}(a_j(X(t))\boldsymbol{\tau})$ can be well approximated by a normal random variable $\mathscr{N}(a_j(X(t))\boldsymbol{\tau}, a_j(X(t))\boldsymbol{\tau})$. The Langevin method smoothly transitions to the deterministic reaction rate equations as the inequalities become strongly satisfied. Therefore, the increment $\sum_j^M k_j \nu_j$ can be written as $\sum_j^M a_j(X(t))\boldsymbol{\tau}\nu_j$, which is an Euler formula for the reaction rate equations.

One important aspect to consider is that in continuous-time processes on complex networks, nodes change state asynchronously, meaning that the change of state or contacts of one node can immediately affect the transition rates of itself and other nodes. For instance, the dissolution of contact between a susceptible and an infected individual instantly reduces the probability of infection. Incorporating adaptive network behaviour highlights the significance of this aspect, where a single reaction can alter the behaviour of the entire system, rendering the approximation made during time step $\boldsymbol{\tau}$ invalid.

3.3.2 HRSSA

Marchetti et al. [116] has proposed a hybrid rejection-based algorithm built on top of the rejection-based SSA. The method employs a concept of propensity upper bounds and fluctuation intervals, along with the categorisation of reactions into classes of slow (R^s) and fast (R^f) reactions. This approach was also previously used by Cao [107] for the $\boldsymbol{\tau}$-leap algorithm to avoid negative values. The system partitioning is updated only when the current state exits the fluctuation interval. HRSSA calculates the sum of upper propensity bounds of slow reactions $\overline{a_0^s(X(t))}$, and then samples the candidate firing time of a slow reaction $\tau \sim Exp\left(\overline{a_0^s(X(t))}\right)$. Assuming the system state will remain within the fluctuation interval during $[t, t+\tau]$, the upper bound $\overline{a_0^s(X(t))}$ is considered time-independent over the interval, allowing for the simulation of fast reactions without side effects on slow reactions. Fast reactions are then simulated for the interval $[t, t+\tau]$ according to an approximate simulation algorithm that can be either stochastic or deterministic. Following the simulation of fast reactions, a slow reaction is chosen and validated to fire using a rejection test, following the RSSA simulation strategy. To maintain the accuracy of the simulation of slow reactions,

the feasibility of the current system state must be preserved during the simulation of fast reactions. Thus, when the system state goes beyond its bounds, the simulation is halted, and the fluctuation interval is updated.

While presenting a promising avenue for simulating spreading processes within complex systems, this algorithm is not without limitations. The original reaction partitioning algorithm introduced by the authors split reactions into two classes – slow and fast – based solely on the reaction's propensity and does not take the type of reaction (whether populational or contact) or its dependencies into consideration. This approach could result in populational reactions being inaccurately classified as fast reactions, potentially leading to their approximate simulation and overlooking the necessary adaptivity step, if such a reaction is triggering for adaptivity. Moreover, the method proposed by the authors for estimating the upper and lower bounds of propensities presumes that reaction propensities remain constant over time and are unaffected by any external processes, either stochastic or deterministic.

3.3.3 Discrete time-approximation

Another way to simulate continuous-time Markov process spreading models is to use discrete-time approximation. This approach has gained popularity, especially in modelling spreading processes involving complex networks to represent the dynamics and structure of contact networks [117–119, 74]. Its popularity is due largely to its simplicity and minimal simulation time, making it a convenient alternative to exact algorithms that are unnecessarily detailed and to $\boldsymbol{\tau}$-leap methods that require meticulous $\boldsymbol{\tau}$-selection procedures, which in some cases can be captious to the complexity of the modelled system.

In discrete-time simulation of continuous-time Markov processes, time is not treated as a continuous variable, but instead as a discrete variable advancing in time intervals of length Δt. Typically, Δt is assumed to have a value of 1. In this framework, reaction rates γ_i are replaced by reaction probabilities $\tilde{\gamma}_j = \gamma_j \Delta t$. For instance, susceptible nodes become infected through their infected neighbours with a probability of $\tilde{\gamma}_{tr} = \gamma_{tr} \Delta t$ per infected neighbour. At each time step interval $[t + \Delta t]$, reactions occur according to their corresponding probabilities.

While the discrete-time approximation is easy to implement and useful for modelling systems of varying complexities, it introduces several deviations from the continuous-time process. When the time interval Δt is very small, the probability of multiple reactions occurring can be ignored, and the probability that reaction R_j will occur during this interval is precisely $\gamma_j \Delta t$. However, as Δt grows, this expression becomes an approximation, and the exact probability for the continuous-time Markov process that reaction R_j with rate γ_j will occur is $1 - \exp(-\gamma_j \Delta t)$. Unlike in the $\boldsymbol{\tau}$-leaping methods, there is typically no control over the step size Δt in the discrete-time framework.

Another crucial consideration is the synchronous updates occurring during the time-step Δt, similar to τ-leaping. Particularly when the step size selection is uncontrolled, such updates can produce notable discrepancies between predicted outcomes and reality. These limitations have been demonstrated by Fennel et al. [120] and partially by Malysheva et al. [121], showing that this discrete-time approach can accurately replicate the continuous-time dynamics for relatively small Δt. However, it is important to note that a smaller step size, while increasing accuracy, also leads to greater computational time. As the step size increases, the accuracy significantly decreases, highlighting a key limitation of this method. These deviations impact both the range of applicable model parameters and the feasible size of Δt. A more comprehensive examination of the limitations of the discrete-time framework can be found in [120].

3.4 Simulation approaches considering dynamic environment

The impact of a dynamic environment on the spread of different phenomena can be significant and can arise from external aspects such as weather patterns, biological systems interactions, and ecological processes. However, the most significant impact on the spread of phenomena is often caused by human behaviour such as social interactions. In recent years, the incorporation of these factors into the modelling of spreading processes has gained popularity and has led to new insights into the dynamics of various phenomena. While some methods already described in this chapter allow for the inclusion of dynamic inputs into simulations, several newer methods have been developed allowing not only for the incorporation of environmental impacts but also for computational advantages such as speed.

3.4.1 Temporal Gillespie algorithm

Vestergaard's Temporal Gillespie algorithm [122] presents a modified version of the direct method to simulate jump processes occurring in switching temporal networks. These networks are characterised by abrupt changes in their structure at specific time points. For example, many real-world networks are constructed based on observed contact statistics within certain time intervals, denoted as Δ_n. The author considers networks evolving over time, independently from the processes taking place on them, excluding the possibility of adaptive behaviour.

The behaviour of switching temporal networks, particularly those constructed from empirical data, displays notable intermittency, complicating any analytical description of

spreading processes occurring on these networks. Consequently, the survival function for the waiting time τ until the next event among all the processes denoted as $\Psi(\tau,t) = \exp\left(-\int_t^{t+\tau} a_0(X(\tau'))d\tau'\right)$, where $a_0(X(t))$ is a propensity total as defined in 3.2, cannot be straightforwardly inverted to sample the waiting time to the next event, as is typically done in the direct method. To address this, the author suggests the use of unitless, normalised waiting times $\tilde{\tau} = \int_t^{t+\tau} a_0(X(\tau')),d\tau'$. This normalised waiting time follows an exponential distribution with an expected value of one. Since the temporal network undergoes discrete changes with intervals of Δ_n, $a_0(X(\tau'))$ remains constant between these changes. To determine the interval in which the next event occurs, the algorithm compares the generated value of $\tilde{\tau} \sim Exp(1)$ with the integral value $\int_{t'}^{\Delta_n} a_0(X(t)),dt$.

Algorithm 4. Temporal SSA.

$t = 0$
Compute the sum of the reaction propensities $a_0 = \sum_j a_j$
$i = 1$
while $i \leq n, t < T_F$ **do**
    Draw a normalized waiting time until the first event from a standard exponential distribution $\tilde{\tau} \sim Exp(1)$
    accepted = FALSE
    repeat
        if $\Delta_i a_0 \leq \tau$ **then**
            ▷ *no reaction happened* ◁
            $\tilde{\tau} = \tilde{\tau} - a_0\Delta_i$
            $t = t + \Delta_i$
            $i = i + 1$
            Update propensities a_j affected by changes in the temporal network and the a_0 accordingly
        else
            ▷ *some reaction happened* ◁
            Calculate reaction time $\tau' = \dfrac{\tilde{\tau}}{a_0}$
            Choose the firing reaction with probability $\dfrac{a_j}{a_0}$
            Execute chosen reaction
            $t = t + \tau'$
            accepted = TRUE
            Update the remaining length of the present time window $\Delta_i = \Delta_i - \tau'$
            Update propensities a_j affected by changes in the temporal network and the a_0 accordingly
    until accepted

The algorithm operates by iterating through the list of discrete times at which the network changes. Within each interval between network switches, it compares the normalised waiting time $\tilde{\tau}$ with the total instantaneous rate integrated over the time interval $a_0\Delta_n$. If $\tilde{\tau}$ is greater than or equal to $a_0\Delta_n$, nothing happens. After $a_0\Delta_n$ is subtracted from $\tilde{\tau}$, the algorithm proceeds to the next interval Δ_{n+1}. On the other hand, if $\tilde{\tau}$ is smaller than $a_0\Delta_n$, an event occurs within the nth time window. The algorithm then determines the timing of the event, selects the reaction channel responsible, updates the system, draws a new normalised waiting time, and repeats the procedure.

The temporal Gillespie algorithm guarantees stochastic exactness, meaning that all distributions and moments of a stochastic process evolving on a time-varying network, obtained through simulations, converge to their exact values.

The expected complexity of the algorithm is $O(\langle E(t)\rangle n) + O(\langle F(t)\rangle n)$, where n is a number of discrete steps of the network change, $\langle E(t)\rangle$ is the mean number of contacts per time-step and $\langle F(t)\rangle$ is the mean number of transitions that occur per time-step.

Djurdjevac Conrad et al. [84] applied the temporal Gillespie algorithm to explore the spread of innovation within the context of human mobility. They introduced an event-based variant of the algorithm tailored for a 2-state ABM. In this model, an agent's state is binary: either having adopted an innovation, indicated by state value 1, or not, represented by state value 0 (analogous to the SI state compartments in epidemiological models). In their study, the aspect of human mobility, which consequently shapes the contact network defined by the agents' positions, was simulated using the Euler–Maruyama scheme [123].

The authors have also proposed an extension allowing the incorporation of the adaptive behaviour, which can be executed after the relevant transition reaction is performed (after line 14 in Alg. 5).

A limitation of this method arises from the fact that the size of the discrete-time step Δ_i is determined by the network's dynamics rather than the spreading dynamics. This can be advantageous when the network dynamics are considerably slower or at least comparable to the spreading dynamics. In such cases, it is reasonable to examine every time window of network change to check whether the transition reaction has occurred. However, if the network dynamics are significantly faster than the spreading dynamics, this approach can lead to numerous network updates with very few transition reactions firing. Since the primary research focus often lies on predicting spreading patterns, this approach may not be optimal.

Algorithm 5. Temporal SSA variation.

$t = 0$
Init $\mathscr{I}(t)$ – the set of agents that have not adopted the innovation until time t.
Choose a time step Δ_i of the network updates
Draw a normalised waiting time until the first event from a standard exponential distribution $\tilde{\tau} \sim Exp(1)$
while $t < T_F$ and $\mathscr{I}(t) \neq \varnothing$ **do**
    update $\mathscr{I}(t)$
    compute adaption rates a_j for each agent and its total a_0
    if $\Delta_i a_0 \leq \tilde{\tau}$ **then**
        Position update using Euler–Maruyama scheme for the time-step Δ_i.
        $\tilde{\tau} = \tilde{\tau} - a_0 \Delta_i$
        $t = t + \Delta_i$
    else
        Select the firing reaction (agent adopting the innovation) with the probability $\frac{a_j}{a_0}$
        Execute chosen reaction by changing the state of the chosen agent to 1
        Calculate reaction time $\tau' = \frac{\tilde{\tau}}{a_0}$
        Position update using Euler-Maruyama scheme for the time-step $(t, t+\tau')$
        $t = t + \tau'$
        Draw a normalized waiting time until the first event from a standard exponential distribution $\tilde{\tau} \sim Exp(1)$

3.4.2 Extra reaction algorithm for networks in dynamic environments (Extrande)

Proposed by Voliotis et al. [124], the extra reaction algorithm for networks in dynamic environments (Extrande) allows the exact stochastic simulation of reaction-diffusion models influenced by the time-dependent external process $I(t)$. The dynamic of this external process is initially assumed to be simulated in advance. The Extrande method uses the concept of the total propensity upper bound $B(T_L)$ which is valid over a certain time interval T_L. To generate a tentative reaction time τ, the method employs this bound on total propensity, generating a value from an exponential distribution: $\tau \sim Exp(B(T_L))$. If the reaction time surpasses the time horizon T_L, it is discarded. Consequently, the time advances by T_L, and the process restarts by establishing a new bound.

Alternatively, if the reaction time falls within the time horizon, the actual propensities $a_j(t+\tau, I(t+\tau))$ and the total propensity a_0 are calculated. Based on these values, the reaction is either selected with a probability of $a_j/B(T_L)$, or "thinned", meaning that none of the reactions has fired and the state of the spreading process remains unchanged. The time advances by τ.

Algorithm 6. Extrande.

$t = 0$
Precalculate the dynamics of the background process $I(t)$ for the time interval $[0, T_F]$
while $t < T_F$ **do**
    Define *look-ahead time* $T_L \leq T_F - t$
    Define propensity upper bound $B(T_L)$ such as: $a_0(t+u) \leq B(T_L)$ for $u \in [0, T_L]$
    Generate putative reaction time $\tau \sim Exp\,(B(T_L))$
    if $\tau > T_L$ **then** ▷ *leftyleft*
        ▷ *reject* ◁
        $t = t + T_L$
    else
        $t = t + \tau$
        Obtain the value $I(t)$ from pre-simulated data, update all propensities $a_j\,(X, I(t))$ that depend on $I(t)$
        Compute $a_0 = \sum_j a_j$
        Sample $u \sim \mathscr{U}\,(0,1)$
        if $a_0 \geq B(T_L) \cdot u$ **then**
            ▷ *accept:* ◁
            Choose the next reaction with probability $\dfrac{a_j}{B(T_L)}$
        else
            ▷ *thin: network remains the same* ◁

This algorithm enables one to incorporate the influence of diverse external processes on the dynamics of spreading. The function $I(t)$ employed to simulate the inputs relies on the nature of the input processes. It could take the form of a deterministic solution of ordinary or partial differential equations. It can also be the functions that return values on a discrete grid, such as those obtained through the Euler–Maruyama method for stochastic differential equations, or the states of the empirical contact network observed at some points in time. The method, however, offers no possibility to incorporate adaptive dynamics, where the $I(t)$ can change in response to the spreading dynamics.

Unlike Vestergaard's method, the step size in this approach is determined by the spreading process itself, regulated by the look-ahead time T_L and upper bound $B(T_L)$. By selecting this bound appropriately, the frequency of "thinning" events can be minimised. Consequently, the method can focus more on simulating the spreading process itself, improving computational performance. Assuming that the upper bound $B(T_L)$ is determinable within a constant time, and considering that the process $I(t)$ is precomputed prior to the beginning of the simulation, the computational complexity of the Extrande algorithm can be approximated as $O(n_{sim}M)$, linear with regards to the total number of reactions M. However, the number of simulation steps n_{sim} depends upon the chosen value of $B(T_L)$. If $B(T_L)$ is set too high, numerous

of "thinning" events result, thereby, increasing the number of simulation steps. Thus, the selection of $B(T_L)$ demands careful consideration to optimise the efficiency and efficacy of the simulation.

3.4.3 Event-based simulation algorithm

An event-based algorithm to simulate spreading dynamics on contact networks was proposed by Holme [125]. It is based on the event-driven algorithm for SIR on static networks by Kiss, Miller, and Simon [30] and connects to the next reaction method.

This algorithm is also based on the assumption that the dynamic of the temporal contact network is known in advance and can be represented as a sequential collection of contacts (v_i, v_j, t), meaning that individuals v_i and v_j were in contact at time t. Every interaction between infected and susceptible nodes carries an equal contagion probability β. This scenario is conceptualised as a Bernoulli process applied to these contacts, subject to specific criteria. A contagion event occurs during the initial non-zero interaction following the transition of a node pair to the susceptible-infected (SI) state, provided that the nodes remain in the SI state at the time of this interaction.

The algorithm uses a priority queue Q to maintain a sorted sequence of potential infection events, prioritising earlier occurrences to enhance computational performance.

The algorithm strategically avoids searching through all contact events between individuals v_i and v_j when determining the specific event where v_i successfully infects v_j. Instead, it leverages the knowledge that the infection process between v_i and v_j can be modelled as a finite sequence of binary random variables, following a Bernoulli process with a probability of β. The number of these random variables corresponds to the number of contacts, denoted as $n_{ij}(t_i)$, between v_i and v_j for $t > t_i$. Consequently, the probability that the k-th contact transmits the disease is given by $\beta(1-\beta^{k-1})$. To determine the value of the random variable k, the algorithm samples a random number u from a uniform distribution in the range $(0,1)$ and computes k as $\left\lfloor \log(1-u)/\log(1-\beta) \right\rfloor$.

Notably, the author remarks that if there are too many contacts between two nodes, then the distribution of inter-event times gets further from exponential.

The computational complexity of the algorithm expectedly depends on the density of the network, and in the worst-case scenario when the contact network is dense, the time complexity would be $O(N^2 \log N \log C)$, where N is the number of nodes and C is the total number of contacts. The expected complexity for the sparse networks, when $C \gg N$, is estimated as $O(N \log N \log C)$.

Algorithm 7. Event-based SIR algorithm.

Initialize and store contact data
Initialize infected nodes by placing them in queue Q with an initial infection event time $t = 0$
while Q is not empty **do**
    Extract the node v_i with the smallest infection time from Q
    for all v_j – neighbours of v_i **do**
        if v_j is Susceptible **then**
            ▷ *get the time τ_j when it would be infected by v_i* ◁
            From the contact list, find the index k of the contact between v_i and v_j with the smallest timestamp, such that $t_i < t_{i,j}$
            sample $u \sim \mathscr{U}(0,1)$ and calculate $k' = k + \left\lfloor \frac{\log(1-u)}{\log(1-\beta)} \right\rfloor$
            if k' exceeds the number of contacts between v_i and v_j **then**
                No contact will spread the disease.
            else
                The k'ths contact between v_i and v_j that could be contagious. Set $\tau_j = t_{i,j}(k')$
            if There is no earlier infection event of v_j on the Q **and** i's recovery time is not earlier than τ_j **then**
                put the contagion of v_j by v_i in the Q

3.5 Summary

This chapter synopsises contemporary stochastic simulation methods and their capacities for simulating time-varying adaptive network behaviour. It categorises these methods into exact algorithms, approximation algorithms, and a subgroup dedicated to incorporating the effects of external processes, like contact dynamics in populations. While exact methods facilitate the integration of behavioural responses and adaptivity without compromising accuracy, approximation methods struggle with this integration, often introducing additional biases. Moreover, methods that explicitly account for external processes typically presuppose known or pre-simulated behaviour, constraining the integration of adaptive simulations. Despite advancements in the modelling of dynamic environments, significant challenges persist in the accurately simulation of spreading processes on time-varying adaptive networks. Therefore, ongoing research and development are crucial to enhance the range and efficacy of methods that can not only capture and simulate adaptive network behaviour, but also maintain time efficiency.

Chapter 4

Designing a hybrid algorithm to simulate spreading on temporal adaptive networks

4.1 Introduction

In the context of dynamic networks, the simulation of phenomena such as infectious disease spread or information flow is challenging due to the dynamic nature of networks and the complexity of the spreading processes. Traditional simulation methods frequently face challenges in effectively capturing the dynamic interplay between spreading phenomena and the evolving structure of the network, often resulting in significant computational overhead. This chapter presents the ***s**tochastic **s**imulation **a**lgorithm for effective spreading dynamics simulation on **t**ime-evolving **a**daptive **n**etwor**x** (SSATAN-X)*, a hybrid algorithm developed to address these challenges in simulating spreading processes on temporal adaptive networks. SSATAN-X combines innovative techniques to accurately and efficiently model these dynamics.

The design of the SSATAN-X algorithm combines the precision of detailed spreading simulations with the efficiency of more general approaches in simulating contact dynamics. This balance allows SSATAN-X to offer significant computational advantages, allowing it to handle the complexities of temporal adaptive network models without compromising accuracy or speed.

The chapter provides insights into the core principles of the SSATAN-X algorithm. It discusses the algorithm's capabilities in simulating spreading and contact dynamics, highlighting the balance between precision and computational load. The chapter comprehensively explains the algorithm's methodology, complexity, and computational advantages.

4.2 The idea

This study focusses on a class of models describing the propagation of a phenomenon within a population, where the spread occurs stochastically through relevant interactions. Moreover, the dynamics of these interactions are also subject to stochastic behaviour.

Consider a population denoted as $\mathfrak{P}$, which is partitioned into distinct compartments represented by $\mathfrak{P} = \bigcup_i \mathfrak{P}_i$. Within this population, individuals are interconnected through a network of contacts. As elaborated in chapter 2, the precise definition of a "contact" can vary depending on the specific model and its intended purpose. For instance, in the context of STDs, contact may refer to sexual interactions, while in the case of airborne diseases, it may be associated with the spatial proximity between individuals. Importantly, the interconnections among individuals within the population exhibit temporal variability, as pairs of individuals can initiate or terminate contacts at different points in time. The characteristics and attributes of individuals (e.g., age, sex, social status, profession, and other individual-specific traits) can influence various aspects of contacts and spreading dynamics within the model. This influence encompasses the quantity, duration, and intensity of contacts experienced by each individual, as well as their susceptibility to and contagiousness of the phenomenon under study. Additionally, the population under investigation may be subject to vital dynamics, wherein new individuals are introduced (through births or migration from external sources) or removed (through deaths or departures) from the population.

Within the population $\mathfrak{P}$, individuals can transition between different compartments $\mathfrak{P}_i$. Notably, certain transitions depend upon the presence of specific contacts. For instance, in the context of pathogen transmission, for an infected individual to transmit the pathogen to a susceptible individual, thus transitioning an individual from the susceptible compartment to the infected compartment, direct contact between the infected and susceptible individuals is required.

Moreover, the spreading process can be influenced by various interventions that modify the dynamics of contact. Additionally, individual behavioural changes may occur as a result of transitioning to a particular compartment. For instance, in the context of disease transmission, an individual who becomes aware of their infectious status or receives a diagnosis may temporarily reduce or halt their contact activities to restrain further transmission.

This generalised formulation of the model provides a framework by which to understand the spreading process with adaptive dynamics. In this framework, the dynamics of contacts exhibit adaptability, wherein changes occur discontinuously, depending on the epidemic state. This adaptive behaviour allows for a more realistic representation of the spreading process, accounting for the dynamic nature of contacts and their dependency on the current state

of the epidemic. It also allows for the incorporation of the impact of possible containment measures.

The above-described model can be mathematically formulated as follows: population $\mathfrak{P}$ can be represented as a weighted temporal contact network, denoted as $G(t) = \{V(t), E(t)\}$. Here, $V(t)$ represents the set of nodes $\{v_1(t), v_2(t), ..., v_N(t)\}$ at a given time t. Each node $v_k \in V$ corresponds to an individual $\mathfrak{p}_k$ from the population $\mathfrak{P}$ and is assigned the set of individual characteristics essential for the model (e.g., age, profession, and sex).

The set of edges is denoted as $E(t) = \{e_{jk}(t)\}$, where $e_{jk} = (v_j, v_k; \phi)$ for $j \neq k$ if individuals v_j and v_k are in contact. The edges of the network can be either directed or undirected. For simplicity, the undirected variant is considered in subsequent analysis, although the transformation to a directed or mixed graph with both directed and undirected edges can be easily implemented. The graph $G(t)$ is a simple graph, meaning it neither possesses multiple edges connecting the same pair of nodes nor contains loops. The weight $\phi = \{\phi_1, ...\phi_s\}$ represents a set of parameters specific to this interaction. For instance, it can include the rate of virus transmission between individuals v_j and v_k. In addition, each existing edge in the network is assigned a disassembly rate, representing the probability of the edge being removed. Conversely, each possible, but not yet existing edge is assigned an establishment rate, representing the probability of a new edge being formed.

The evolution of the complex system $G(t)$ is governed by a continuous-time Markov process denoted as X_t. The events in this process can be categorised into two main groups:

1. *Contact dynamics* $R^c = \{R^+, R^-\}$

   - $R^+ = \{R^+_{jk}, \lambda^+_{jk}\}$ – assembly of a new contact. For each pair of nodes (v_j, v_k), where $j \neq k$ and there is no existing edge, an edge can be formed with the rate λ^+_{jk}.
   - $R^- = \{R^-_{jk}, \lambda^-_{jk}\}$ – disassembly of an existing contact. Each edge connecting a pair of nodes (v_j, v_k), where $j \neq k$, can be disassembled with the rate λ^-_{jk}.

2. *Population dynamics* R^s includes, but is not limited to the following types of reactions:

   - $R^{\mathfrak{P}_i \to \mathfrak{P}_j} = \{R_k^{\mathfrak{P}_i \to \mathfrak{P}_j}, \gamma_k^{\mathfrak{P}_i \to \mathfrak{P}_j}\}$ – transition of the individual node v_k from compartment $\mathfrak{P}_i$ to the compartment $\mathfrak{P}_j$ happens with the corresponding rate $\gamma_k^{\mathfrak{P}_i \to \mathfrak{P}_j}$.
   - $R^{\varnothing \to \mathfrak{P}_i} = \{R_k^{\varnothing \to \mathfrak{P}_i}, \beta^{\mathfrak{P}_i}\}$ – birth or introduction of a new individual in the compartments $\mathfrak{P}_i$ happens with the corresponding rate $\beta^{\mathfrak{P}_i}$.
   - $R^{\mathfrak{P}_i \to \varnothing} = \{R_k^{\mathfrak{P}_i \to \varnothing}, \rho^{\mathfrak{P}_i}\}$ – death or departure of an individual from the compartments $\mathfrak{P}_i$ happens with the corresponding rate $\rho^{\mathfrak{P}_i}$.

3. *Intervention measures and adaptive behaviour change*. Although not typically stochastic reactions themselves, but are integral to population dynamics due to their close association with these processes.

The events within the group R^s can be either contact-dependent or contact-independent. If the event is contact-dependent, its occurrence rates may vary based on the current state of the contact network $G(t)$. For instance, in SI or SIR state space models, the transition of a node v_k from the susceptible (S) to the infectious (I) compartment is directly influenced by its contagious contacts. An example of a contact-independent event is the death of a node v_k, not influenced by the quantity or quality of its contacts at the time of death, but rather depending on its individual status.

Given the significant heterogeneity within the population, it is logical to approach these as agent-based models. In this approach, every individual in the population is treated as a distinct agent, and state change of each agent is treated as an individual reaction. Additionally, each interaction between agents is considered an individual reaction, acknowledging the individuality and uniqueness of agent behaviours and interactions within the system. This ABM framework allows for a more detailed and realistic representation of the complex dynamics arising from the heterogeneity present in the population.

The fundamental concept of the proposed algorithm is to divide the stochastic process X_t into two distinct processes: Y_t and Z_t. The process Y_t focusses on contact dynamics, encompassing all reactions R^c as previously described, concentrating on events involving the formation of new contacts and the dissolution of existing ones. On the other hand, the process Z_t governs reactions R^s related to population dynamics, as well as the adaptive changes triggered by this reactions. It captures the dynamics associated with the spread of the epidemic, as well as other relevant factors such as vital dynamics, intervention measures, and adaptive behavioural changes.

The algorithm then introduces a putative reaction time Δt for the first reaction of the process Z_t, which acts as a reference point. This reference point allows for the independent simulation of the contact dynamics process Y_t, either exactly or approximately, up to this predetermined time Δt, without needing to account for any potential impact on the process Z_t. This process distinction allows for detailed study of how the phenomenon spreads within the population, without the need for precise control of the temporal contact dynamics captured by Y_t.

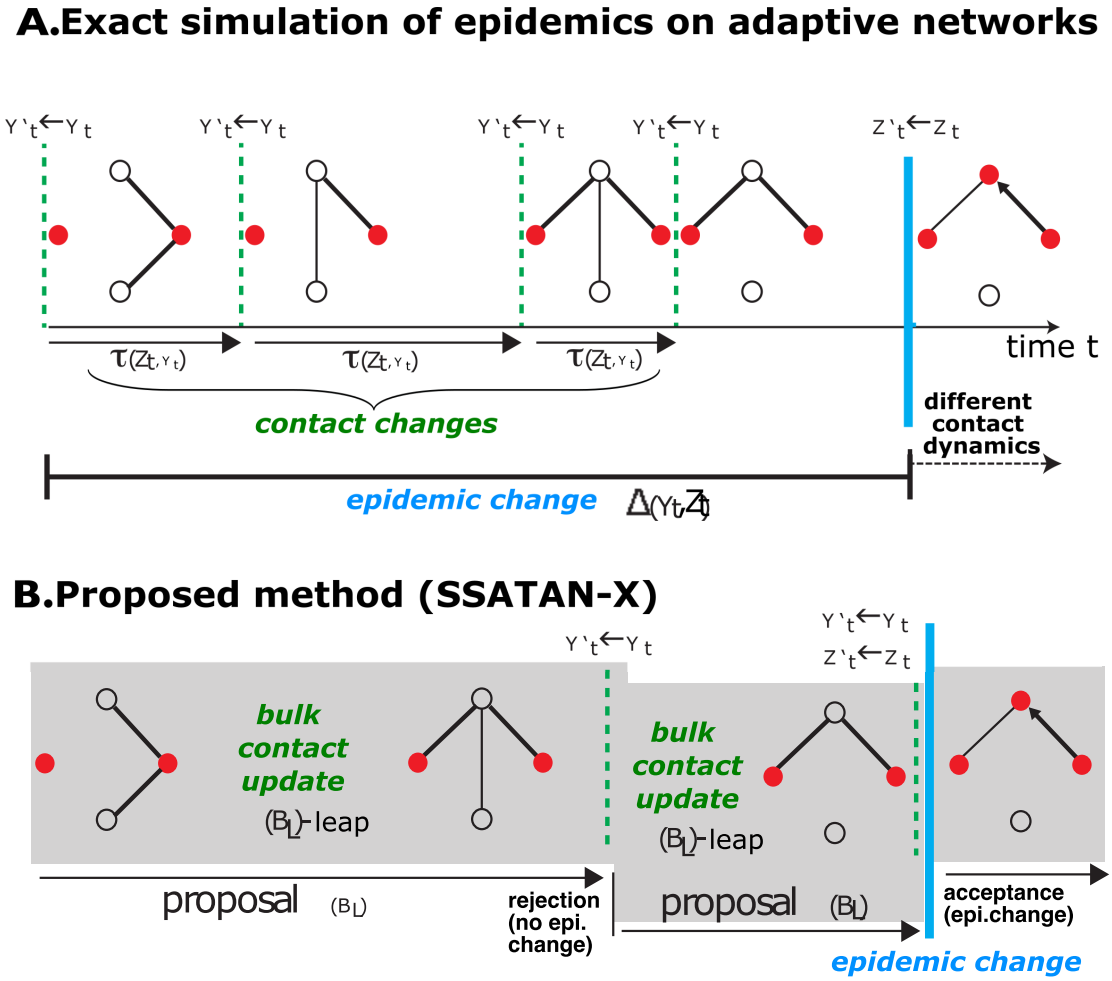


Fig. 4.1. Methods for simulating spreading processes on networks. Two processes $Y|Z : E \to E$ and $Z|Y : V \to V$ are considered. The process $Y|Z$ affects the edges E of the network. It governs the topology and *contact dynamics* within the network. The second process $Z|Y$ rules the V (*population and spreading dynamics*), focussing on how the state (e.g., infected, susceptible) of each node evolves based on the network's topology and other nodes' states. **(A)** Exact stochastic simulation [126] of spreading processes on adaptive networks. In this approach, every change of the network, either topological or populational, is explicitly modelled. This includes every contact alternation, and the state of the nodes (empty circles vs. filled red circles). **(B)** Hybrid approach of SSATAN-X: The proposed method is more focussed on the *overall* effect of topological changes on *spreading dynamics*. It prioritises understanding how changes in network connections influence the spread of an epidemic. An upper bound B_L is computed such that $B_L \geq \sum_{j \in Z} a_j(V,E)$. Then, time-step $\Delta(B_L)$ of the first possible reaction is chosen. This choice reassures that no contact change leads to an earlier putative time than $\Delta(B_L)$. The network topology for the time step $\Delta(B_L) \sim \exp(B_L)$ is either mass-updated using the approximate method or explicitly updated with methods such as NRM [103]. A change the node state is conducted if $u \leq a_{0,Z}/B_L$; $u \sim \mathscr{U}(0,1)$ and $a_{0,Z} = \sum_{j \in Z} a_j(V,E)$.

4.3 Exact simulation of spreading process: Extrande approach

Introduced by Voliotis et al. [124], the Extrande algorithm, was initially proposed for simulating well-stirred chemical systems, where interactions could occur between any pair of chemical particles – agents. This work extends the algorithmic concepts to dynamic contact networks, which more accurately represent the interaction patterns among agents over time. Unlike the well-stirred assumption of universal interaction potential, dynamic contact networks introduce a realistic element of sparsity and variability, determining which agents are in contact at specific instances.

The adaptation of the algorithm for dynamic, adaptive contact networks enables the simulation of systems with changing interaction patterns, facilitating a detailed exploration of complex behaviours emerging from these evolving dynamics. Moreover, a proposed concept combines the principles of the Extrande algorithm with mechanisms for efficient and independent updating of the contact network topology. This update can be performed using approximate methods, capable of handling multiple contacts simultaneously, significantly reducing the computational burden associated with individual contact updates in large systems. Alternatively, the contact structure can be updated using exact stochastic simulation algorithms, such as the next reaction method. In this case, until the proposed putative time Δt, contact updates inflict no impact on the process Z_t, allowing one to minimise the number of propensity updates in the priority queue Q impacted by the contact changes.

By merging the Extrande approach with contact dynamics updating techniques, this method allows efficiency and accuracy to be balanced in simulating spreading processes. This hybrid approach opens new possibilities for researchers to simulate and analyse complex systems with evolving contact networks in a computationally manageable way.

4.3.1 Algorithm

The pseudocode of the enveloping SSATAN-X algorithm is presented in the Algorithm 8. In this outline, **line 3** defines the look-ahead time T_L. As suggested in [124], a safe choice for the look-ahead time horizon is $T_L = T_F - t$, where T_F represents the final time of the simulation. However, researchers can choose a tighter estimation of the look-ahead time if necessary for specific modelling needs.

For the time span $[t, t+T_L]$, the upper limit $B(T_L)$ for the total reaction propensities associated with the process Z_t encompassing both epidemic and vital dynamics, is calculated in **line 4**. In other words, $B(T_L)$ is selected such that $a_0^s(t+h) \leq B(T_L)$ for any $h \in [0, T_L]$.

Algorithm 8. SSATAN-X envelope algorithm.

Input T_F, $G(0)$

1: $t = 0$
2: **while** $t < T_F$ **do**
3: Define *look-ahead time* $T_L = T_F - t$
4: Define propensity upper bound $B(T_L)$, as detailed in Section 4.3.2 :
5: Sample $\Delta t \sim Exp(B(T_L))$
6: **if** $\Delta t > T_L$ **then**
7: ▷ *reject* ◁
8: $t = t + T_L$
9: Update the edges of $G(t)$ for time interval $[t, t+T_L]$ if necessary.
10: **else**
11: Update the contacts of $G(t)$ for time interval $[t, t+\Delta t]$ using exact or approximate algorithms.
12: $t = t + \Delta t$
13: Compute reaction propensities a_j^s of the reactions R^s, based on the current network $G(t)$;
14: Compute $a_0^s = \sum_j a_j^s$
15: Sample $u \sim \mathscr{U}(0,1)$
16: **if** $a_0^s \geq B(T_L) \cdot u$ **then**
17: ▷ *accept:* ◁
18: Choose the ℓ-*th* reaction with probability $\dfrac{a_\ell}{B(T_L)}$
19: Execute chosen reaction
20: Execute adaptivity step if required
21: **else**
22: ▷ *thin: network remains the same* ◁

Here a_0^s denotes a total of the reaction propensities within the group R^s. As the parameter $B(T_L)$ provides an upper bound for the sum of all reaction propensities of the reactions from the subgroup R^s within the specified time interval, this upper bound is specific to the particular model under consideration. A large upper bound value may lead to an increased number of "thinning" events (**line 22**), potentially impacting the efficiency of the algorithm. Maintaining a balance and selecting an appropriate upper bound is crucial to optimise the algorithm's performance and computational efficiency. Section 4.3.2 discusses various approaches, each with differing levels of accuracy, for selecting an upper bound.

The constant $B(T_L)$ is then used to propose the time to the next reaction, denoted as Δt in **line 5** of the algorithm. Due to $B(T_L)$ being an upper bound, the proposed time to the next reaction event might be shorter than the actual time to the next reaction. This apparent underestimation is addressed through the subsequent "thinning" step, as outlined in **lines**

21–22 of the algorithm. This step ensures that the statistics of the process are preserved. For more detailed information, please refer to Voliotis et al. [124].

If the proposed time step Δt falls within the look-ahead horizon T_L, the simulation time t is updated by Δt in **line 12**. At this stage, it is necessary to update the actual propensity functions a_j^s and their total a_0^s. As some or all of these reactions depend on the contacts between individuals, it is necessary also to update the actual contact state. For this purpose, the contact dynamics (edges) of the network $G(t)$ are updated in **line 11** using approaches further discussed in sections 4.4 and 4.5. Additionally, the next reaction event, which updates or "thins" the state of the nodes in the epidemic process, is selected in **lines 15–22**. The adaptivity step in **line 20** accounts for any significant changes to the contact network that may occur as a result of a change in population dynamics. For example, change in the epidemic state of the particular individual may lead to them "isolating" (i.e., cutting all existing contacts immediately).

4.3.2 Estimation of the upper bound

The parameter $B(T_L)$ establishes an upper limit on the sum of reaction propensities a_0^s of the reactions in R^s within the time interval $[t, t+T_L]$. Using the terminology of section 4.2, this upper limit can be represented as $B(T_L) = \sum_{R \in R^s} B_R(T_L) = B_{\mathfrak{P}_i \to \mathfrak{P}_j}(T_L) + B_{\varnothing \to \mathfrak{P}_i}(T_L) + B_{\mathfrak{P}_i \to \varnothing}(T_L) + \ldots$. In simpler terms, the upper bound for the total reaction propensities of all reactions in R^s is the sum of the upper bounds for the reaction propensities of each reaction type (transition, death, birth, etc.).

Certain reaction types are independent of the interactions between individuals, meaning they do not rely on the current state of the contact network at any point in time. For these types, the propensities upper bound can be straightforwardly calculated, as it is equal to the propensities total of all individual reactions of that type: $a_0^{R^s} = \sum_{\mathfrak{p} \in \mathfrak{P}} a_{\mathfrak{p}}^{R^s}(t)$.

In contact-dependent reactions, accurate estimation of the upper bound of propensities relies heavily on the determination of the maximum possible number of edges or contacts $E^{max}(T_L)$ that could potentially lead to the occurrence of such reactions within the look-ahead interval $[t, t+T_L]$. To illustrate this concept, let us examine the S-I (i.e., susceptible-infected) transmission reaction represented as $S_i + I_j \to I_i + I_j$. On an individual or agent level, this representation indicates a susceptible individual (S_i) became infected (I_i) due to the interaction with an infected individual (I_j). Therefore, to accurately estimate the upper bound for the total of all the S-I transmission reactions, it is necessary to estimate the maximum number of edges or contacts between susceptible and infected individuals that could occur during the duration of T_L.

Several approaches for estimating the quantity $E^{max}(T_L)$ are described below. For the purpose of simplicity, these approaches are explained using the above example of pathogen transmission $S_i + I_j \rightarrow I_i + I_j$ where an infected individual (I_j) transmits the pathogen to a susceptible individual (S_i).

Naive approaches. Though simplistic, one initial approach rests on the understanding that the maximum number of contacts $C_j^{\max}$ an individual (node) v_j can have at any given time-point t is constrained by the maximum size of the population, N, and is calculated as $N-1$. Nevertheless, the actual limitation on contacts varies depending on the specific phenomenon being modelled. For instance, Kretzschmar et al. [35] has demonstrated that the number of sexual partnership contacts an individual has is considerably lower than the overall population size. Knowing this, a rough estimation of $E^{max}(T_L)$ can be achieved by summing the maximum number of contacts for each infected individual: $E^{max}(T_L) = \sum_{v_k \in I} C_k^{\max}$, where the summation is performed over the subset of infected individuals, denoted by I.

However, this estimation overlooks the size of the susceptible population, which can provide valuable information for refining the upper bound estimation. The overall population size imposes constraints on the number of susceptible individuals and, consequently, the number of potential transmission edges in the network. Thus, an improvement can be made by considering $E^{max}(T_L) = \min\left(\sum_{v_k \in I} C_k^{\max}, \sum_{v_k \in S} C_k^{\max}\right)$. This approach considers the minimum value between the total possible number of edges originating from infected individuals and the total possible number of edges directed toward susceptible individuals. By considering both population compartments, a more accurate estimation of the upper bound for potential contacts between susceptible and infected individuals during the look-ahead time T_L can be obtained.

Another approach to estimating the upper bound of potential contacts is to calculate it as $E^{max}(T_L) = N_I \cdot N_S$, where N_I represents the number of infected individuals and N_S represents the number of susceptible individuals. This approach calculates the maximum possible number of contacts between infected and susceptible populations. Notably, this estimation may provide acceptable results for models where $C_k^{\max}$ is comparable to the population size N. However, for scenarios where $C_k^{\max} \ll N$, this approach tends to significantly overestimate the maximum number of potential contacts $E^{max}(T_L)$, yielding the upper bound of the reaction rate $B(T_L)$. This means $B(T_L) \gg a_0$, resulting in numerous "thinning" events (referring to line 22 in **Algorithm 8**) during the algorithm's execution. Consequently, the efficiency of the algorithm can be compromised.

Tighter estimates of $B(T_L)$. This work introduces more precise estimations for $E^{max}(T_L)$ and consequently for $B(T_L)$. When considering each node independently, the individual

contact dynamics of a node – edge gaining and losing – can be described as follows:

$$\varnothing \xrightarrow{\hat{\lambda}_j^+(C_j^{\max}-C_j)} C_j, \quad C_j \xrightarrow{\hat{\lambda}_j^-} \varnothing, \quad j = 1 \dots N,$$

where C_j denotes the existing contacts of the node v_j, rates $\hat{\lambda}_j^+ = \langle \lambda_{ij}^+ \rangle$, for all v_i not connected to v_j and $\hat{\lambda}_j^- = \langle \lambda_{ij}^- \rangle$ for all v_i connected to v_j denote the average rates of gaining and losing an edge respectively for the node v_j.

The evolution equation for the expected number of edges of the node v_j is then given as such:

$$\frac{\mathrm{d}\mathbb{E}[C_j(t)]}{\mathrm{d}t} = \hat{\lambda}_j^+(C_j^{\max} - \mathbb{E}[C_j(t)]) - \hat{\lambda}_j^-\mathbb{E}[C_j(t)], \tag{4.1}$$

with initial condition $\mathbb{E}[C_j(t_0)] = C_j(t_0)$.
For readability, equation (4.1) can be re-written as follows:

$$\frac{\mathrm{d}\mathbb{E}[C_j(t)]}{\mathrm{d}t} = \hat{\lambda}_j^+ C_j^{\max} - (\hat{\lambda}_j^+ + \hat{\lambda}_j^-)\mathbb{E}[C_j(t)]. \tag{4.2}$$

This linear ODE, describing the evolution of the expected number of edges of the node v_j, can be solved analytically for the time interval $t \in [t_0, t_0 + T_L]$:

$$\mathbb{E}[C_j(t)] = \frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - \left(\frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - C_j(t_0) \right) \cdot \exp\left(-(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t \right) \tag{4.3}$$

However, to predict the upper bound of the number of contacts, it is crucial to consider fluctuations around this expectation. Given that the standard deviation is the square root of the variance of the number of edges of the node v_j, $\pm\sqrt{\mathbb{V}[C_j(t)]}$, it becomes necessary to also establish a time-evolution equation for the variance $\mathbb{V}[C_j(t)]$. The variance $\mathbb{V}[C_j(t)]$ can be described with the following equation:

$$\mathbb{V}[C_j(t)] = \mathbb{E}[C_j^2(t)] - \mathbb{E}^2[C_j(t)]. \tag{4.4}$$

The time-evolution equation of the expectation $\mathbb{E}[C_j(t)]$ was derived in (4.3). However, the $\mathbb{E}[C_j^2(t)]$ remains unknown. As presented in the [12], the equation for the variance can also be represented using the equation of the second moment $V[C_j(t)]$. Using the knowledge that second moment can be represented with the equation

$$V[C_j(t)] = \mathbb{E}[C_j^2(t)] - \mathbb{E}[C_j(t)], \tag{4.5}$$

equation (4.4) of the variance $\mathbb{V}[C_j(t)]$ can be rewritten as follows:

$$\mathbb{V}[C_j(t)] = V[C_j(t)] + \mathbb{E}[C_j(t)] - \mathbb{E}^2[C_j(t)]. \tag{4.6}$$

ODE for the second moment $V[C_j(t)]$, according to [12], takes the following form:

$$\frac{\mathrm{d}V[C_j(t)]}{\mathrm{d}t} = -2(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)V[C_j(t)] + 2\hat{\lambda}_j^+ C_j^{\max}\mathbb{E}[C_j(t)]. \tag{4.7}$$

Upon substituting equation (4.3) for the expectation $\mathbb{E}[C_j(t)]$ into equation (4.7), the following equation is derived:

$$\begin{aligned}\frac{\mathrm{d}V[C_j(t)]}{\mathrm{d}t} =& -2(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)V[C_j(t)] + \\ &+ 2\hat{\lambda}_j^+ C_j^{\max}\left(\frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - \left(\frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - C_j(t_0)\right)\cdot\exp\left(-(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right)\right),\end{aligned}$$

which can be re-ordered:

$$\begin{aligned}\frac{\mathrm{d}V[C_j(t)]}{\mathrm{d}t} =& -2(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)V[C_j(t)] + \frac{2(\hat{\lambda}_j^+)^2(C_j^{\max})^2}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - \\ &- 2\hat{\lambda}_j^+ C_j^{\max}\left(\frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - C_j(t_0)\right)\cdot\exp\left(-(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right).\end{aligned} \tag{4.8}$$

Next, the initial condition $V[C_j(t_0)]$ must be formulated. It is known that $\mathbb{V}[C_j(t_0)] = 0$ and $\mathbb{E}[C_j(t_0)] = C_j(t_0)$. Substituting these values into equation (4.6), it can be derived that $V[C_j(t_0)] = C_j^2(t_0) - C_j(t_0)$. Given these initial condition, equation (4.8) can be solved analytically for the time interval $t \in [t_0, t_0 + T_L]$:

$$\begin{aligned}V[C_j(t)] =& \frac{(\hat{\lambda}_j^+ C_j^{\max})^2}{(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)^2} - \frac{2\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-}\left(\frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - C_j(t_0)\right)\cdot\exp\left(-(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right) + \\ &+ \left(C_j^2(t_0) - C_j(t_0) + \frac{(\hat{\lambda}_j^+ C_j^{\max})^2}{(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)^2} - \frac{2\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-}C_j(t_0)\right)\cdot\exp\left(-2(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right).\end{aligned} \tag{4.9}$$

Resubstituting equations (4.3) and (4.9) into equation (4.6), the following equation for the time-evolution of the variance $\mathbb{V}[C_j(t)]$ of the expected number of edges of the node v_j can

be derived:

$$\begin{aligned}\mathbb{V}[C_j(t)] = &\frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - \left(\frac{\hat{\lambda}_j^+ C_j^{\max}}{\hat{\lambda}_j^+ + \hat{\lambda}_j^-} - C_j(t_0)\right) \cdot \exp\left(-(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right) - \\ &- C_j(t_0) \cdot \exp\left(-2(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right).\end{aligned} \tag{4.10}$$

Comparing this equation with equation (4.3), it can be seen that

$$\mathbb{V}[C_j(t)] = \mathbb{E}[C_j(t)] - C_j(t_0) \cdot \exp\left(-2(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right). \tag{4.11}$$

In this example, the assumption is made that the edge-assembling and -disassembling process for each node is a linear Poisson process, similar to the birth-death process. Consequently, the number of edges for a single node v_j is governed by a Poisson distribution $\mathscr{P}(\alpha)$[127] with $\alpha = \lambda_j^+ C_j^{\max}/(\lambda_j^+ + \hat{\lambda^-}_j)$. From Chebyshev's Inequality [128], it can be derived that for non-normally distributed variables at least 88.8% of values of the $C_j(t)$ – the number of contacts of node v_j at a given timepoint t – will fall within the interval $\mathbb{E}[C_j(t)] \pm 3\sqrt{\mathbb{V}[C_j(t)]}$. However, the Poisson distribution $\mathscr{P}(\alpha)$ is primarily unimodal, albeit certain parameter values lead to two adjacent values having equal probability. In such cases, the Vysochanskij–Petunin inequality, which extends Chebyshev's Inequality to unimodal distributions, can be more appropriate [129]. Applying this inequality suggests that the likelihood of values $C_j(t)$ falling within the interval $\mathbb{E}[C_j(t)] \pm 3\sqrt{\mathbb{V}[C_j(t)]}$ is at least 95%.

Because the main focus is on finding the upper bound for the number of contacts $C_j(t)$, the interval of interest can be described as $\left[0, \mathbb{E}[C_j(t)] + 3\sqrt{\mathbb{V}[C_j(t)]}\right]$. The observations made during this study (Fig. 4.2) allow for the conclusion that at least 95% of values $C_j(t)$ are falling within an even tighter interval of $\left[0, \mathbb{E}[C_j(t)] + 2\sqrt{\mathbb{V}[C_j(t)]}\right]$. To find the exact probability of a value $C_j(t)$ from a Poisson distribution with mean α falls within a given interval, the cumulative distribution function (CDF) is used:

$$Pr\left(C_j(t) \leq \mathbb{E}[C_j(t)] + 2\sqrt{\mathbb{V}[C_j(t)]}\right) = \exp(-\alpha) \sum_{k=0}^{\mathbb{E}[C_j(t)]+2\sqrt{\mathbb{V}[C_j(t)]}} \frac{\alpha^k}{k!}. \tag{4.12}$$

As t increases, $\mathbb{V}[C_j(t)] \to \mathbb{E}[C_j(t)]$ and $\mathbb{E}[C_j(t)] \to \alpha$, and (4.12) takes form:

$$Pr\left(C_j \leq \alpha + 2\sqrt{\alpha}\right) = \exp(-\alpha) \sum_{k=0}^{\alpha+2\sqrt{\alpha}} \frac{\alpha^k}{k!}.$$

For smaller values of α the Poisson distribution $\mathscr{P}(\alpha)$ is notably skewed towards the left, concentrating around 0. This makes the considered interval more likely to include these high-probability values. For larger values of α (typically $\alpha > 20$), the Poisson distribution $\mathscr{P}(\alpha)$ can be approximated with a normal distribution $\mathscr{N}(\alpha,\alpha)$. This approximation allows the application of the "two sigma rule" suggesting that about 95% of the values fall within $\mathbb{E}[C_j(t)] \pm 2\sqrt{\mathbb{V}[C_j(t)]}$. Since the focus is on the upper bound, considering $\left\lceil \mathbb{E}[C_j(t)] + 2\sqrt{\mathbb{V}[C_j(t)]} \right\rceil$, it can be expected to cover $> 95\%$ of values by looking at the approximate 97.5^{th} percentile in the time interval $t \in [t_0, t_0 + T_L]$. The time evolution of the upper border of this approximate percentile is,

$$\begin{aligned} F_j(t) =& \mathbb{E}[C_j(t)] + 2\sqrt{\mathbb{V}[C_j(t)]} = \\ =& \mathbb{E}[C_j(t)] + 2\sqrt{\mathbb{E}[C_j(t)] - C_j(t_0) \cdot \exp\left(-2(\hat{\lambda}_j^+ + \hat{\lambda}_j^-)t\right)}, \end{aligned} \tag{4.13}$$

where $\mathbb{E}[C_j(t)]$ is given by equation (4.3).

Knowing that, the maximum number of contacts C_j^* for a particular node v_j during look-ahead time can be estimated:

$$C_j^* = \max_{t \in [t_0, t_0 + T_L]} \left(\left\lceil F_j(t) \right\rceil \right). \tag{4.14}$$

$F_j(t)$ has one point of extremum:

$$t^* = \frac{\ln\left(\dfrac{(B_j K_j - A_j)\sqrt{4B_j^2 K_j + (A_j - B_j K_j)^2} + |A_j^2 - B_j^2 K_j^2|}{2B_j K_j \sqrt{4B_j^2 K_j + (A_j - B_j K_j)^2}} \right)}{-B_j}, \tag{4.15}$$

where $A_j = \hat{\lambda}_j^+ C_j^{\max}$, $B_j = \hat{\lambda}_j^+ + \hat{\lambda}_j^-$ and $K_j = C_j(t_0)$.
This point t^* exists if $B, K > 0$, $A \geq 0$ and $BK > A$; therefore,

$$C_j^* = \max\left(\left\lceil F_j(t_0) \right\rceil, \left\lceil F_j(t^*) \right\rceil, \left\lceil F(T_L) \right\rceil \right). \tag{4.16}$$

Alternatively, if $F(t)$ is assumed to be monotonic:

$$C_j^* \approx \max\left(\left\lceil F_j(t_0) \right\rceil, \left\lceil F(T_L) \right\rceil \right). \tag{4.17}$$

While this above-stated formula can already be used to estimate the upper bound of the number of contacts, it is logical also to consider the limitation given by the maximum number

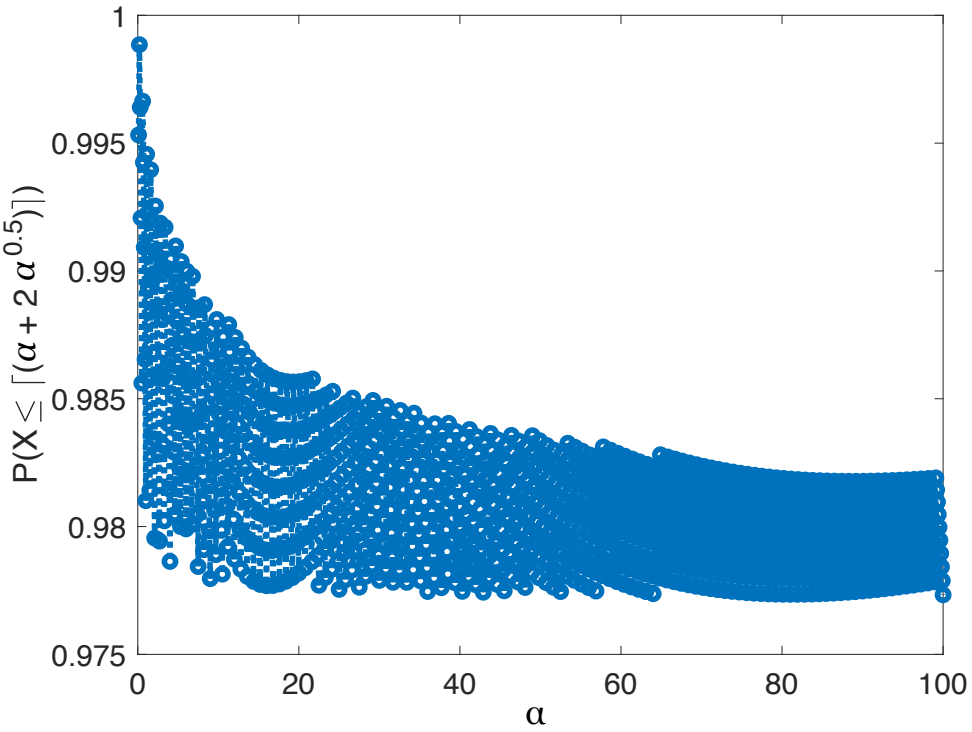


Fig. 4.2. Proportion of random variables smaller than the expected value plus two standard deviations in a Poisson distribution. The graphic depicts for random variables X, drawn from a Poisson distribution with expectation values α, the percentile of values that the random variable is smaller than $\left\lceil \alpha + 2 \cdot \sqrt{\alpha} \right\rceil$.

of contacts for each node:

$$C_j^* = \min\left(C_j^*, C_j^{\max}\right) = \min\left(\max\left(\left\lceil F_j(t_0)\right\rceil, \left\lceil F_j(t^*)\right\rceil, \left\lceil F(T_L)\right\rceil\right), C_j^{\max}\right). \tag{4.18}$$

The estimation of the upper bound of the contact-dependent reaction $S_i + I_j \to I_i + I_j$ can be calculated as follows:

$$B_{S\to I}(T_L) = \min\left(\sum_{v_k \in I} C_k^*, \sum_{v_k \in S} C_k^*\right) \cdot \max(\gamma), \tag{4.19}$$

where S denotes the susceptible fraction of the population I denotes the infected fraction and $\max(\gamma)$ – maximum possible transmission rate.

4.3.3 Proof of correctness

At a particular time point, given the present system state, it is possible to calculate the probability distribution for the firing time of a specific reaction from R^s, which results in a change in the population state. This section, partially adapted from [121], aims to demonstrate that the distribution of the firing time of such a reaction, as generated by the SSATAN-X envelope algorithm, is identical to the authentic probability distribution generated by the system. It is assumed the background contact dynamics are simulated with the required level of precision.

Proposition 5. *The SSATAN-X envelope algorithm (Algorithm 8) is an exact simulation algorithm of the spreading process Y_t with reactions R^s if a constant $B(T_L)$ exists, such that $a_0^s(t) \leq B(T_L)$ for any $t \in [t_0, t_0 + T_L]$.*

Proof. Without loss of generality, the reaction R_1^s with propensity a_1^s is selected as the target reaction for this analysis. Denote the CDF of the putative time τ to the next firing of reaction R_1^s given the state of the network $G(t_0)$ as $F_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$. Using the property of the CDF and the definition of the derivative, the probability density function (PDF) $f_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$ can be represented:

$$f_{R_1^s}\big(\tau \mid G(t_0), t_0\big) = \frac{\partial F_{R_1^s}\big(\tau \mid G(t_0), T_0\big)}{\partial \tau} = \lim_{\Delta\tau \to 0} \frac{F_{R_1^s}\big(\tau + \Delta\tau \mid G(t_0), t_0\big) - F_{R_1^s}\big(\tau \mid G(t_0), t_0\big)}{\Delta\tau}. \tag{4.20}$$

The term $F_{R_1^s}\big(\tau + \Delta\tau \mid G(t_0), t_0\big) - F_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$ is the probability that no events of the *populational dynamics* R^s occurs in the time interval $[t_0, t_0 + \tau]$ and the first event occurs in the small interval $(t_0 + \tau, t_0 + \tau + \Delta\tau]$. The probability of no events over the entire interval $[t_0, t_0 + \tau]$ can be represented as the product of the probabilities of no event in each infinitely small subinterval of length h and the probability that an R_1^s event occurs in the small interval $(t_0 + \tau, t_0 + \tau + \Delta\tau]$:

$$F_{R_1^s}\big(\tau + \Delta\tau \mid G(t_0), t_0\big) - F_{R_1^s}\big(\tau \mid G(t_0), t_0\big) = \lim_{h \to 0} \prod_{i=0}^{\tau/h-1} \Big(1 - h a_0^s(t_0 + ih)\Big) \cdot a_1^s(t_0 + \tau)\Delta\tau. \tag{4.21}$$

Substituting (4.21) into (4.20), the following representation of the density function $f_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$ is derived:

$$f_{R_1^s}\big(\tau \mid G(t_0), t_0\big) = \lim_{\Delta\tau \to 0} \lim_{h \to 0} \Big(\prod_{i=0}^{\tau/h-1} \Big(1 - h a_0^s(t_0 + ih)\Big)\Big) \frac{a_1^s(t_0 + \tau)\Delta\tau}{\Delta\tau}. \tag{4.22}$$

Consider

$$A = \lim_{h\to 0} \prod_{i=0}^{\tau/h-1} \Big(1 - h a_0^s(t_0 + ih)\Big).$$

Taking the natural logarithm on both sides:

$$\ln A = \lim_{h\to 0} \sum_{i=0}^{\tau/h-1} \ln\Big(1 - h a_0^s(t_0 + ih)\Big).$$

For small values of h, $ha_0^s(t_0+ih)$ is also small. Using the Taylor series expansion for $\ln(1-x)$ around $x=0$, which is $\ln(1-x) \approx -x$:

$$\ln A \approx -\lim_{h\to 0} \sum_{i=0}^{\tau/h-1} h a_0^s(t_0 + ih).$$

As $h \to 0$, the summation becomes an integral:

$$\ln A = -\int_{t_0}^{t_0+\tau} a_0^s(t)\mathrm{d}t. \tag{4.23}$$

Substituting A into equation (4.22) gives the desired equation for the PDF of the putative waiting time τ to the next firing of reaction R_1^s:

$$f_{R_1^s}\big(\tau \mid G(t_0), t_0\big) = a_1^s(t_0+\tau)\exp\left(-\int_{t_0}^{t_0+\tau} a_0^s(t)\mathrm{d}t\right). \tag{4.24}$$

Now consider the probability distribution of the waiting time τ generated by the SSATAN-X envelope algorithm, denoted further as $g_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$. The constant $B(T_L)$ is chosen such that it satisfies the condition $B(T_L) \geq a_0^s(t)$ for all $t \in [t_0, t_0+T_L]$.

With the introduction of a look-ahead horizon T_L, the next firing time τ for the reaction R_1^s falls into one of two categories: either $\tau \leq T_L$ or $\tau > T_L$. In the first case, the rejection, outlined in **lines 6 – 9** is infeasible due to how the next putative time Δt is generated. Therefore, there are following possibilities: (i) Time-step Δt is accepted, and reaction R_1^s fires with the probability $a_1^s(t_0+\Delta t)/B(T_L)$. In this scenario it is considered that $\Delta t \equiv \tau$, and the probability of this event is $a_1^s(t_0+\tau)/B(T_L) \cdot B(T_L)\exp(-B(T_L)\cdot\tau) = a_1^s(t_0+\tau)\cdot\exp(-B(T_L)\cdot\tau)$. (ii) Thinning channel fires with probability $(B(T_L) - a_0^s(t_0+\Delta t))/B(T_L)\cdot B(T_L)\exp(-B(T_L)\cdot\Delta t) = (B(T_L) - a_0^s(t_0+\Delta t))\cdot\exp(-B(T_L)\cdot\Delta t)$. Subsequently, the time gets updated on Δt, and the PDF of the next firing time $g_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$ becomes $g_{R_1^s}\big(\tau - \Delta t \mid G(t_0+\Delta t), t_0+\Delta t\big)$. Notably, after updating time t, a new look-ahead horizon and corresponding value of $B(T_L)$ are recalculated. Thinning implies that $\Delta t < \tau$. To account for all possible values of Δt with limits

$(0,\tau)$, definite integral is employed: $\int_0^\tau (B(T_L) - a_0^s(t_0+\Delta t)) \cdot \exp(-B(T_L)\cdot\Delta t) \cdot g_{R_1^s}\big(\tau - \Delta t \mid G(t_0+\Delta t), t_0+\Delta t\big)\mathrm{d}\Delta t$.

In the case of $\tau > T_L$, time-step Δt generated in **line 5** faces rejection with the probability $\exp(-B(T_L)\cdot T_L)$. Consequently, time t advances by T_L, and the pdf $g_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$ transforms into $g_{R_1^s}\big(\tau - T_L \mid G(t_0+T_L), t_0+T_L\big)$. If the generated value of Δt is too small, the thinning event will fire again. A similar definite integral as in the previous scenario is employed to account for all possible values of Δt with limits $(0, T_L)$: $\int_0^\tau (B(T_L) - a_0^s(t_0+\Delta t)) \cdot \exp(-B(T_L)\cdot\Delta t) \cdot g_{R_1^s}\big(\tau - \Delta t \mid G(t_0+\Delta t), t_0+\Delta t\big)\mathrm{d}\Delta t$.

Considering these factors, the recursive equation for the $g_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$ can be formulated:

$$g_{R_1^s}\big(\tau \mid G(t_0), t_0\big) = \begin{cases} \int_0^\tau (B(T_L) - a_0^s(t_0+\Delta t)) \cdot \exp(-B(T_L)\cdot\Delta t) \cdot g_{R_1^s}\big(\tau - \Delta t \mid G(t_0+\Delta t), t_0+\Delta t\big)\mathrm{d}\Delta t + \\ + a_1^s(t_0+\tau)\cdot\exp(-B(T_L)\cdot\tau), \qquad \text{if } \tau \le T_L, \\ \int_0^{T_L} (B(T_L) - a_0^s(t_0+\Delta t)) \cdot \exp(-B(T_L) \cdot \Delta t) \cdot g_{R_1^s}\big(\tau - \Delta t \mid G(t_0+\Delta t), t_0+\Delta t\big)\mathrm{d}\Delta t + \\ + \exp(-B(T_L)\cdot T_L) \cdot g_{R_1^s}\big(\tau - T_L \mid G(t_0+T_L), t_0+T_L\big), \qquad \text{if } \tau > T_L. \end{cases} \tag{4.25}$$

To show that $f_{R_1^s}\big(\tau \mid G(t_0), t_0\big) = g_{R_1^s}\big(\tau \mid G(t_0), t_0\big)$, substitute (4.24) into (4.25). Here,

$$\begin{aligned} f_{R_1^s}\big(\tau - \Delta t \mid G(t_0+\Delta t), t_0+\Delta t\big) &= a_1^s(t_0+\tau)\exp\left(-\int_{t_0+\Delta t}^{t_0+\tau} a_0^s(t)\mathrm{d}t\right), \\ f_{R_1^s}\big(\tau - T_L \mid G(t_0+T_L), t_0+T_L\big) &= a_1^s(t_0+\tau)\exp\left(-\int_{t_0+T_L}^{t_0+\tau} a_0^s(t)\mathrm{d}t\right). \end{aligned} \tag{4.26}$$

Now consider

$$\begin{aligned}
&\int_0^{\tau}(B(T_L)-a_0^s(t_0+\Delta t))\cdot\exp(-B(T_L)\cdot\Delta t)\cdot f_{R_1^s}\big(\tau-\Delta t \mid G(t_0+\Delta t),t_0+\Delta t\big)\mathrm{d}\Delta t = \\
&=\int_0^{\tau}(B(T_L)-a_0^s(t_0+\Delta t))\cdot\exp(-B(T_L)\cdot\Delta t)\cdot a_1^s(t_0+\tau)\exp\left(-\int_{t_0+\Delta t}^{t_0+\tau}a_0^s(t)\mathrm{d}t\right)\mathrm{d}\Delta t = \\
&=a_1^s(t_0+\tau)\int_0^{\tau}(B(T_L)-a_0^s(t_0+\Delta t))\cdot\left[\exp(-B(T_L)\cdot\Delta t)\cdot\exp\left(-\int_{t_0+\Delta t}^{t_0+\tau}a_0^s(t)\mathrm{d}t\right)\right]\mathrm{d}\Delta t = \\
&=a_1^s(t_0+\tau)\int_0^{\tau}(B(T_L)-a_0^s(t_0+\Delta t))\cdot\exp\left(-B(T_L)\cdot\Delta t-\int_{t_0+\Delta t}^{t_0+\tau}a_0^s(t)\mathrm{d}t\right)\mathrm{d}\Delta t = \\
&=a_1^s(t_0+\tau)\int_0^{\tau}(B(T_L)-a_0^s(t_0+\Delta t))\cdot\exp\left(-\int_{t_0}^{t_0+\tau}B(T_L)dt+\int_{t_0+\Delta t}^{t_0+\tau}(B(T_L)-a_0^s(t))\mathrm{d}t\right)\mathrm{d}\Delta t = \\
&=a_1^s(t_0+\tau)\cdot\exp(-B(T_L)\cdot\tau)\int_0^{\tau}(B(T_L)-a_0^s(t_0+\Delta t))\cdot\exp\left(\int_{t_0+\Delta t}^{t_0+\tau}(B(T_L)-a_0^s(t))\mathrm{d}t\right)\mathrm{d}\Delta t = \\
&=a_1^s(t_0+\tau)\cdot\exp(-B(T_L)\cdot\tau)\Big(\exp\left(\int_{t_0}^{t_0+\tau}(B(T_L)-a_0^s(t))\mathrm{d}t\right)-1\Big).
\end{aligned} \tag{4.27}$$

Similarly,

$$\begin{aligned}
&\int_0^{T_L}(B(T_L)-a_0^s(t_0+\Delta t))\cdot\exp(-B(T_L)\cdot\Delta t)\cdot f_{R_1^s}\big(\tau-\Delta t \mid G(t_0+\Delta t),t_0+\Delta t\big)\mathrm{d}\Delta t = \\
&=a_1^s(t_0+\tau)\cdot\exp(-B(T_L)\cdot\tau)\left(\exp\left(\int_{t_0}^{t_0+\tau}(B(T_L)-a_0^s(t))\mathrm{d}t\right)-\exp\left(\int_{t_0+T_L}^{t_0+\tau}(B(T_L)-a_0^s(t))\mathrm{d}t\right)\right).
\end{aligned} \tag{4.28}$$

By substituting (4.26), (4.27), and (4.28) into (4.25), it can be seen that equality holds.

Thus, the distribution of the firing time of the reaction R_1^s is simulated exactly by the SSATAN-X envelope algorithm 8. Consequently, the aforementioned conclusion holds for all reactions R^s. With a mathematical induction, it is easy to prove the equality of the density function

$$f_{SSATAN-X}(\tau_1,R_1^s,\tau_2,R_2^s,\cdots,\tau_n,R_n^s)=f(\tau_1,R_1^s,\tau_2,R_2^s,\cdots,\tau_M,R_M^s),$$

where $f_{SSATAN-X}(\tau_1,R_1^s,\tau_2,R_2^s,\cdots,\tau_n,R_M^s), f(\tau_1,R_1^s,\tau_2,R_2^s,\cdots,\tau_n,R_M^s)$ represent the probability density function of the SSATAN-X envelope algorithm and the network respectively, that reaction R_i^s fires at τ_i for all $i=1,\cdots,M$. Therefore, the SSATAN-X envelope algorithm is exact for the population dynamics reactions, given $B(T_L)$ is correctly selected and contact dynamics is correctly simulated. □

4.3.4 Computational complexity

In the context of stochastic simulations, it is most interesting to consider the expected complexity given a set of parameters (e.g., the mean running time of an algorithm averaged over an ensemble of simulations), not the worst-case complexity usually considered for deterministic algorithms. This sections provides detailed estimation of the expected complexity of the SSATAN-X envelope algorithm.

Lines 3–5. The definition of the look-ahead time-horizon (line 3) and the determination of the next putative time Δt (line 4) both can be executed in constant time with a complexity of $O(1)$. Similarly, calculating the propensity upper bound (line 3) using any of the methods suggested in the previous section can be accomplished in constant time. One approach to achieve this is to store the mean edge-addition/deletion rate instead of computing it repetitively for each calculation.

Line 8. Updating time in case of the Δt exceeding the look-ahead time-horizon is also a constant time operation.

Line 9. The complexity associated with updating the network topology for the time interval $[t, t+T_L]$ is determined by the specific algorithm employed for this task. To account for its general nature, this complexity will be denoted as Φ, where the actual value it takes depends essentially on the chosen algorithm. The value of Φ for certain algorithms is discussed in the further sections.

Line 11. As with **line 9**, the complexity of this operation is denoted as Φ.

Line 12. Advancing the time by a time step Δt is a constant time operation with a complexity of $O(1)$.

Lines 13–14. The complexity of the propensities update and calculation of their total is generally each $O(M_{R^s})$.

Line 18. To identify the reaction channel R^s_ℓ responsible for generating the event, it is necessary to iterate through approximately half of the list of event rates on average. Consequently, this step requires a time complexity of $O(M_{R^s})$, where M_{R^s} denotes a total number of populational dynamics reactions R^s.

Line 19. The execution of the chosen reaction R^s_ℓ can generally be done in constant time.

Line 20. Performing the adaptivity step, which involves updating the network's state and event rates after a triggering event occurs, requires a certain number of operations directly proportional to the number of reaction channels impacted by the event. In the case of a network, this number is often proportional to the average node degree. Typically, the average node degree remains relatively small and does not significantly increase with the number of nodes. Consequently, this step can be executed in constant time, denoted as $O(1)$. However,

in dense or heterogeneous networks, where the number of reaction channels affected by an event may scale with M_{R^s}, this step could require a time complexity of $O(M_{R^s})$.

Overall complexity estimation. The anticipated running time of a *single iteration* of the envelope SSATAN-X envelope algorithm 8 is $O(M_{R^s} + \Phi)$. Here, M_{R^s} pertains to the count of reactions within type R^s (encompassing both spreading and vital dynamics), and Φ signifies the complexity of the algorithm employed to update the contact network. Moreover, the number of simulation steps required is determined primarily by the quantity and velocity of "upper level" populational reactions R^s. This approach significantly reduces the number of simulation steps compared to the SSA, where the number of steps must be adjusted to both reaction groups R^s and R^c. This is particularly advantageous, considering that contact dynamics usually unfold more rapidly than spreading dynamics (see sections 5.2.5 and chapter 7).

4.4 Approximate simulation of contact dynamics: $\boldsymbol{\tau}$-leaping approach

This algorithm for the contact dynamics updates is based on the $\boldsymbol{\tau}$-leaping algorithm, originally proposed by Cao and Gillespie [107] and later refined by Anderson [115], who introduced post-leap checks to enhance accuracy.

Specifically, this algorithm applies to a set of reactions describing the *contact dynamics*, $R^c = \{R^+, R^-\}$. Within the network, there are two primary types of reactions that alter the contact structure. The first type $R^+ = \{R^+_{jk}, \lambda^+_{jk}\}$ denotes the set of reactions that create new connections (or edges) between nodes v_j and v_k ($j \neq k$), occurring at a specified rate λ^+_{jk}. The second type $R^- = \{R^-_{jk}, \lambda^-_{jk}\}$ includes reactions that dissolve existing connections between nodes v_j and v_k ($j \neq k$) at a rate λ^-_{jk}.

Instead of individually simulating each contact-changing event, as is done in methods like the Gillespie algorithm, the tau-leap algorithm leaps over bigger time intervals, called $\boldsymbol{\tau}$-leaps (to prevent confusion with waiting times denoted by τ, the notation will be prominently displayed in bold), which allows the simulation of multiple reaction events simultaneously over each time interval. A visual comparison of these two approaches is presented in Fig. 4.3. Although this strategy reduces the computational burden, especially in cases where there are a large number of contact events to consider, it trades off some accuracy, as the method assumes that reaction propensities remain constant over each time step, which may not always be the case.

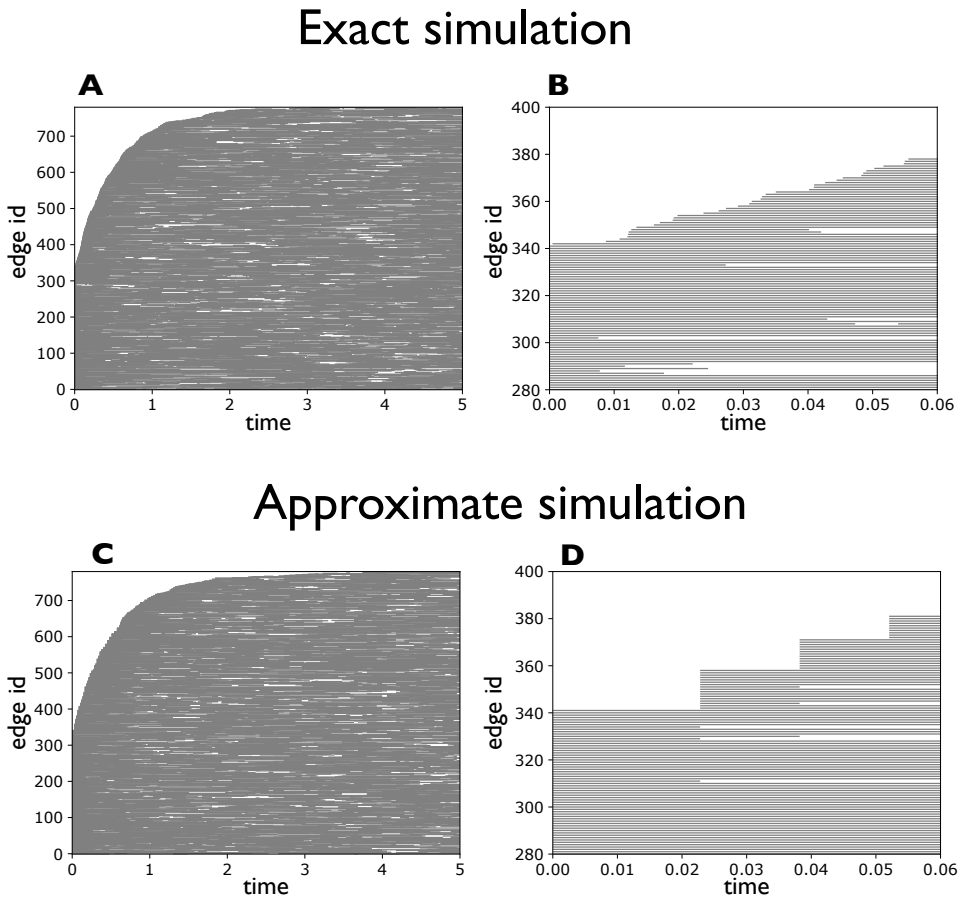


Fig. 4.3. Comparative visualisation of exact vs. approximate contact update approaches. The figure depicts an edge activity plot, illustrating the duration of each edge throughout the simulation. In the exact approach **(A–B)** every contact change is explicitly simulated. In the approximate approach **(C–D)** multiple edge updates occur within a single time step. While at a glance the overall patterns appear similar **(A, C)**, a closer examination **(B, D)** reveals the difference between the two methods.

4.4.1 Algorithm

Algorithm 9 describes the τ-leaping algorithm, modified for an approximate update of the contact dynamics of the network. Aside from the contact graph $G(t)$, the contact dynamics reactions, and the time horizon, the algorithm requires a few design parameters p, p^*, q and ε that can, in principle, take many values. In this work the optimal values derived in [115, 130] was used (e.g., $p = 0.75$, $p^* = 0.9$, $q = 0.98$, $\varepsilon = 0.03$ and $q^* = 1.02$). In addition, the algorithm utilises supplementary storage structures: T^{ξ}, C^{ξ}, and S^{ξ}. The term T^{ξ} denotes the current internal time of a specific subprocess ξ. The cumulative number of firings of reactions of type ξ up to this internal time T^{ξ} are tracked by C^{ξ}. The associated matrix S^{ξ} is employed to capture data from unsuccessful leaps during the post-leap check and is

structured with two columns: the first column records the internal times T^{ξ}, while the second column logs the corresponding C^{ξ} values.

For an approximate network update, there is a limitation on the number of edges that can be added or removed during one approximation step. The potential for edge deletion is capped by the number $|E(t)|$ of edges existing in the network at any given time t. Furthermore, the maximum number of edges in a network is restricted by the size of the population it represents. For a finite population of N individuals, the upper limit of edges is $N(N-1)/2$, if no other stricter constrains are specified. As such, the available slots for adding new edges at any time can be calculated as $N(N-1)/2-|E(t)|$. Notably, this limitation is relevant only in scenarios where no additional constraints on the number of individual contacts are imposed. For a more detailed exploration of situations in which such constraints are relevant, refer to chapter 7.

The process of calculating the limitations on the number of edges that can be added, as well as the mechanics of adding and deleting edges, is more straightforwardly executed using the complement graph approach. The contact network $G(t)=\{V(t),E(t)\}$ is complemented by the graph $G'(t)=\{V(t),E'(t)\}$. The contact graph and its complement are interlinked: a deletion of an edge from the contact graph $G(t)$ corresponds to the addition of the same edge in the complement graph $G'(t)$, and vice versa. This relationship simplifies the simulation of contact dynamics, as it allows for the implementation of less complex edge-search algorithms. With this terminology, the processes of constructing and dismantling edges in the network can be described by the following first-order reactions:

- **Edge addition:** If reaction R^{+}_{jk} fires with rate $e'_{jk}\cdot\lambda^{+}_{jk}$, set $e'_{jk}=0$ and $e_{jk}=1$.
- **Edge deletion:** If reaction R^{-}_{jk} fires with rate $e_{jk}\cdot\lambda^{-}_{jk}$, set $e'_{jk}=1$ and $e_{jk}=0$.

In this context, e_{jk} and e'_{jk} denote the presence (with a value of 1) or absence (with a value of 0) of an edge between nodes v_j and v_k ($j\neq k$) in the corresponding graph.

At a given time t, the time step $\boldsymbol{\tau}$ for the leap is calculated (**line 4** of the **Algorithm 9**) based on the following formulas:

$$\boldsymbol{\tau}=\min\left\{\frac{\max(\varepsilon|E(t)|,1)}{|\mu_E(t)|},\frac{\left(\max(\varepsilon|E(t)|,1)\right)^2}{\sigma^2_E(t)},\frac{\max(\varepsilon|E'(t)|,1)}{|\mu_{E'}(t)|},\frac{\left(\max(\varepsilon|E'(t)|,1)\right)^2}{\sigma^2_{E'}(t)}\right\}. \tag{4.29}$$

Here $|E(t)|$, $|E'(t)|$ represent the counts of edges in the contact network and its complement graph at time t respectively, and $\varepsilon=0.03$ is a preset parameter mentioned above. The expected change in the number of edges within both the network graph and its complement

Algorithm 9. Approximate contact dynamics update: $\boldsymbol{\tau}$-leap approach.

Input T_F, $G(0)$

1: Set $T^{\xi} = C^{\xi} = 0$, $S^{\xi} = [0,0]$ for each reaction type $\xi = \{+,-\}$
2: t = 0
3: Calculate propensities R^{ξ}_{jk} for $\xi = \{+,-\}$ based on the current state of $G(t)$, their totals $R^{\xi}_0 = \sum R^{\xi}_{jk}$, as well as $R^c_0 = \sum_{\xi} R^{\xi}_0$
4: Calculate $\boldsymbol{\tau}$ according to eq. (4.29)
5: **while** $t < T_F$ **do**
6: $\quad \tau = \min(\tau, T_F - t)$
7: $\quad$ **if** $\tau < \dfrac{h}{R_0}$ **then**
8: $\quad\quad$ Execute SSA 100 times
9: $\quad\quad$ Return to **line 3**
10: $\quad$ **else**
11: $\quad\quad$ **for** ξ **do**
12: $\quad\quad\quad$ H^{ξ} = number of rows in S^{ξ}
13: $\quad\quad\quad$ **if** $R^{\xi}_0 \tau + T^{\xi} \geq S^{\xi}[H^{\xi}, 1]$ **then**
14: $\quad\quad\quad\quad$ $M^{\xi} \sim \mathscr{P}\big(R^{\xi}_0 \boldsymbol{\tau} + T^{\xi} - S^{\xi}[H^{\xi}, 1]\big) + S^{\xi}[H^{\xi}, 2] - C^{\xi}$
15: $\quad\quad\quad\quad$ $row^{\xi} = H^{\xi}$
16: $\quad\quad\quad$ **else**
17: $\quad\quad\quad\quad$ Find K^{ξ} such that $S^{\xi}[K^{\xi} - 1, 1] \leq R^{\xi}_0 \boldsymbol{\tau} + T^{\xi} < S^{\xi}[K^{\xi}, 1]$
18: $\quad\quad\quad\quad$ $u = \frac{R^{\xi}_0 \boldsymbol{\tau} + T^{\xi} - S^{\xi}[K^{\xi}-1,1]}{S^{\xi}[K^{\xi},1] - S^{\xi}[K^{\xi}-1,1]}$
19: $\quad\quad\quad\quad$ $M^{\xi} \sim \mathscr{B}\big(S^{\xi}[K^{\xi}, 2] - S^{\xi}[K^{\xi} - 1, 2], u\big) + S^{\xi}[K^{\xi} - 1, 2] - C^{\xi}$
20: $\quad\quad\quad\quad$ $row^{\xi} = K^{\xi} - 1$
21: $\quad\quad$ **if** leap conditions (4.32) holds **then**
22: $\quad\quad\quad$ **for** ξ **do**
23: $\quad\quad\quad\quad$ Delete all rows from S^{ξ} on the positions less than or equal to row^{ξ}
24: $\quad\quad\quad\quad$ Add new first row $[R^{\xi}_0 \boldsymbol{\tau} + T^{\xi}, C^{\xi} + M^{\xi}]$
25: $\quad\quad\quad\quad$ $T^{\xi} = T^{\xi} + R^{\xi}_0 \boldsymbol{\tau}$
26: $\quad\quad\quad\quad$ $C^{\xi} = C^{\xi} + M^{\xi}$
27: $\quad\quad\quad$ $t = t + \boldsymbol{\tau}$
28: $\quad\quad\quad$ **if** leap would have failed the leap condition for $\varepsilon' = 0.75\varepsilon$ **then**
29: $\quad\quad\quad\quad$ $\boldsymbol{\tau} = \boldsymbol{\tau} p^*$
30: $\quad\quad\quad$ **else**
31: $\quad\quad\quad\quad$ $\boldsymbol{\tau} = \boldsymbol{\tau}^q$ if $t < 1$ and otherwise $\boldsymbol{\tau} = \boldsymbol{\tau}^{q^*}$
32: $\quad\quad\quad$ Update network by executing M^{ξ} reactions using, for example, Alg. 10
33: $\quad\quad\quad$ Recalculate R^{ξ}_{jk}, R^{ξ}_0 and R_0 based on updated network state $G(t)$
34: $\quad\quad$ **else**
35: $\quad\quad\quad$ **for** ξ **do**
36: $\quad\quad\quad\quad$ Add row $[R^{\xi}_0 \tau + T^{\xi}, C^{\xi} + M^{\xi}]$ between row^{ξ} and $row^{\xi} + 1$
37: $\quad\quad\quad$ $\boldsymbol{\tau} = \boldsymbol{\tau} p$

is calculated as follows:

$$\begin{aligned} \mu_E(t) &= -R_0^- + R_0^+, \\ \mu_{E'}(t) &= -R_0^+ + R_0^-, \end{aligned} \tag{4.30}$$

where $R_0^{+/-}$ denotes the sum of reaction propensities for addition and deletion of edges respectively at the current network state $G(t)$. The variance of the edge change is given by

$$\sigma_E^2(t) = \sigma_{E'}^2(t) = R_0^- + R_0^+. \tag{4.31}$$

These equations represent a specific case derived from the original formulation found in [115], considering that both addition and deletion reactions are first-order reactions in the contact network and its complement graph.

If the chosen $\boldsymbol{\tau}$ is smaller than the ratio h/R_0, where h is typically set to 10, a predetermined number of steps, usually 100 [106], are executed using the SSA (**lines 7–9**). It is important to accurately manage the values T^ξ, C^ξ and the S^ξ during the SSA execution to maintain precision. Further specifics about this process can be found in the original article [115].

Alternatively, when the leap is initiated, the algorithm samples the number of executions M^+ and M^- for each reaction type, R^+ and R^- (**lines 11–20**). This sampling is done using either a Poisson distribution (**line 14**) or a binomial distribution (**line 19**), depending on the context. The leap is then validated based on a specific condition

$$\begin{aligned} \Big||E(t+\boldsymbol{\tau})| - |E(t)|\Big| &\leq \max(\varepsilon|E(t)|, 1), \\ \Big||E'(t+\boldsymbol{\tau})| - |E'(t)|\Big| &\leq \max(\varepsilon|E'(t)|, 1). \end{aligned} \tag{4.32}$$

If these conditions are met, the leap is accepted (**line 21**), the storage variables T^ξ, S^ξ and C^ξ are updated (**lines 22–26**), the time t is incremented by $\boldsymbol{\tau}$ (**line 27**), and the value of $\boldsymbol{\tau}$ is adjusted (**lines 28–31**). The bulk update (**line 32**) of the contacts is executed, applying the chosen values of M^ξ. One of the possible variations of the network update algorithm is detailed in Algorithm 10.

In the case of rejection, the sampled values are recorded (**lines 35–36**). Subsequently, $\boldsymbol{\tau}$ is reduced by multiplying it with a design parameter $0 < p < 1$. The algorithm then loops back to **line 5** for re-evaluation.

4.4.2 Network Update

Once the number of contact updates to be performed is determined, the actual execution of these updates must preserve the computational advantage of the algorithm. Randomly adding and deleting the predetermined number of edges may adversely affect accuracy, particularly in highly heterogeneous networks. Therefore, it is equally important to establish the order in which the edges will be modified. On the other hand, using direct approaches like the reaction choice step in Gillespie's algorithm requires recalculation of the propensity sum each time the reaction is performed.

Algorithm 10 outlines a method to perform the network update while largely preserving the order of contact modifications. This approach need not recalculate the propensity sum, resulting in faster performance, although at the possible cost of slight bias.

Algorithm 10. Network Update.

Input M^{ξ} for each reaction type $\xi = \{+,-\}$, $G(0)$

1: Shuffle M^- deletion and M^+ addition reactions. Save the order in vector **A**
2: Calculate cumulative sums vectors Λ^+ and Λ^-, total propensity sums R_0^- and R_0^+
3: **for** each contact update event type in **A do**
4: Sample $u \sim \mathscr{U}(0,1)$
5: **if** deletion **then**
6: Find first y such as: ${\Lambda^-}_y \geq u \cdot R_0^-$
7: Remove and edge $\{v_j, v_k\}$ corresponding to index y from the network
8: Remove the element ${\Lambda^-}_y$ from Λ^-
9: Decrease R_0^- to the propensity of chosen reaction: $R_0^- = R_0^- - R_{jk}^-$
10: Calculate $tmp = R_{jk}^+ +$ value of last element in Λ^+
11: Append tmp to the end of Λ^+
12: $R_0^+ = R_0^+ + R_{jk}^+$
13: **else if** addition **then**
14: Find first y such as: ${\Lambda^+}_y \geq u \cdot R_0^+$
15: Add edge $\{v_j, v_k\}$ corresponding to index y to the network.
16: Remove the element from Λ^+
17: Decrease R_0^+ to the propensity of chosen reaction: $R_0^+ = R_0^+ - R_{jk}^+$
18: Calculate $tmp = R_{jk}^- +$ value of last element in Λ^-
19: Append tmp to the end of Λ^-
20: $R_0^- = R_0^- + R_{jk}^-$

In **line 1**, the sequence for adding and removing edges is established and recorded in the vector **A**. For example, if $M^+ = 3$ and $M^- = 5$ was proposed, necessitating three edge-addition operations, denoted as $\{+,+,+\}$, and five edge-deletion operations, denoted as $\{-,-,-,-,-\}$ are to be executed. These operations are randomly mixed, resulting in the

vector **A** being structured as $\mathbf{A} = \{-,-,+,+,-,-,-,+\}$, representing the shuffled order of these addition and deletion actions.

Then, in **line 2**, two vectors of cumulative sums are created: Λ^- for the edge-deletion reactions and Λ^+ for the edge-addition reactions. These vectors contain cumulative sums of corresponding reaction propensities, and the last element of each vector always represents an upper bound for the reaction propensity sum of the corresponding reaction type.

For each reaction in the vector **A** (**lines 3– 20**), the algorithm selects an edge to be either added or deleted. This selection depends on the values in the corresponding cumulative sum vector Λ^ξ. This selection is done by checking whether the condition ${\Lambda^\xi}_y \geq u \cdot R_0^\xi$ is met (**lines 6, 14**), $\xi = \{+,-\}$. This step resembles the reaction selection step of the SSA algorithm and ensures that the probability of particular edge $\{v_j, v_k\}$ being modified is influenced by the propensity of that modification. Based on the outcome of the selection process, the algorithm performs the corresponding network change – either the addition or deletion of an edge.

Under the assumptions regarding the contact limitations made in this chapter, each edge addition in the network introduces a new reaction for edge deletion, and conversely, the deletion of an edge prompts a reaction for edge addition. To account for such changes and maintain accuracy, whenever a new edge is added or deleted, the propensity for the corresponding counterreaction (edge deletion or addition, respectively) is calculated. The calculated propensity of the new reaction is then added to the current upper bound, which is the last element of the cumulative sums vector (**lines 11, 19**). This updated value is subsequently appended to the end of the cumulative sum vector, ensuring that the vector reflects the current state. After the execution of each reaction, the corresponding propensity totals, R_0^- and R_0^+ (**lines 9, 12, 17, 20**), must be updated to ensure that the upper bound $u \cdot R_0^\xi$ is not violated. Failing to update these values could lead to significant inaccuracies in the simulation, as the probability of reaction selection would not correctly reflect the current state of the network.

Because vectors of the cumulative propensities sum Λ^ξ are not updated during the execution of the loop, the algorithm does not achieve exact precision. However, it still offers a valuable approximation of the contact dynamics while avoiding the need to recalculate the total propensity sum after each step.

4.4.3 Computational complexity

In this section the expected complexity of the Algorithm 9 is estimated.

Lines 1–2. The initialisation of the algorithm parameters is a constant time operation with a complexity of $O(1)$ each.

Lines 3, 33. To update the reaction propensities and calculate their sum, it is necessary to iterate through the list of event rates. When implemented with care, only one iteration through the list is needed, making the complexity of the steps described in the lines 3 and 33 $O(M_{R^c})$, where M_{R^c} denotes the number of reactions in the subgroup R^c responsible for the contact dynamics.

Lines 4, 6. Calculation of the leap size τ in line 4, and the subsequent adjustment to the simulation time in line 6 are both operations with a time complexity of $O(1)$.

Lines 7–9. Although this is a component of the algorithm, transitioning to the SSA algorithm implies that only a few reactions need to be performed during the chosen leap time interval, making the SSA algorithm more efficient in such cases. This typically occurs when the contact dynamics of the modelled system are relatively slow or have a comparable speed to the transition/vital dynamics R^s. However, the proposed algorithm is specifically designed for models with relatively fast contact dynamics, which is the case for most real-life spreading processes. Consequently, the transition described in lines 7–9 occurs infrequently. If it does occur, the complexity is determined by the SSA Algorithm 1, as discussed in chapter 3, for a fixed and constant number of runs.

Lines 11–20, 22–26,35–36. Since there are only two possible reaction types, denoted as $\xi = +, -$, which correspond to adding and deleting an edge, respectively, all loops described in lines 11–20, 22–26, and 35–36 can be executed in constant time.

Line 32. The complexity of the network update, which relies on the precalculated numbers of edge operations M^ξ, is contingent upon the specific algorithm employed, and for the general case will be denoted as Ψ.

For the approach described in **Algorithm 10**, the complexity of constructing the vector of reaction order $\mathbf{A}$ is linear and depends on the total number of changes to be made. It is denoted as $O(\sum_\xi M^\xi)$, where M^ξ represents the number of changes for each reaction type ξ. The complexity of building the cumulative sum vectors, Λ^+ and Λ^-, is $O(M_{R^+})$ and $O(M_{R^-})$, respectively. Here, M_{R^ξ} is the total number of contact-changing reactions of a particular type, $\xi = \{+, -\}$. Since the cumulative sum vectors Λ^+ and Λ^- are sorted arrays, the search operation can be performed in $O(\log(M_{R^-}))$ or $O(\log(M_{R^+}))$, respectively. As the loop iterates over the array $\mathbf{A}$, the overall complexity of the network update step is approximately $\sum_\xi M^\xi + M_{R^c} + \log(\prod_\xi M_{R^\xi})$, which can be approximated as $O(M_{R^c})$.

Lines 28–31, 37. The update of the leap size τ can be executed in constant time, resulting in a complexity of $O(1)$.

Overall complexity estimation. The expected complexity of a single iteration of **Algorithm 9** is $O(M_{R^c} + \Psi)$, where M_{R^c} is the total number of contact dynamics reactions R^c, and Ψ is a complexity of the actual network update.

Although this approach still exhibits linear scaling in relation to the number of contact reactions, the actual number of leaps required is significantly lower than the number of steps needed in an exact simulation. This reduction in steps allows for a more rapid simulation of contact events that occur between spreading events R^s.

4.5 Exact simulation of contact dynamics: Next reaction method approach

Another alternative for simulating the contact dynamics is to utilise the next reaction method (Algorithm 2), previously described in chapter 3. The putative time Δt for the reactions of type R^s, as defined in the SSATAN-X envelope algorithm, is determined based on the upper bound of the propensities total. This choice of Δt ensures that contact updates carried out within this time frame cannot result in an earlier putative time. Hence, there is no need to update the propensities of contact-dependent reactions in R^s every time a contact change occurs. Instead, these propensities can be recalculated once the entire network update for the given time frame is completed. This allows the implementation of the next reaction method algorithm for the contact dynamics reactions R^c by utilising an indexed priority queue Q, containing the reactions responsible for edge additions and edge deletions. Within the priority queue, reactions with smaller putative times are assigned the highest priority. These putative times are determined by the reaction propensities and sampled according to the principles of the next reaction method. Furthermore, as earlier discussed, the reactions for edge additions and deletions are complementary. Hence, there is no need to extract a reaction from the queue and add its complementary reaction at the end of the queue, which would require $2(\log(M_{R^c}))$ operations. Instead, the reaction type, propensity, putative time, and position in the queue can all be updated within the $\log(M_{R^c})$ operations. Here, M_{R^c} denotes the total number of contact dynamics reactions R^c.

4.5.1 Algorithm

First, a priority queue is constructed to hold all of the contact change reactions along with their corresponding absolute times (**line 2**). Subsequently, as long as the simulation time has not been exceeded, the reactions are executed sequentially (**lines 3–11**). Upon executing a reaction, the complementary reaction is introduced into the queue in its place (**8**). The propensity for the new reaction is recalculated (**line 9**), and a new putative time is sampled for the reaction (**line 10**). Finally, the position of the reaction in the priority queue is updated based on its new time value (**line 11**).

Algorithm 11. Exact contact dynamics update: next reaction method.

Input T_F. $G(0)$

1: t = 0
2: Sample all putative times $\tau_j \sim Exp(a_j)$ of all possible reactions R^c ; store absolute times $t_j = t + \tau_j$ in a priority queue Q
3: **while** t_j of the reaction $Q.top \leq T_F$ **do**
4: Get the reaction R_μ with the smallest reaction time t_μ from Q
5: $t = t_\mu$
6: execute reaction R_μ
7: ▷ *Update the reaction to the complementary one* ◁
8: Change the reaction type of R_μ to the complementary one
9: Set the corresponding reaction propensity a_μ
10: Sample new putative time $\tau_\mu \sim Exp(a_\mu)$ and store times $t_\mu = t + \tau_\mu$ in a priority queue Q
11: Update the position of the reaction R_μ in the priority queue Q

Importantly, in addition to simulating the network updates, the contact dynamics reactions can only be affected by the envelope algorithm during the adaptivity step (line 20 in **Algorithm 8**). Consequently, it is possible to further optimise the algorithm by constructing the priority queue only once at the beginning of the simulation and updating affected propensities only when necessary, due to the adaptivity step.

4.5.2 Computational complexity

Line 1. The initialisation of the simulation time is a constant time operation.
Line 2. The formation of the priority queue has a complexity of $O(M_{R^c}\log(M_{R^c}))$.
Lines 4–5. Accessing the top element of the queue and assigning a new time value are both constant time operations with a complexity of $O(1)$.
Line 6. It can generally be assumed that executing a unitary contact change takes constant time, resulting in a complexity of $O(1)$ for this step.
Lines 8–10. Updating the current reaction to a complementary one can also be performed in constant time, with a complexity of $O(1)$.
Line 11. Updating the position of the reaction in the queue has a complexity of $O(\log(M_{R^c}))$.
Overall complexity estimation. The loop defined in **lines 3–11** repeats as long as the simulation time does not exceed the predefined time limit. Assuming that during this time m simulations are performed, the complexity of the loop is $O(m\log(M_{R^c}))$. Thus, the overall

complexity of **Algorithm 11** scales as $O(M_{R^c}\log(M_{R^c})) + O(m\log(M_{R^c}))$. As noted above, if the priority queue is formed only once and updated during the simulation, the complexity of **Algorithm 11** reduces to $O(m\log(M_{R^c}))$. However, this approach will introduce the additional complexity of $O(\log(M_{R^c}))$ to **Algorithm 8** at the adaptivity step, due to the need to update specific reactions in the priority queue Q, and the complexity of the initial queue initialisation performed only once, remains as $O(M_{R^c}\log(M_{R^c}))$.

4.6 Summary

This chapter introduces a hybrid algorithm designed to simulate the spreading processes on temporal adaptive networks. It combines exact and approximate simulation methods to efficiently handle the dynamics of these complex systems. The chapter elaborates on the assumptions the author made on the generalised model, the application of the Extrande approach for exact spreading simulation, and both the $\boldsymbol{\tau}$-leap approach for approximate contact dynamics and the next reaction method for exact contact dynamics simulation. It also discusses the computational complexity and effectiveness of these methods, providing a comprehensive view of their capabilities in simulating dynamic network behaviours.

Chapter 5

Simulation Findings

5.1 Introduction

This chapter presents an analysis of the results from simulations of an abstract but representative model using the SSATAN-X – hybrid algorithm to simulate spreading processes on temporal adaptive networks, introduced in Chapter 4. The aim is to demonstrate the algorithm's effectiveness and robustness under various network conditions. The chapter begins by describing the genaralised model used for validation. Then the algorithm's ability to simulate both contact and spreading dynamics accurately is explored. The chapter also discusses the influence of network adaptivity on these dynamics, offering important insights into the interplay between network structure, individual's behaviour and spreading process. Finally, this chapter evaluates the computational performance particularly in scenarios where contact dynamics outpace spreading dynamics, as in many epidemiological diseases. This analysis not only confirms the algorithm's theoretical strengths but also showcases its potential to address real-world challenges in phenomena spreading on various dynamic network, thereby making a valuable contribution to the field.

5.2 Adaptive model

This chapter evaluates the simulation algorithm SSATAN-X by applying it to a basic spreading process occurring on a time-varying adaptive contact network. To test the effectiveness of the proposed algorithm, an illustrative scenario involving adaptivity is considered. Specifically, this scenario features discontinuous rewiring dynamics, a challenging numerical problem.

Let $\mathfrak{P} = \{\mathbf{S}, \mathbf{I}, \mathbf{D}\}$ represent a population comprising susceptible ($\mathbf{S}$), infected ($\mathbf{I}$) and diagnosed ($\mathbf{D}$) individuals. The number of contacts C_j for each individual v_j within this

population changes over time. There are no strict limitations on the number of contacts an individual can have at any given time. Therefore, it is essentially limited only by the size of the population. In this scenario, the transmission of a virus, for instance, can occur from an infected individual to a susceptible individual or from a diagnosed individual to a susceptible individual, but only when the respective individuals come into contact with each other. For simplicity, the birth process has been excluded from this model. In general, however, the birth process can be included as a component of populational dynamics.

Reactions R^s of the population dynamics group can be summarised as follows:

- spreading of the pathogen from the infected individual (I_j) to the susceptible (S_k) with the rate $\gamma_{j,k} : I_j + S_k \xrightarrow{\gamma_{j,k}} I_j + I_{N_I+1}$, at rate $\gamma_{j,k} > 0$ if corresponding nodes v_j and v_k are connected;
- spreading of the pathogen from the diagnosed individual (D_i) to the susceptible (S_k) with the rate $\gamma_{i,k} : D_i + S_k \xrightarrow{\gamma_{i,k}} D_i + I_{N_I+1}$, at rate $\gamma_{i,k} > 0$ if corresponding nodes v_i and v_k are connected;
- diagnosis of the infected individual (I_j) with the rate δ_j: $I_j \xrightarrow{\delta_j} D_{N_D+1}$;
- death of the infected individual (I_j) with the rate β: $I_j \xrightarrow{\beta_j} \varnothing$; and
- death of the diagnosed individual (D_i) with the rate β_i: $D_i \xrightarrow{\beta_i} \varnothing$.

To describe the contact dynamics within the population, the "contact activity" of each individual is described by the rates of acquiring and losing contacts over time, represented as λ_j^+ and λ_j^-, respectively. The formation of a new contact between individuals v_i and v_j is possible at a rate of $\lambda_{jk}^+ = \lambda_j^+ \cdot \lambda_k^+$. Similarly, the dissolution of already existing contact occurs at a rate of $\lambda_{jk}^- = \lambda_j^- \cdot \lambda_k^-$.

When an individual is diagnosed with the infection, they become aware of their status, leading to a change in their behaviour. This adaptive behaviour is incorporated into the model, specifically through a discontinuous adjustment. In this example, upon diagnosis, the individual severs all existing contacts, and their rate of establishing new contacts decreases to 30% of its value prior to diagnosis. Consequently, this simple model captures the concept of adaptive dynamics, where the contact dynamics undergo discontinuous changes based on the epidemic state.

Unless otherwise specified, the following setup will be used in this section: The population was initially composed of $N = 200$ nodes, with the distribution of individuals as follows: 180 susceptible (S), 20 infected (I), and 0 diagnosed (D). Additionally, the initial network consisted of 3000 randomly created edges.

Each node v_j was assigned two parameters: λ_j^+ representing the rate of establishing a new contact, and λ_j^- representing the rate of losing an existing contact. These parameters were randomly drawn from uniform distributions within the ranges of (0.5, 2.5) for λ_j^+ and (0.4, 2.0) for λ_j^-.

For the population dynamics, the transmission rate for each $I-S$ edge (contact) was set to $\gamma_{jk} = \gamma = 0.004$, and it was set to $\gamma/2$ for each $D-S$ edge. The death rate for infected and diagnosed individuals was $\beta_i = \beta = 0.08$. The diagnosis rate was set to $\delta_j = \delta = 0.5$, indicating the frequency of individuals becoming diagnosed.

To evaluate the system, 10^3 simulations of each algorithm and parameter set were performed over the time interval $t \in [0,5]$.

The results presented in this section were achieved by employing the *approximate network update* approach, as outlined in section 4.4. Since the accuracy of the exact approach is evident and does not require confirmation, the focus of this analysis was placed on evaluating the performance and effectiveness of the approximate approach.

The results obtained from utilising the SSATAN-X algorithm are compared and benchmarked against the results achieved using the SSA algorithm, considered the ground truth in this context. By comparing the outcomes of both algorithms under the same model settings, the performance and effectiveness of SSATAN-X can be evaluated and assessed.

5.2.1 Correctness of contact approximation

To evaluate the accuracy and performance of SSATAN-X, initial simulations were performed with a specific focus on contact dynamics. These simulations aimed to analyse the behaviour and effectiveness of SSATAN-X in accurately capturing and modelling the dynamics of contacts within the population.

The exact SSA (Algorithm 1) modifies the contact network by executing each change one by one, as depicted in Fig. 4.3A,B. In contrast, SSATAN-X employs bulk updates of the contact network, as outlined in Algorithm 9 and illustrated in Fig. 4.3C, D. It is important to note SSATAN-X's main principle is to accurately capture the statistics of the contact dynamics R^c that are relevant to modelling the population dynamics R^s. The result is that the statistics of the contact dynamics only need to be precise at the time when a bulk update occurs, but not in between bulk updates.

To evaluate the accuracy of SSATAN-X in capturing the statistics of the contact dynamics, dedicated simulations were performed with a specific focus on the pure contact dynamics aspect. These simulations were designed such that no epidemic transitions occurred. In other words, the reaction rates a_ℓ for all epidemic reactions $R_\ell^s \in R^s$ were set to zero. By excluding

the influence of epidemic transitions, the simulations allowed for a detailed analysis and evaluation of the contact dynamics in isolation.

Fig. 5.1 illustrates the degree distribution of the contact network at various time points, comparing the results obtained from SSA and SSATAN-X. The simulation begins with a deterministic initial network, as indicated by the absence of shaded areas in Fig. 5.1A. As the simulation progresses, the degree distribution of the temporal network undergoes changes (Fig. 5.1B–D) until it eventually reaches a stable distribution (Fig. 5.1E–F).

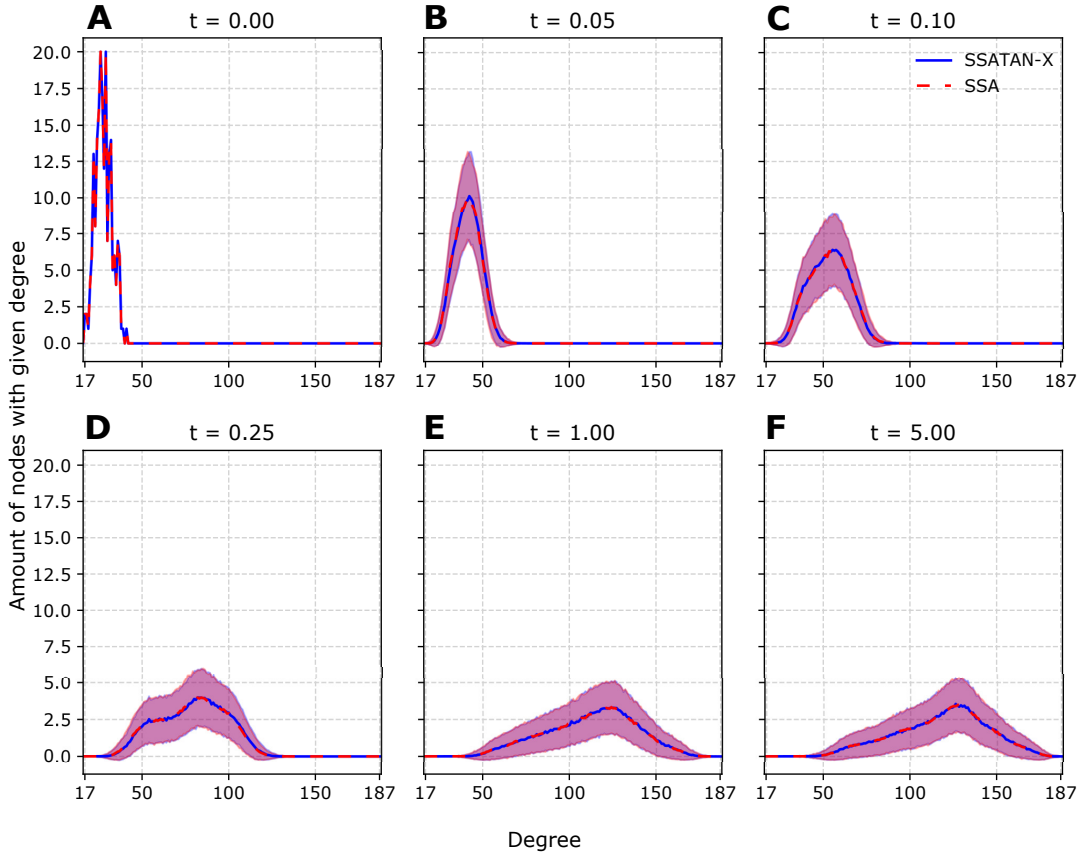


Fig. 5.1. Degree distributions of contacts within the population during simulation. The graphic depicts the mean degree distribution of the network during distinct time points within the simulation time ($t \in [0,5]$). The dashed red and solid blue lines represent mean degree distribution over 10^3 simulations using the SSA and SSATAN-X respectively. Shaded areas represent the corresponding standard deviations for the 10^3 simulations.

Throughout the entire simulation interval, both the mean degree distribution (solid blue line for SSA, dashed red line for SSATAN-X) and the standard deviations (blue shading for SSA, red shading for SSATAN-X) remain visually indistinguishable between the two algorithms.

To provide a more quantitative analysis of the differences in the evolving contact network over time, the Kolmogorov–Smirnov test was employed, and the corresponding statistics were calculated. This test allowed for a rigorous assessment of the degree of similarity between the contact network distributions generated by SSA and SSATAN-X simulations. Fig. 5.2A shows the probability (p-value) that the degree distributions generated by SSA and SSATAN-X are identical for each time point. It can be observed that the probability is consistently greater than 0.95, indicating that the degree distributions from both algorithms are statistically similar.

In Fig. 5.2B, the test statistic of the Kolmogorov–Smirnov test, which represents the distance between the empirical cumulative distribution functions (ECDFs), is presented. The test statistic remains below the threshold of 0.0015 for all time points. This result further reinforces that the observed differences in the degree distribution between SSA and SSATAN-X are statistically insignificant.

Overall, the results from the Kolmogorov–Smirnov test confirm that there are no substantial differences in the degree distribution of the evolving contact network between SSA and SSATAN-X simulations.

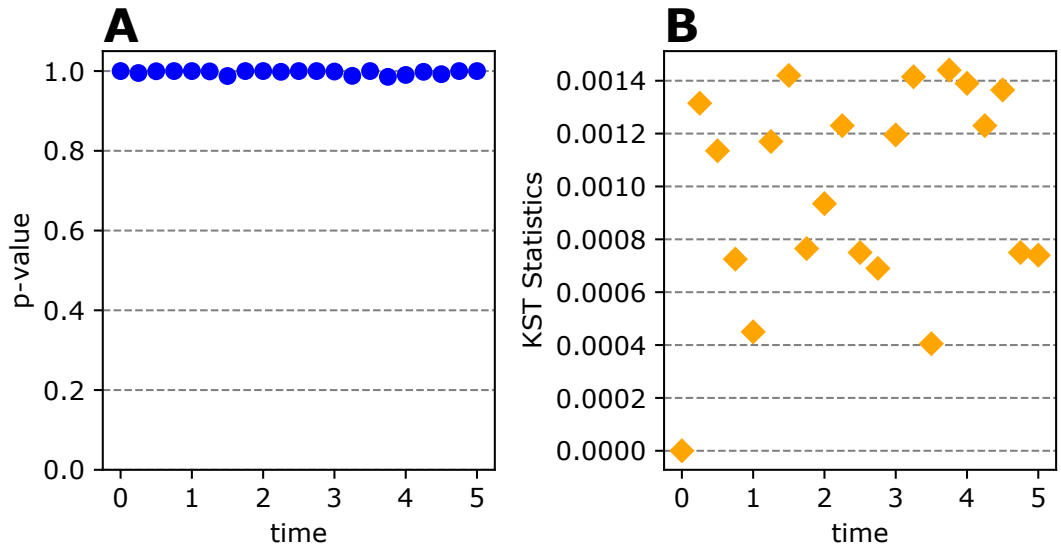


Fig. 5.2. Statistical assessment of differences in the simulated contact network. The Kolmogorov–Smirnov test was used to compare the degree distributions obtained by SSA and SSATAN-X (represented on Fig. 5.1). **(A)** The p-value denotes the probability that these two degree distributions derived from SSA- and SSATAN-X simulations, respectively, are identical. **(B)** Statistics of the Kolmogorov–Smirnov test (KST), representing the distance between the two ECDFs.

5.2.2 Correctness of spreading process simulation

In section 4.3.3, it was demonstrated that Algorithm 8 guarantees exactness in simulating the population dynamics R^s when an upper bound B_{T_L} for the sum of propensities $a_0^s(h) = \sum_\ell a_\ell^s(h)$ of the reactions $R_\ell^s \in R^s, h \in [t, t+T_L]$ is existent and chosen correctly. Building upon this theoretical foundation, the results of applying the SSATAN-X algorithm to the adaptive susceptible-infected-diagnosed (SID) network model described in the previous section are now presented.

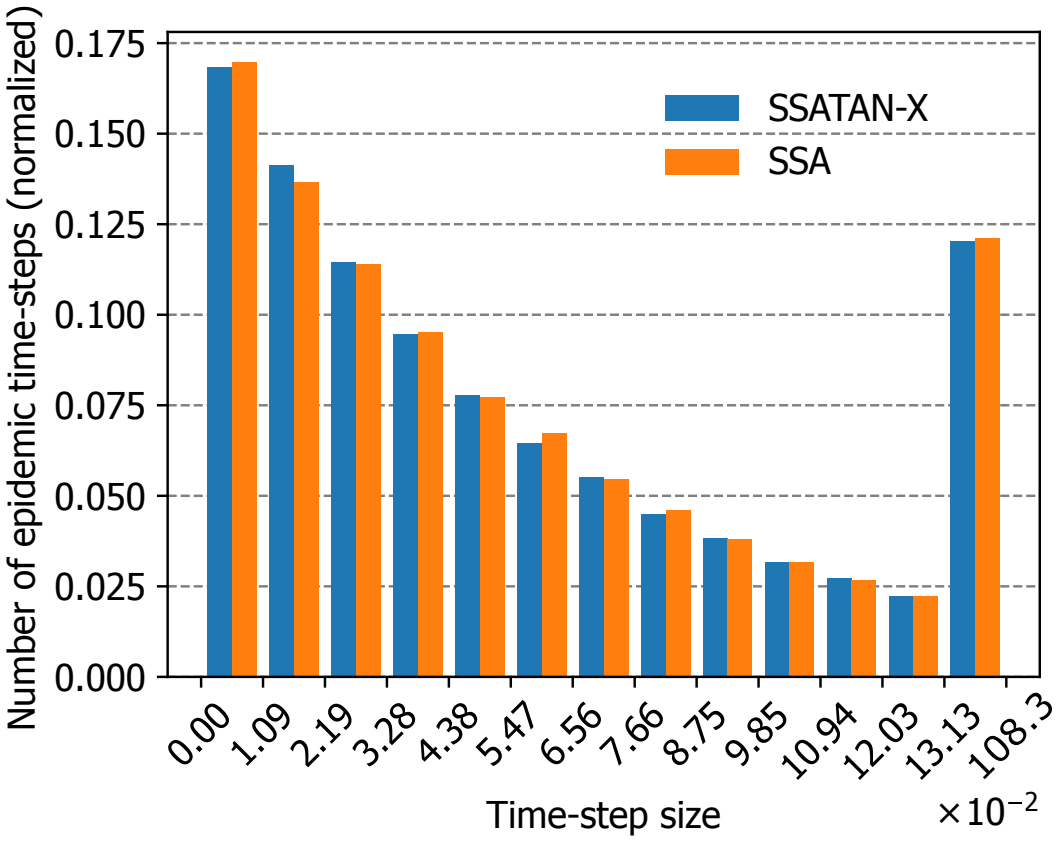


Fig. 5.3. Inter-event time distribution. The histogram depicts the frequency of the time-step sizes to the next epidemic event (infection, diagnosis or death), produced from 10^3 simulations for each algorithm and normalised over the number of samples (overall 79375 *epidemic* time steps for SSATAN-X (blue bars) and 79025 for SSA (orange bars)).

First and foremost, the objective was to examine the accuracy of the statistics of the inter-event times of the population events R^s. In Fig. 5.3, the histogram displays the time steps to the next event (infection, diagnosis, death) for both the SSA (orange bars) and SSATAN-X (blue bars) algorithms. Evidently, the statistics of the inter-event times are virtually identical between the two algorithms, reinforcing the validity of the proposed algorithm.

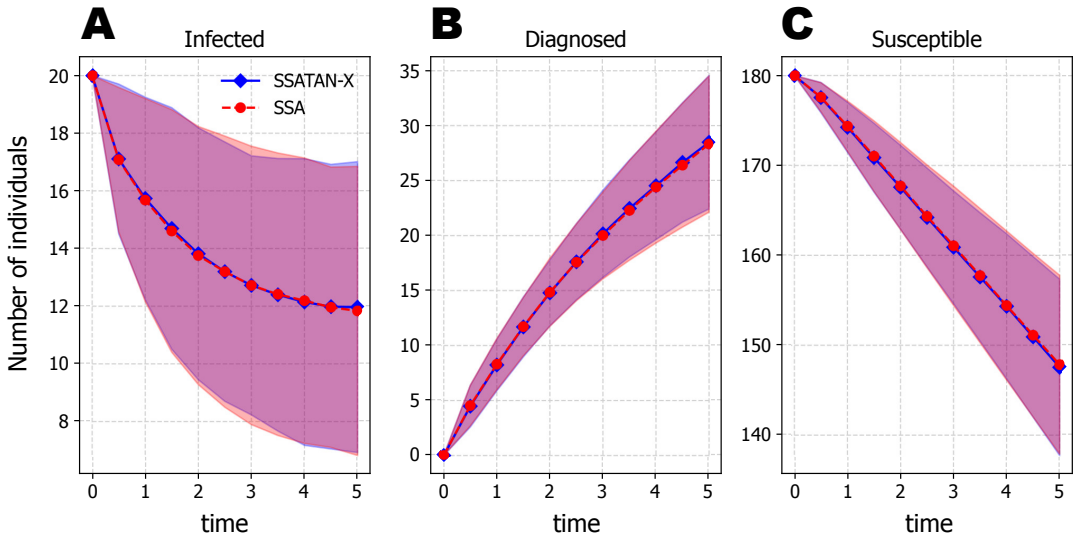


Fig. 5.4. Simulated infection dynamics. The graphic depicts the change in the number of infected (panel **A**), diagnosed (panel **B**), and susceptible (panel **C**) individuals over simulation time. Dashed red and solid blue lines describe the sample means over 10^3 simulations using SSA and SSATAN-X, respectively. Shaded areas represent the corresponding areas encompassed by the mean $\pm$ standard deviation.

Fig. 5.4 illustrates the corresponding simulation outcomes for the population state. The mean (with $\pm$ standard deviation) number of susceptible, infected, and diagnosed individuals is presented in Fig. 5.4A–C, respectively. Throughout the simulation, there are no discernible differences in both the mean and standard deviation when comparing the simulations conducted with the SSA (red line and red shading) and SSATAN-X (blue line and blue shading). This outcome evidences the accuracy of SSATAN-X in simulating the spreading dynamics on an adaptive contact network, solidifying its reliability and effectiveness.

5.2.3 Impact of contact network adaptivity on the spreading dynamics

The analysed S-I-D model belongs to the class of adaptive network models, where both the *contact dynamics* and the *populational dynamics* mutually influence each other (see Fig. 2.3C). Specifically, in this model, the event of diagnosis not only affects the contact network itself by severing all edges of the diagnosed individual, but also influences its future dynamics by reducing the rate of establishing new contacts to 30% of the pre-diagnosis level.

To investigate the impact of adaptivity on the *spreading* and *contact dynamics*, two representative examples of the contact network model were considered. In the first example, the diagnosis rate was set to $\delta = 0$, representing a non-adaptive temporal network model (described in Fig. 2.3B). In the second example, the diagnosis rate was set to $\delta = 0.5$,

reflecting an adaptive model. In both examples, the transmission rate γ_{jk} is set to equal 0.004 for all $I-S$ and $I-D$ edges, while the death rate β is set to 0, allowing the isolation of the impacts of adaptivity on the epidemic and on the contact dynamics from the impacts inflicted by other factors.

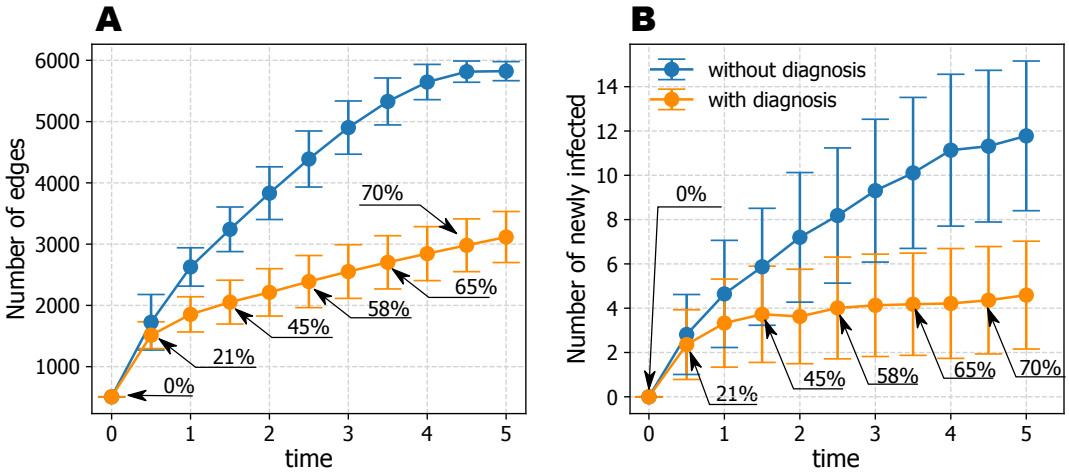


Fig. 5.5. Influence of diagnosis and the behavioural response on the degree distribution of contacts and of new infections. Blue error bars portray the results from a non-adaptive model, where we set the diagnosis rate $\delta = 0$. Orange bars depict predictions from an adaptive model, where the diagnosis rate $\delta = 0.5$. In the adaptive model, a diagnosis of individual j, leads to a loss of all contacts (self-isolation) and a reduced rate of forming new contacts $\lambda_j^+ = 0.3 \cdot \lambda_j^+$. Both panels depict results from 10^3 SSATAN-X simulations for each parameter set. **(A)** Error bars represent the mean amount $\pm$ standard deviation of the number of edges eligible for transmission of the virus (i.e., number of contacts between infected and susceptibles and between diagnosed and susceptibles) over time. **(B)** Error bars represent the mean amount $\pm$ standard deviation of newly infected individuals over time. Annotations on both panels show the percentage of infected individuals diagnosed (D)/(I + D).

Fig. 5.5A compares the number of edges in the network eligible for transmission in the non-adaptive temporal network ($\delta = 0$; blue error bars) versus the adaptive network ($\delta = 0.5$; orange error bars). The annotations indicate the percentage of infectious individuals diagnosed (e.g., $N_D/(N_D + N_I)$). Fig. 5.5B compares the infection dynamics between the non-adaptive and adaptive models (blue vs. orange error bars).

Evidently, the diagnosis event alters the *spreading dynamics* by reducing the occurrence of new infections in the adaptive network. This decrease in infection dynamics can be attributed solely to changes in the *contact dynamics* within the adaptive networks (Fig.5.5A). As the percentage of infected individuals being diagnosed increases, the number of edges eligible for pathogen transmission decreases. Consequently, there are fewer opportunities for disease spread, decreasing the number of new infections (orange error bar in Fig. 5.5B).

5.2.4 Runtime performance

The runtime comparison between SSATAN-X and SSA for different population sizes (number of nodes) was analysed, with simulations starting with a fixed number of infected individuals ($N_I = 20$). The simulation results, as shown in Fig. 5.6A, indicate that the runtime increases as the population size grows. However, for the parameter setting used, SSATAN-X exhibits a runtime approximately 100 times shorter than that of SSA.

Furthermore, the runtime performance was assessed as a function of the specific contact dynamics simulation parameters. In this analysis, the homogeneous contact dynamics rates λ_i^+ and λ_i^- were set uniformly across the population, meaning all individuals had the same rates. This setting allowed for an evaluation of how the runtime changes depend on the probability of transmission occurring on a contact edge before it dissolves. This probability is determined by $\gamma/(\gamma+(\lambda^-)^2)$, where γ represents the transmission rate on a contact edge, and $(\lambda^-)^2$ denotes the rate of losing an existing edge. In this analysis, the λ^+ and λ^- were increased proportionally, maintaining the same average number of contacts but shortening their lifetimes. The value of the transmission rate γ remained unchanged.

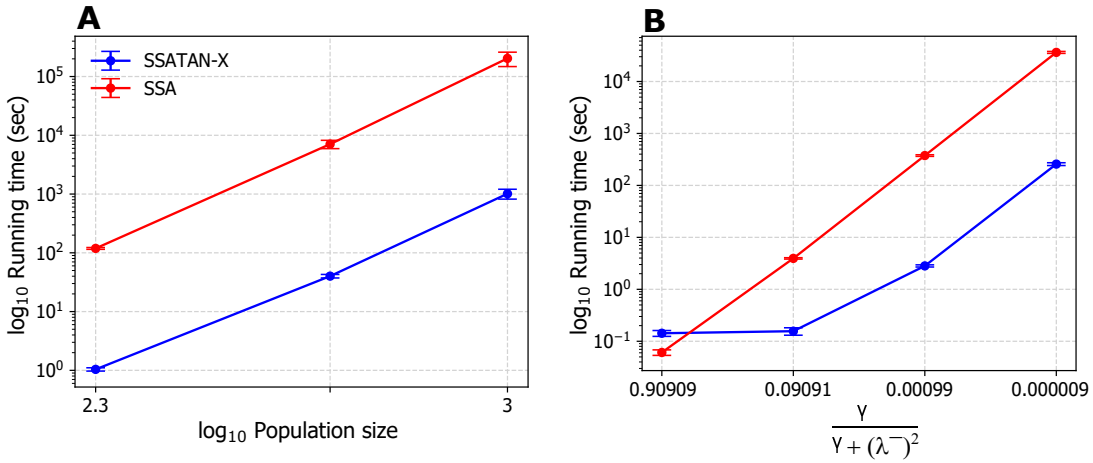


Fig. 5.6. Mean runtime comparison Between SSATAN-X and SSA. (A) Log-log plot showing the mean ($\pm$ standard deviation) runtime of SSATAN-X (blue error bars) vs. SSA (red error bars) for different population sizes (200, 500 and 1000 individuals) over 10^3 simulations for each parameter set on a single `2x AMD Epyc 7742 64-Core` compute node with a base clock of 2.25 GHz, respectively. **(B)** Semi-log plot showing mean ($\pm$ standard deviation) runtime of SSATAN-X (blue error bars) vs. SSA (red error bars) for different parameterisations for a network with 200 individuals. The relation $\gamma/(\gamma+(\lambda^-)^2)$, with γ being the transmission rate and λ^- being the rate of edge disassembly, can be interpreted as the probability of transmitting an infection before the contact dissolves. The plot is presented for the following $(\lambda^+;\lambda^-)$ combinations: $\{(0.025,0.02),(0.25,0.2),(2.5,2.0),(25.0,20.0)\}$.

The runtime of both SSA and SSATAN-X for conducting 10^3 simulations for each parameter set is depicted in Fig. 5.6B. The figure makes evident when the ratio $\gamma/(\gamma+(\lambda^-)^2)$ approaches 1, indicating that transmission always occurs before an edge dissolves, SSA might be marginally faster than SSATAN-X. This variation arises because the time interval between two population dynamics events becomes extremely small, causing the network update algorithm to essentially reduce to SSA. Such models may not require separate modelling of the contact dynamics at all, rendering this example somewhat artificial (further discussion on the usefulness of different modelling approaches can be found in section 5.2.5 and in chapter 7).

When the ratio $\gamma/(\gamma+(\lambda^-)^2)$ decreases, indicating that the contact dynamics are much faster than the spreading dynamics, the runtime of SSATAN-X becomes superior to that of SSA. Therefore, with low transmission rates and fast contact dynamics, SSATAN-X avoids computational overheads associated with contact dynamics simulation by utilising an approximate contacts update approach, considering only their *net* effect on spreading dynamics. In contrast, SSA requires the execution of each contact and population reaction individually.

As a conclusion, SSATAN-X offers computational efficiency advantages over SSA. Both algorithms exhibit runtime scaling with population size, and the "boost factor" of SSATAN-X remains consistent across different population sizes. However, the computational advantage of SSATAN-X becomes more pronounced when the *contact dynamics* are faster compared to the *population dynamics*.

5.2.5 Advantage for models with fast contact and slow spreading dynamics

The efficiency of the SSATAN-X algorithm in hastening computations significantly hinges on the relationship between the varying speeds of contact dynamics and transmission dynamics. If contacts rarely lead to transmission, SSATAN-X can greatly increase computational speed. However, if transmission almost always happens during a contact, the speed increase will be less significant or non-existent.

Fig. 5.6B shows this relationship using the formula $\gamma/(\gamma+(\lambda^-)^2)$. Here, $(\lambda^-)^2$ represents the rate at which a contact ends, assuming that the rate of losing a contact, denoted by λ^-, is the same for all involved parties. The parameter γ refers to the rate at which transmission occurs during a contact.

The transmission rate γ is a comprehensive parameter that can be broken down for more detailed modelling and realistic parameter setting. For instance, if exposure at a contact

happens at rate r_e and the exposures are of the same type, then the transmission rate γ is simply a scaled version of the exposure rate, calculated as $\gamma = r_e \cdot p_{\mathrm{tr}|e}$. Here, $p_{\mathrm{tr}|e}$ is the probability of transmission per exposure event e. This probability is known for many infectious diseases [131, 132] and exposure types, and the impact of both pharmaceutical and non-pharmaceutical interventions on $p_{\mathrm{tr}|e}$ can be quantified [133–135].

The combined transmission rate γ is determined by both the probability of transmission per exposure $p_{\mathrm{tr}|e}$ and the frequency of exposures r_e when individuals are in contact. For many diseases and contact types, the ratio $\gamma/(\gamma + (\lambda^-)^2)$ is likely to be much less than one, meaning most brief contacts do not lead to pathogen transmission. Behavioural changes like social distancing, isolation, or quarantine can further reduce the transmission rate by decreasing r_e. Similarly, pharmaceutical interventions (vaccination, prophylaxis, treatment) and non-pharmaceutical measures (e.g., masks for airborne diseases, condoms for SDIs) can lower the probability of transmission per exposure, either in a static or time-dependent manner [136–140]. This illustration further reinforces that the computational benefits of SSATAN-X are likely to persist when applied to realistic models.

5.3 Summary

This chapter analyses simulation results using a hybrid algorithm for temporal adaptive networks. It confirms the algorithm's accuracy in simulating contact dynamics and its exactness in simulating spreading dynamics. It also highlights the impact of network adaptivity on these processes. The chapter evaluates the computational efficiency of the algorithm, especially in scenarios with faster contact dynamics than spreading dynamics, relevant to a large group of epidemiological diseases. This analysis establishes the algorithm's practicality and robustness, highlighting its potential for real-world applications in simulating spreading processes on complex networks.

Chapter 6

adaptiveSpreadX: C++ application and Python package

6.1 Introduction

This chapter presents *adaptiveSpreadx*, the software package implementing the discussed algorithm, rendering it accessible for practical application. This software package consists of two main components: a core part written in C++ and a Python wrapper. The C++ core encompasses all the algorithm functionality, ensuring efficient execution of its operations. Complementing this, the Python wrapper enables users to conduct simulations within a Python environment and to visualise results, providing an accessible interface to interact with the algorithm. The package is available at the link https://github.com/nmalysheva/adaptiveSpreadX/.

This chapter delves deeper into the rationale behind the choice of programming languages. It also introduces the core components of the software, elucidates the intricacies of the configuration file, and discusses the limitations inherent in the current iteration of adaptiveSpreadX.

6.2 Implementation and contribution

In the initial phase of this project, the first version of the C++ implementation was developed and published by the author of this dissertation [121]. This version included the core functionalities of the algorithm and laid the foundation for further improvements.

Subsequently, another programmer joined the project to enhance the C++ implementation. Their contributions included optimizing the code for better performance, fixing bugs, and

adding new features. These enhancements significantly improved the software's overall efficiency and reliability. Further details can be found in the Contributors section of the repository linked above.

To increase the accessibility and usability of the software package, the author developed a Python wrapper around the C++ implementation. This wrapper allows users to interact with the algorithm through Python, facilitating easier integration with other Python-based tools and workflows. The development process involved addressing compatibility and ensuring seamless integration between the C++ core and the Python interface.

6.3 Programming language

In deciding on a suitable programming language for this scientific project, the options between Python and C++ were evaluated. Both Python and C++ hold prominent positions in scientific computing, each catering to distinct needs and scenarios. adaptiveSpreadX project, characterised by complex computations and large datasets, necessitates a language where execution speed is paramount, thereby inclining the decision towards C++.

C++ stands out for its notable performance and generally exhibits faster execution compared to Python. It offers more direct control over memory management, a critical factor for optimising performance in resource-intensive scientific applications. As a statically typed language, C++ catches many errors at compile-time, an attribute invaluable to ensure the correctness of scientific applications. Moreover, its ease of integration with various languages and technologies is pivotal for utilising existing software components or interfacing with other systems. The deterministic object lifecycle in C++, featuring constructors and destructors, is instrumental for resource management in prolonged scientific simulations and experiments.

Given these considerations, the decision was made to employ C++ for the performance-critical segments of the project, specifically the direct implementation of the algorithm. Subsequently, the core algorithm module was integrated into Python module for results processing and visualisation.

The choice of the C++17 standard was influenced by several factors. C++17 introduced a host of beneficial features not present in its predecessors, such as `std::optional`, which proved advantageous for our project. It also brought about optimisations and enhancements, promising improved performance of the generated code. At the juncture of the decision-making, full support for C++20 might not have been universally available across all compilers or development tools, rendering C++17 a more reliable choice for widespread compatibility and support. Furthermore, the features introduced in C++20 were assessed as

non-essential for the project requirements, making C++17 a fitting and sufficient choice for our developmental needs.

In the subsequent development phase of the Python wrapper, the commitment to Python 3 was cemented for several poignant reasons, consciously sidestepping the use of Python 2. Primarily, Python 2 was officially decommissioned as of 1 January 2020, and ceased to receive updates, including those critical for security. This lack of support not only raises potential security concerns but also hints at likely compatibility issues, as modern software libraries and tools progressively withdraw support for Python 2, highlighting Python 3 as the secure and long-term alternative for the project.

6.4 C++ Core: Code Structure and Design

The implementation of the *adaptiveSpreadX* is decomposes into the following modules:

1. **Configuration module.** Extraction and semantic interpretation of model parameter settings provided in the configuration file.
2. **Graph module.** Fundamental implementation of an undirected graph, independently of external graph libraries.
3. **Contact Network module.** Designed to represent a population, capturing its intricate contact network as well as interaction rules of the simulated model.
4. **Core Algorithm module.** Implements the main logic of the algorithm.
5. **Types** and **Utils modules.** Contain types definition, helper functions, and utilities.

6.4.1 Configuration module

The configuration module assumes responsibility for processing either a configuration file or stream, subsequently interpreting it into settings that govern the algorithm, as well as the modelled population and its contact network changes (interactions, transmissions, etc.). The principal functionality resides in the `Configuration` class, encompasses parsing settings from the file or stream, verifying their validity, and also affording the capability to serialise configurations into the `.json` format. Serialisation into the `.json` format ensures that configurations are easily transferable, storable, and retrievable in a standardised format. Meanwhile, ancillary classes like `Exception`, `Helper`, and `Stream` are employed, each encapsulating their respective functionalities. A single `Configuration` instance can be created by calling,

```
auto file = std::ifstream{filename};
auto const config = configuration::Configuration{file};
```

6.4.2 Graph module

Employing a simple undirected graph to represent the contact network amid the population fundamentally was principally guided by its inherent simplicity and straightforwardness in illustrating mutual interactions, thereby facilitating an uncomplicated yet effective, model for exploring and analysing contact dynamics.

While the current framework utilises a simple undirected graph for its inherent simplicity, future developments and expansions of the project may warrant the introduction of directed graphs or even mixed graphs, comprising both directed and undirected edges, to accommodate more complex interaction dynamics.

While the preceding implementation of the SSATAN-X algorithm (https://github.com/nmalysheva/SSATAN-X) utilised the external LEMON library (https://lemon.cs.elte.hu/trac/lemon) for graph representation, the decision to implement a simple graph without employing extant libraries such as LEMON or Boost is informed by several strategic considerations. Primarily, the functionality requisite for the current project is relatively trivial, negating the necessity for the comprehensive (and oftentimes, complex) feature sets offered by such libraries. Moreover, issues such as discontinued support could challenge the future sustainability and usability of the project. Nevertheless, contributors do not preclude the potential incorporation of an external library for graph representation, should future project development necessitate a more sophisticated structure and functionality.

The Graph module facilitates the structured storage of the contact network using an adjacency list representation. Its capabilities encompass the addition and deletion of nodes and the insertion and removal of edges within the graph.

The code architecture for this module employs the Bridge Pattern [141], in which the abstract class `IGraph` stands as a pure abstract class, or interface, serving as an abstraction for a graph. This establishes an interface that all concrete graph implementations must follow, thus delineating a family of algorithms and encapsulating each as an individual object.

The `GraphImpl` class, a concrete entity, inherits from `IGraph` and furnishes a specific graph implementation. While it utilises `IGraph` as its interface, the actual implementation remains distinct and can independently vary from the interface.

Leveraging this design pattern enables future implementation switches, for instance, adopting an external library, without disrupting the project's functionality.

6.4.3 Contact Network module

The core class of this module, `ContactNetwork`, encapsulates a simulated population, including its contact network and the inherent dynamics, interaction protocols, and transmission guidelines integral to the model. All necessary settings pertinent to the population and spreading dynamics, such as rates of contact changes, birth and death rates, and transition rates and distributions, are managed by the `Settings` class. The network consists of nodes, portrayed by the `Individual` structure, whose interconnections (edges) are administered using `Graph<GraphImpl>` class, in alignment with the network dynamic rules outlined in `Settings`. Additionally, the `ContactNetwork` manages various events and transitions occurring on the network, including births, deaths, interactions, and state transitions of nodes, while facilitating necessary adaptation changes.

The structure (or simply struct) `Individual` represents a member of the population. It contains information about the individual's state, the last modification time (the last time it changed its state), contact rates (indicating how likely this individual is to establish a new contact or terminate an existing one), and various rates influencing the state transitions of this specific individual.

The `IndividualFactory` class adeptly encapsulates the creation of population individuals, serving as a distinct entity responsible for generating instances of the `Individual` struct. That encapsulation defines the Factory Method pattern [141] and provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created. This segregation proves immensely useful in isolating object creation logic from client code, thereby enhancing code reusability and maintainability. Should the creation logic of `Individual` instances require modifications or expansions in the future – for example, introducing additional parameters or varying initialisation processes – such alterations can be gracefully implemented within `IndividualFactory` without affecting other parts of the code. This consolidation streamlines code maintenance and future modifications.

The class `IntractionManager` oversees all interactions among individuals within a network, serving as a centralised management point. This encapsulation enables it to systematically coordinate interactions, abstracting and addressing the associated complexity while offering simplified interfaces.

Node- and edge-management methods create, remove, and modify nodes and edges within the `ContactNetwork` class:

- `create(simulation_time, state)` – create a new node of a given state and store the simulation timestamp;
- `remove(node)` – remove a node from the network;

- `change(simulation_time, node, to_state)` – change the state of a given node and store the simulation timestamp. Changing the state of a node consists of several steps:
  - changing the state of the node and updates its modification time,
  - updating possible interactions, and
  - performing adaption rules if needed;
- `create_edge(node_from, node_to)` – create a new edge between given nodes; and
- `delete_edge(node_from, node_to)` – delete an edge between given nodes.

Other methods retrieve rates of various events and transitions occurring within the network:

- `get_edge_deletion_rates()` – returns a collection of two connected nodes and a rate at which a connection will be removed. The rate is the product of the individual's contact removal rates;
- `get_edge_creation_rates()` – returns a collection of two unconnected nodes and a rate at which a connection will happen. The rate is the product of the individual's connection rates;
- `get_deaths()` – return a collection of nodes and their death rates;
- `get_births()` – return a birth rate;
- `get_transitions()` – returns the collection of triplets "<rate, Node, State>", where with the given rate, the Node will change to State; and
- `get_interactions()` – returns the collection of triplets "<rate, NodeA, NodeB>", where with the given rate, NodeA will interact with NodeB. Interaction here assumes the reaction between two nodes (transmission)

`.json` serialisation method:

- `to_json()` – serialises the network to the `.json` representation.

An instance of the `ContactNetwork` can be created by calling the initialising constructor:

```
auto network = network::ContactNetwork{config};
```

where `config` is a `Configuration` instance initialised before in 6.4.1.
Explicitly deleted copy constructor and copy assignment operator ensure a single instance of `ContactNetwork` per run.

6.4.4 Core Algorithm module

This module encapsulates the core implementation of the algorithms, introducing three pivotal components within a network-based simulation framework: `Algorithm`, `SSATAN-X`, and `SSA`. Each of these classes is geared towards establishing a framework for implementing specific network algorithms that comply with given settings and operate on defined contact networks. The module additionally includes the class `Settings`, tasked with managing pertinent algorithm-related settings, including simulation time and algorithm design parameters. The base class `Algorithm` serves as a base for specific algorithm implementations. The design leverages the strategy pattern, where different algorithms can be used interchangeably, based on the provided settings and network context. It defines the following key responsibilities of all specific algorithms that are intended to be implemented based on it:

- `initialization` – configures the algorithm using the provided settings and network, offering a common setup for derivatives;
- `run` – pure virtual function requires descendants to define their own algorithm execution logic;
- `to_json` – converts the algorithm's data into a `.json` structure, facilitating easy logging or data extraction.

The derived from `Algorithm`, class `SSATAN-X` implements the specific SSATAN-X Algorithm 8 in the method `run`. It oversees the simulation execution, ensuring compliance with established parameters and conditions. This method also documents the simulation data in a provided `.json` object.

The class `SSA`, another derivative of `Algorithm` class, encapsulates the implementation logic for the SSA Algorithm 1 in the method `run`. This method also allows to log a resultant data in a `.json` object. This class was implemented primarily for benchmarking purposes. Although it generates correct results, using the SSA algorithm for models with a population size over 100 is not recommended, nor is it advisable for models with significantly fast dynamics due to its slow-running performance. For more information, please see https://github.com/nmalysheva/adaptiveSpreadX/tree/cpp_package.

The factory method, `make_algorithm`, is employed to create and initialise the appropriate algorithm (SSA or SSATAN-X) based on the provided settings, illustrating a use of the factory pattern [141]. This pattern allows for the encapsulation of the logic that determines which specific algorithm implementation to use, providing a straightforward API for client code and ensuring that it does not need to manage the instantiation logic directly.

The base `Algorithm` assumes that the resultant data is collected and stored in the `.json` format throughout the entirety of the programme execution, which may lead to substantial memory usage. Thus, memory utilisation and management might be a critical consideration in the practical deployment of these algorithms.

Both `SSATAN-X` and `SSA` assume that the `Settings` class contains valid data, indicating that data validation is considered a prerequisite for these algorithms.

The module also features a τ-leap sub-component of the SSATAN-X algorithm, encapsulated within an autonomous class. While it does not inherit from the `Algorithm` class, it can be independently invoked, if desired, by supplying the `ContactNetwork` instance, simulation parameter epsilon, and specified simulation time constraints:

```
auto tau_leap = TauLeap{network, epsilon};
tau_leap.execute(start_time, end_time);
```

The ancillary `Action` class encapsulates all potential reactions relevant to the simulation, while the `ActionsFactory` class facilitates expedited access to individual or grouped `Actions` (whether spreading, contact, or both). Additionally, it enables the recycling of memory allocated by all `Actions` instances.

The recommended approach for invoking an algorithm instance is as follows:

```
auto algo = algorithm::make_algorithm(config, network);
algo->run(json_string);
```

Here, `config` and `network` are previously initialised instances of `Configuration` and `ContactNetwork`, respectively. Additionally, the `json_string` is initialised before `.json` object employed to collect the resultant data.

6.4.5 Types and Utility modules

These modules, which encompass descriptions of user types and assorted functions like generating random numbers and crafting `.json` strings, are utilised throughout various locations and modules within the project. Furthermore, they are designed to be reusable, serving not only different components within the current project but also possessing the potential to be integrated into other projects.

6.4.6 Testing and validation

All core modules of the C++ project are subjected to rigorous testing to ensure their individual performance and reliability in isolation from the entire system. The core of the testing routine

is the strategic implementation of unit-tests. The essence of utilising unit testing lies in its ability to validate the precise functionality of each module—spanning functions, classes, and more—under a variety of conditions and inputs. This method involves testing each component independently, significantly enhancing the robustness of the application by enabling accurate localisation and efficient troubleshooting of potential issues and bugs. Moreover, unit-testing techniques serve a dual purpose: they not only validate the correct functionality of each system component, but also are crucial to maintaining code quality and ensuring smooth integration throughout the development process.

6.5 Python wrapper

The linkage between C++ functionality and Python is achieved through the use of the `pybind11` library (https://github.com/pybind/pybind11). As a well-regarded open-source tool, `pybind11` facilitates the binding that enables smooth interoperability between C++ and Python code. Designed to be a lightweight, header-only library, `pybind11` allows developers to seamlessly expose C++ classes and functions to the Python language, thereby enabling C++ code to be invoked as if it were native Python code.

After compiling the C++ code with `pybind11` and installation of the Python module, it can be used in Python as follows:

```
import adaptiveSpreadX as asx

# Create a Configuration object using a file name
config = asx.Configuration("config_file_path")

# Create a Network object using the Configuration object
network = asx.Network(config)

# Simulate and get the JSON results
results = asx.run(config, network).
```

The `result` variable returned after the simulation is the collection of the contact, epidemiological and populational states of the simulated system throughout the simulation time.

Furthermore, the visualization of results implemented on the Python side enables the generation of several key plots. By invoking the function

```
asx.plot_species_count_by_state(result,
```

```
    timesteps=timestepsCompare, states_list=["I", "D"])
plt.show()
```

it is possible to visualise the number of individuals in each state from the `states_list` at specified time points defined in `timesteps`. An example of this plot can be seen in Fig. 6.1A.

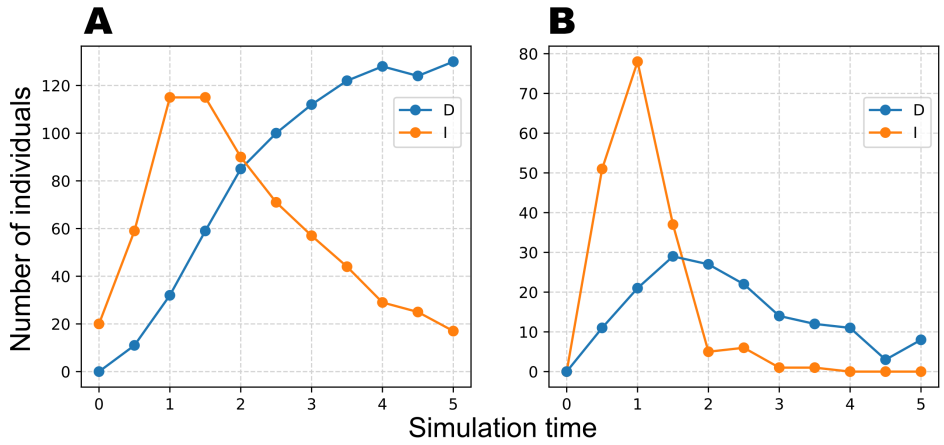


Fig. 6.1. adaptiveSpreadX one trajectory visualisation example. A single trajectory of the model similar to the one outlined in Chapter 5. Plot **(A)** illustrates the population dynamics of individuals within the "I" and "D" compartments throughout the simulation. Plot **(B)** depicts the number of individuals transitioning to states "I" or "D" at specific intervals throughout the simulation. For this instance, the simulation duration was set to be 5 units, with visualisations generated at time points from 0 to 5 with the step size of 0.5 units

Another significant plot illustrates the number of individuals who transitioned to one of the states listed in `states_list` between the time points specified by `timesteps`. This plot can be generated by invoking the function

```
asx.plot_new_state_changes(result,
    timesteps=timestepsCompare, states_list=["I", "D"])
```

An example of this plot can be seen in Fig. 6.1B. Both functions accept a single `results` outcome, thus visualising only one trajectory. However, to formulate a reliable prediction, it is generally necessary to simulate and analyse not just one, but dozens, hundreds, or even thousands of trajectories. AdaptiveSpreadX facilitates parallel simulations through the use of the multiprocessing library. This capability can be used by invoking the function below:

```
res_list = asx.run_parallel(n_processes, n_simulations,
                                        config_fname)
```

Here, `n_processes` represents the desired number of processes to be employed. For comprehensive details, please refer to the documentation of the `multiprocessing` library. The parameter `n_simulations` specifies the number of trajectories to be simulated, while `config_fname` is the path to the configuration file as previously described. Currently, it is not possible to directly pass `Network` and `Configuration` entities as arguments of that function, owing to the complexities involved in serialisation and deserialisation processes. Instead, the path to the configuration file is provided for each execution, enabling every process to independently initialise the `Network` and `Configuration` entities. Addressing this limitation is on the agenda for future development by the developers.

After simulating multiple trajectories, adaptiveSpreadX offers a function to depict the average dynamics of each state from the `states_list`. This can be achieved by invoking the following function:

```
asx.plot_mean_dynamics(res_list, timesteps=timestepsCompare)
```

An example of the plot illustrating these mean dynamics is showcased in Fig 6.2

6.6 Functionality and settings

Both C++ core and Python wrapper components necessitate a configuration file to be provided to operate correctly. This file, a simple text document, contains various settings relevant to the simulated model and algorithm, organised into different sections to simplify the navigation and modification. This section outlines the configuration file's structure and details the settings that can be adjusted through this file.

Each section within the configuration files ought to commence with a section header, formatted as "[section]". Subsequent lines are interpreted as settings pertinent to that particular section. It is important to remember each section header should be unique, appearing no more than once, and refrain from incorporating whitespaces. Each setting in the configuration file is a line of text, with segments separated by whitespace. These segments can be either numerical values or sequences of charecters (words), depending on the format specified by the particular section. Typically, the first segment in a line specifies the name of the parameter, while the subsequent segments represent the values assigned to that specific setting parameter (e.g., line " time 5" specifies the simulation time length of 5).
The following sections are *obligatory* for each simulation:
[Algorithm]. Sets up the algorithm settings. Required parameters for this configuration section are following:

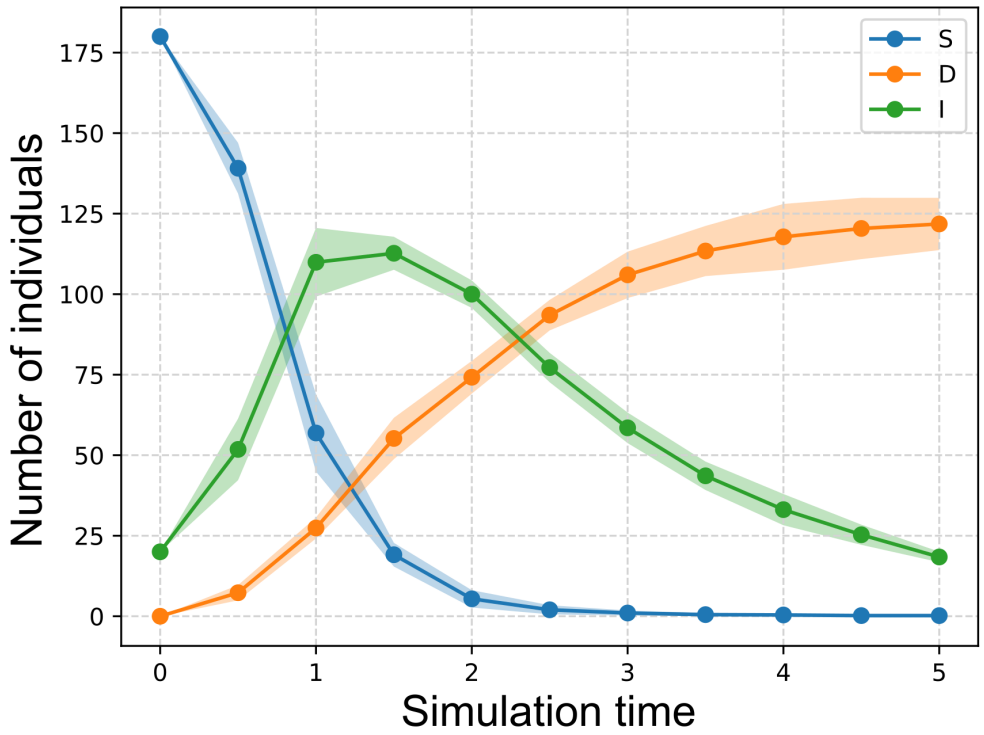


Fig. 6.2. adaptiveSpreadX mean trajectories example. Mean population dynamics for the "I", "D" and "S" states of the model, similar to the one discussed in Chapter 5, are depicted. The mean plot is for 10 simulated trajectories. This mean dynamics plot is based on 10 simulated trajectories. Solid lines illustrate the count of individuals in each compartment, while the shaded areas denote the standard deviation around these means.

- `use` – specifies the algorithm to be utilised, where options include SSA or SSATAN-X; and
- `time` – defines the duration of the simulation.

It is also possible to set up following optional parameters for this section:

- `output` – saves to the result the state of the network at every 10^n steps. In the absence of this parameter, only the initial and final states are saved; and
- `epsilon` – a design parameter utilised in the τ-leap algorithm, intrinsic to SSATAN-X. If omitted, a default value of 0.03 is applied. Note that this parameter is inconsequential if the SSA is chosen as in `use` setting.

Example of the algorithm settings:

```
[Algorithm]
# use SSATAN–X algorithm
use SSATAN–X
# for 100 time units
time 100
# output every step
output 0
# with custom epsilon
epsilon 0.25
```

[Network]. Sets up the initial population parameters and its contact network rules. Required parameters for this configuration section are following:

- `State_1 number_1 ... State_m number_m` – this syntax specifies the states used and the number of nodes in the initial network. Every state utilised in the simulation must be delineated here. If no initial node of a particular state is desired, it should indicate so with a 0; and
- `edges` – dictates the quantity of randomly generated edges. Absence of this parameter results in no edge creation, whereas a number larger than the maximum potential edges implies complete node connectivity.

Additionally, the following optional parameter can be configured for this section:

- `seed` – employed as a seed for the random number generator when creating the initial set of edges. If not specified, a random number is utilised as the seed.

Example of the contact network configuration settings:

```
[Network]
# create 5 nodes with state A
A 5
# and 1 B
B 1
# no C, but needed for transitions
C 0
# and randomly connect 2 nodes
edges 1
```

The following sections are *optional* for each simulation:
[Birth] and **[Death]**. This section establishes the rules for the creation and removal of nodes.

- `State_1 rate_1 ... State_m rate_m` – regulates the creation or deletion of a node of state $State_i$ with the rate $rate_i$. The rate is individually determined for each node upon its creation, ensuring a distinct rate allocation for every single node in the network.

Example of the [Birth] and [Death] configuration settings:

```
[Births]
# create S with a fixed rate
S 0.1
# and I with a rate drawn uniformly
I U(0.2,0.4)

[Deaths]
# remove only nodes of state I with a fixed rate
I 0.9
```

Importantly, the states used in this section must be present in section [Network]. If section [Birth] or [Death] does not presented in the configuration file, these types of reactions will be excluded from simulation.
[AddEdges] and **[RemoveEdges]**. Rules for creating and removing edges.

- `State_1 rate_1 ... State_m rate_m` – controls the rate at which nodes of each state establish or dissolve their connections. Notably, the rate is individually ascertained for each node at the time of its creation.

The rate of the action – whether a deletion or addition of contact between two nodes denoted as a and b – is defined by $rate_a * rate_b$. This formulation ensures that the interaction between two distinct nodes is influenced by the characteristic rates of both involved entities.

If section [AddEdges] or [RemoveEdges] does not presented in the configuration file, these types of reactions will be excluded from contact dynamics during simulation.
Example of the configuration settings for adding and removing edges:

```
[AddEdges]
# S creates edges on a high rate
S 0.8
# I with a low rate
```

```
I 0.1

[RemoveEdges]
# S keeps its contacts, so not listed here
# but I drops connections at a high rate
I 0.9
```

[Transitions]. This section dictates the rules for altering the state of a node, independent of its connections.

- `from_state to_state rate` – a node initially in the `from_state` transitions to the `to_state` at a specified `rate`. The rate is individually determined for each node upon its creation.

Example of the transition settings:

```
[Transitions]
# A turn to B at rate 0.5
A B 0.5

# B turns to A with rate drawn uniformly from range [0, 1]
B A U(0,1)
```

Avoiding [Transitions] section in the configuration file leads to the excluded of these kind of reactions from simulation.

[Interactions]. This section details the rules for altering the state of a node based on its connections.

- `from_state contact_with to_state rate` – if a node in the `from_state` has a connection with another node in the `contact_with` state, it possesses the capability to alter its state to the `to_state` at a specified `rate`. The rate is individually determined for each connection (edge) at the time of its creation.

Example of the interactions configuration:

```
[Interactions]
# If A connected to B, make it C at a rate 0.1
A B C 0.5
# If B is connected to A, make it C at a rate drawn
#uniformly from the range [0.1,0.2]
B A C U(0.1,0.2)
```

If the [Interactions] section is omitted from the configuration file, contact-dependent reactions will not be considered in the simulation.
[Adaptations]. This section delineates rules for the adaptation of a node or its neighbourhood when its state undergoes a change. Two kinds of adaptations are available:

- *Removal of all edges connected to the node.* Utilising the syntax `state part` to signify that when a node transitions to the `state`, `part` *100% of its contacts will be removed. If `part=0`, this adaptation rule is bypassed or ignored. Example of this type of adaptivity:

```
[Adaptaions]
A v B
A v B C D E...
```

- *Alteration of the state of neighbouring nodes*. Employing the syntax `A part B <C D E>` indicating that if any node switches its state to A, `part` *100% of its neighbours will adjust their state to B. If additional states are provided, they are utilised as a filter. Consequently, the adaptation applies only to nodes with the state C, D, or E. Example of this adaptivity type:

```
[Adaptaions]
# If turned to A turn all neighbours to B
A 1.0 B
# If turned to A turn 50% of the neighbours to B,
#if the current state is C or D
A 0.5 B C D
# Ignored: changing a node to its current state
A 0.5 B B
# The "filter" C has no effect and will be ignored
A 0.1 C B C D
# The above rule will be interpreted as
A 0.1 C B D
```

Of all rules, only one will happen (e.g., either remove edges OR change neighbours). If neighbours change and adaption rules can apply, the execution will also performed.

Omitting the [Adaptivity] section from the configuration file means that adaptive behavior will not be included in the simulation, thereby simplifying the model and removing its adaptivity.

For a more detailed description and additional information, refer to the documentation(https://github.com/nmalysheva/adaptiveSpreadX/blob/cpp_package/README.md).

6.7 Limitations

While the software package *adaptiveSpreadX* facilitates the simulation of spreading processes on time-evolving adaptive networks, enabling quick and efficient simulations, it is still in its early stages and consequently has a number of limitations. The software adheres rigorously to a specific contact model in which the rate of gaining or losing a contact is determined by the multiplication of individual rates. For instance, for two nodes i and j with rates of forming a new contact λ_i^+ and λ_j^+ respectively, the rate of forming a contact between them is defined as $\lambda_{ij}^+ = \lambda_i^+ \cdot \lambda_j^+$. This approach may constrain the software's flexibility and applicability, particularly in scenarios requiring different contact dynamics or models.

During the initial formation of the contact network, edges are created randomly and subsequently governed by agent parameters, which may be insufficient for simulations that demand a particular initial edge pattern or network topology. This limitation could affect the replicability or accuracy of certain network models that rely on specific initial conditions.

The software lacks the capability for dynamic reduction, which may impede its ability to efficiently manage large-scale simulations or those that require substantial computational resources. This limitation could be especially decisive when dealing with intricate or extensive networks where computational efficiency is paramount.

The requisite of employing a configuration file, instead of allowing model setup directly within the Python environment, introduces an additional step, potentially complicating the process and hindering the software's ease of use and swift implementation.

The parallel simulations for the trajectories is implemented using the `multiprocessing` library. This approach necessitates specific conditions for the `Network` and `Configuration` entities to be transmitted as arguments within the parallel framework. Specifically, the serialisation and deserialisation of these objects must be facilitated, a functionality not currently available but intended for future development.

Lastly, for various reasons, not all features of the C++ core are accessible in Python. While some functionalities, like network edge manipulation, are excluded because they are solely invoked by the algorithm, others require implementation and may be added in future updates upon user request.

All identified limitations are scheduled to be resolved in order of priority during the future development iterations of the software.All future developer plans, feature updates, and

functionality extensions can be found at https://github.com/nmalysheva/adaptiveSpreadX/issues.

6.8 Summary

This chapter discusses the *adaptiveSpreadX* software package, which implements the SSATAN-X algorithm for practical application. This package comprises two main components: a core part written in C++ for efficient execution and a Python wrapper for easy interaction and visualisation. The chapter elaborates the reason for choosing C++ and Python, the programming languages for the software implementation, the configuration file structure, and current limitations of the software. It details the structural design of the C++ core, including modules like Configuration, Graph, Contact Network, and Algorithm. Each module's functionality and design are explained. The linking C++ functionality with Python using pybind11 is also described. Finally, the chapter outlines the package's limitations and potential areas for future development.

Chapter 7

Discussion and outlook

7.1 Contribution

This work has introduced *SSATAN-X*, an algorithm designed to simulate effective spreading dynamics on adaptive time-evolving networks with efficiency and precision. This algorithm capitalises on the observation that, in models where both contact dynamics and epidemic dynamics are significant, the former tends to occur at a faster pace than the latter (i.e., not every contact leads to transmission). The algorithm employs a focussed strategy, concentrating primarily on the impact of contact dynamics on epidemic trends and vice versa, rather than delving into the sophisticated details of the contact dynamics itself. This can be accomplished through approximate mass-updates of the contact structure between major spreading and vital events, including pathogen transmission, diagnosis, or death. By employing this approach, SSATAN-X achieves computational efficiency while capturing the essential interplay between contact patterns and the spread of the epidemic. This allows for advances in the understanding of dynamic phenomena transmission within an adaptive time-evolving networks framework. Although various algorithms have been developed in recent decades to simulate different kinds of complex dynamics and contact networks, there remains substantial scope for refinement and advancement in this area of research.

In chapter 5, it was demonstrated that the proposed contact-updating method preserves the statistical properties of the underlying contact network, particularly at time points pertinent to the spreading dynamics. This preservation of the contact network statistical characteristics, notably at times crucial to the spreading dynamics, also maintains the statistics associated with triggering the subsequent spreading event. Additionally, the mass contact updating technique circumvents computational burdens originating from updating contact edges individually. Hence, depending on the parameter selection, SSATAN-X can notably accelerate computation times compared to exact methods like the SSA, all without the significant loss of simulation

precision. The acceleration factor is crucially dependent on the relationship between the *contacts dynamics* and the *population, (or spreading) dynamics*. For instance, if contacts generally do not result in transmission, the computational acceleration with SSATAN-X will be significant, as shown in chapter 5.

7.2 Limitations and recommendations

The enhanced computational performance offered by the SSATAN-X is significant for a majority of real-life-scenario models. However, this efficiency is subject to certain constraints, which are explored in detail in this section. Furthermore, the relevance of some outcomes from this research is predominantly within specific parameter contexts, an aspect that is also examined in this section.

7.2.1 Time-dependant reaction rates

In the example provided in chapter 5, a simplistic model is utilised to demonstrate the functionality of the proposed algorithm, serving as a foundational proof-of-principle. Within the employed model, a distinction is made between (i) *population dynamics*, which might encompass reactions that modify the population (e.g., birth or death) or the state of the considered population (e.g., change in the number of susceptible, infected, diagnosed), and (ii) *contact dynamics*, which involve reactions influencing the contact network among individuals within the populations.

Crucially, the algorithmic concept is applicable to any model that can be configured in this manner, regardless of which specific reactions are categorised under each type. Regarding the model under consideration, attention must be directed towards the calculation of the upper bound $B(T_L)$, altering this computation in accordance with the model setup.

In the presented examples, constant reaction rates were assumed for both contact and spreading processes. Nevertheless, the possibility of considering rates that change with time is also possible. Take, for example, the transmission rate γ, which might be time-dependent due to a temporal pharmacokinetic effect from prophylaxis, or may vary based on the duration an individual has been infected or contagious, as supported by various studies [142–144, 134]. Such time-variant effects can be incorporated while determining the upper bound $B(T_L)$ in Algorithm 8 (see section 4.3.2).

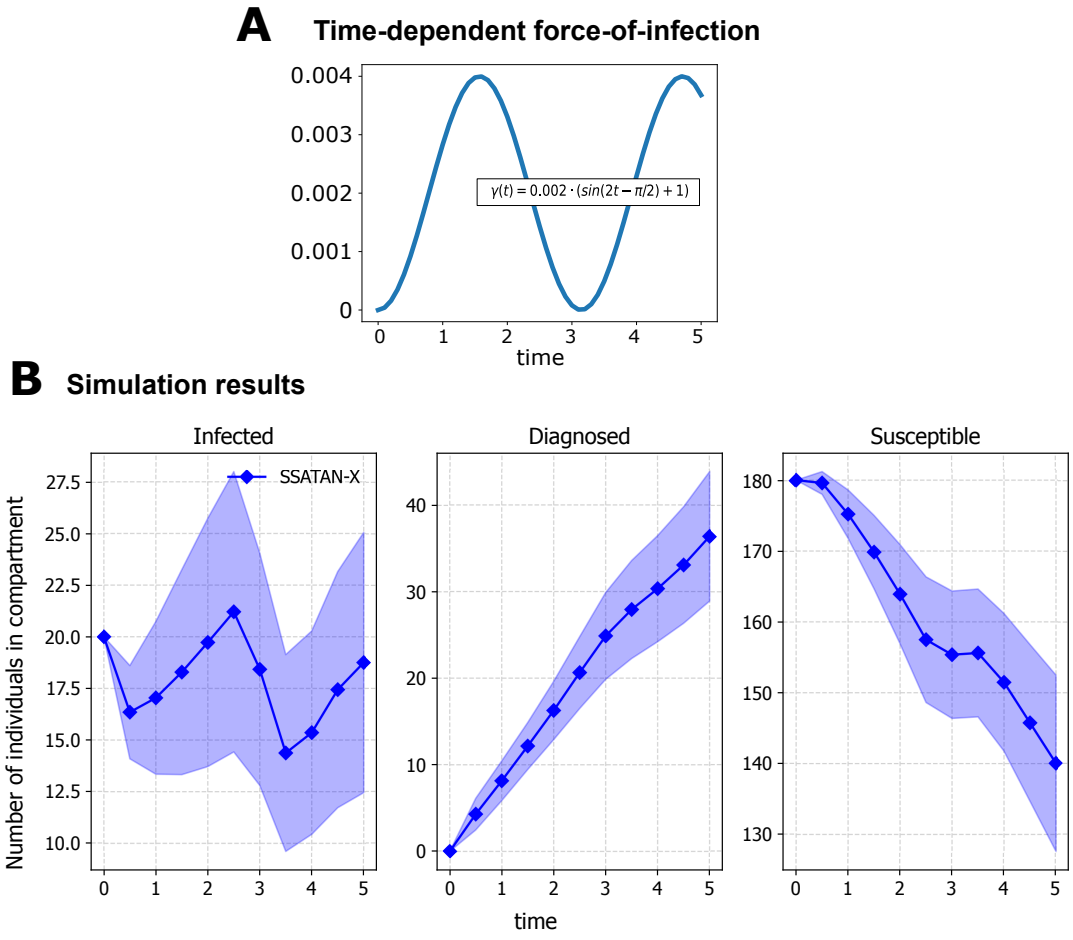


Fig. 7.1. Spreading process model with the time-varying transmission rate. The models described in chapter 5 with the time-dependent transmission rate $\gamma(t) = 0.002(\sin(2t - \pi/2) + 1)$ (panel **A**). The graphic depicts the change in the number of infected, diagnosed, and susceptible (panel **B**) individuals over simulation time. Solid blue lines describe the sample means over 10^3 simulations using SSATAN-X algorithm. The shaded area represents the mean $\pm$ standard deviation.

Fig. 7.1 shows the example model from chapter 5, expanded to include a time-dependent force of transmission, $\gamma(t)$ using a sine function as an example, to depict, for example, the seasonality of the disease. However, any time-variant function can be utilised. As noted above, one must attend to the following when calculating $B(T_L)$, and $\max(\gamma)$ in equation (4.19) is calculated as $\max(\gamma) = \max_{h\in[t,t+T_L]} \gamma(h)$. After the time step $\Delta t \sim \text{Exp}(B_{T_L})$ is sampled, and the progression $t \leftarrow t + \Delta t$ is made, the actual $\gamma(t)$ needs to be computed, and subsequently, all reaction propensities $a_\ell^s(t)$. This example adeptly illustrates dynamic infection control, which affects the transmission rate in a time-dependent manner, as observed with interventions like mask-wearing for airborne infections or for vaccination and prevention initiatives. If a

time-variant contact control is implemented, there are two potential approaches: In instances of abrupt changes (e.g., toggling contact restrictions on/off at a specific time), the approach is straightforward: the look-ahead time horizon T_L aligns with the alteration in contact rates. After the rejection step (line 6 in Algorithm 8) the rates for forming new contacts adjust accordingly from rates *before* to those *after* contact restrictions. Should the rate of forming new contacts $\lambda^+(t)$, or the rate of disassembling existing contacts $\lambda^-(t)$ be an arbitrary, time-continuous deterministic function, adjustments to the computation of B_{T_L} are also necessary. This adaptation can be realised by employing the time-dependent rates in equations (4.1) and subsequently in equations (4.7). These equations might be numerically solved using standard ODE solvers to compute all variables in equation (4.19).

In addition, the utilisation of continuous-time functions in the modelling of either contact or population dynamics reactions introduces a complexity in which the SSA algorithm suffers degradation of numerical precision, since these algorithms are specifically designed for simulating homogeneous Markov processes. In such scenarios, it is advisable to employ the integral-based method instead [9]. Notably, SSATAN-X is also capable of handling these cases effectively.

7.2.2 Restrictions on population size and contacts capacity

In the presented research, consideration is given to finite populations. Within such populations, each individual i inherently possesses an upper theoretical limit for potential contacts of $N-1$, while a complete graph encompasses $N \cdot (N-1)/2$ edges. This paradigm facilitates a direct estimation of the total number of edges available for either deletion or addition at any given time point t: The number of deletable edges equates to the present number of edges $E(t)$, whereas the edges available for addition can be calculated as $N \cdot (N-1)/2 - E(t)$, equivalently, the number of edges within a complement graph. Crucially, in infinite populations, the quantity of edges that can be deleted remains $E(t)$, while for adding edges, one might judiciously forgo the 'complement graph' strategy, thereby enhancing the algorithm's efficiency.

Numerous authors have proposed models imposing stricter restrictions on the number of contacts [35]. Some concepts may derive inspiration from the well-known Dunbar number [145, 146], which presupposes that individuals cannot sustain stable social relationships with more than 150 others, although it is crucial to acknowledge that Dunbar's hypothesis has not been universally accepted without criticism [147].

Moreover, the existence of contact networks, wherein the maximal number of contacts for an individual is fewer than the available individuals within the network, is plausible. For instance, Schmid and Kretzschmar [35] delineate an individual-based model, where each

individual has a definitive upper-limit capacity concerning the number of concurrent sexual partnerships. Pair formation and separation are simulated as dynamic processes, with each individual harbouring unique probabilities of contact formation and separation, derived from a pertinent distribution. In employing SSATAN-X to model such restrictive contact dynamics, the number of edges available for deletion remains unaltered, while the estimation of the number of edges available for addition becomes more intricate: Subject to the order of the addition and remaining contact capacities, various outcomes may emerge, each with different possible numbers of edges to be added. For example, in a network where all nodes but one have exhausted their capacities, no further edges can be added without breaching the contact constraints. Consequently, it becomes imperative to estimate a 'worst case' scenario, that is, a contact bulk update resulting in a minimal overall number of edges. To achieve this, the "reverse" Havel–Hakimi algorithm, with modifications also considering existing edges $E(t)$, might be implemented. The original algorithm [148] is traditionally used to determine whether a sequence of numbers can represent the degree sequence of a simple, undirected graph and to construct the corresponding graph accordingly. This is achieved by sequentially connecting nodes with the highest residual degree. If, in the end, all capacities are exhausted, the result is a simple undirected graph. Contrary to this, the approach outlined in **Algorithm 12** proposes to sequentially connect nodes with the lowest residual degree until no further connections are possible. This method assists in estimating the maximum number of edges that could be added to the network in a 'worst-case' scenario.

Algorithm 12. Havel–Hakimi-based inverse approach.

Input T_F, $G(0)$

Put all vertices in priority queue Q where lower value of the residual capacity C_j has higher priority
nEdges = 0
while $len(Q) > 0$ **do**
    a = Q.pop()
    if $C_a > 0$ **then**
        for i = 1.. $min(C_a, len(Q)$ **do**
            if $C_i > 0$ and vertices a and i not connected **then**
                nEdges = nEdges + 1
                $C_i = C_i - 1$

7.2.3 Non-adaptive contact dynamics

In instances where the contact dynamics are non-adaptive, they may, in fact, be precomputed and seamlessly integrated into the original Extrande Algorithm 6. The calculation of the contact-dependent segment of the upper bound $B(T_L)$ may be executed directly, as all contacts are pre-simulated and known in advance. For instance, in the context of the model utilised in chapter 5, it might be expressed as $B_{S\to I}(T_L) = \max(\gamma) \cdot \sum_{i\in I} \sum_{j\in S} \int_0^{T_L} \delta_{ij}(t)\, \mathrm{d}t$, where $\delta_{ij}(t)$ takes the value of 1 if an edge $\{v_i, vj\}$ is present at time t.

7.2.4 Algorithms used in this work

In this research, the Anderson $\boldsymbol{\tau}$-leap algorithm [115] was employed to calculate the approximate contact updates in the intervals between two epidemiological events. Although alternative tau-leap methodologies, as overviewed in [99], are feasible, the Anderson tau-leap algorithm was selected due to its assurance of accuracy via adaptive step size adjustment. The $\boldsymbol{\tau}$-leaping method, akin to explicit first-order Euler methods in solving systems of ODEs [106, 149], can be viewed as a stochastic simulation algorithm with particular analogies. An alternative strategy for the mass updating of the contact dynamics might involve adapting higher-order schemes with adaptive step sizes, as proposed in ODE contexts [150]. A notable example might be the utilisation of the Runge-Kutta-Fehlberg method [151] to modulate step sizes for mass contact updating. The benefit of employing higher-order schemes in mass contact updating resides in the ability to select larger step sizes while maintaining accuracy [152]. Subsequent advancements of SSATAN-X are expected to involve the integration of these higher-order approximation methodologies.

As proposed in chapter 4, the calculation of the putative time for reactions of type R^s in the SSATAN-X envelope algorithm is informed by the upper bound of total propensities ensuring that contact updates within this time frame will not necessitate an earlier putative time. This facilitates the implementation of the next reaction method algorithm for contact dynamics reactions R^c using an indexed priority queue Q, containing the edge-addition and edge-deletion reactions, prioritising those with smaller putative times that are shaped by reaction propensities and sampled via the next reaction method's principles. This approach enables the rapid and precise simulation of contact dynamics.

Additionally, some authors also report the observation of inter-contact times not distributed according to exponential distributions [153–155]. In order to capture these dynamics, one may look into simulating contact dynamics using, for example, Non-Markovian Gillespie algorithm proposed by Masuda et al. [156, 97].

7.3 Outlook

7.3.1 Non-Markovian dynamics

While the SSATAN-X Algorithm is primarily engineered to simulate Markovian processes, it can be moderately and coherently extended to integrate certain non-Markovian dynamics. For instance, should certain processes—like recovery or other state transitions—be steered by a deterministic duration, the look-ahead time T_L might be aptly set to reflect this duration.

7.3.2 Model scaling

Simulating agent-based models, particularly during peak spreading stages of phenomena like epidemics or viral content dissemination on social networks, demands substantial memory and processing power due to the need to manage numerous agents. Unlike compartmental modelling approaches, which maintain consistent computation times regardless of population size, agent-based methods typically require linear or supralinear scaling with population size. This scaling necessitates the introduction of a "scaling factor," whereby one agent represents multiple individuals, thereby simplifying calculations.

While implementing this technique, meticulous attention to detail is paramount as it can compromise result granularity and introduce inaccurate stochastic variations, particularly when too few agents are used. Although beneficial for conserving computational resources, this approach can affect the model's resolution and accuracy, notably for rare events and in smaller-scale scenarios. The integration of dynamic scaling into the SSATAN-X algorithm and software is planned for future project development, with a particular focus on precision and nuanced execution.

7.3.3 Eliminating Absorbing States

When investigating spreading processes on contact networks, the focus is frequently directed toward the proliferation of phenomena over an extended temporal scale. Consequently, when any agent enters an absorbing state (e.g., recovery state R in the SIR model), it can be judiciously excluded from further consideration. Furthermore, given that the transmission of a pathogen is typically confined to interactions between individuals in two specific states (e.g., only from I to S in the SIR model, or from I or D to S in the scenario delineated in chapter 5), it becomes pertinent to sideline all irrelevant edges. The focus should, therefore, pivot towards exclusively simulating and updating those contacts that remain relevant for the ongoing transmission of the phenomena. Enhancing SSATAN-X's performance by tailoring its focus in this manner represents a promising avenue for future refinement and development.

7.3.4 Minimising Propensity Updates

An additional refinement to the algorithm, aimed to boost its computational efficiency, involves minimising updates to the reaction propensities. Although this is already mostly incorporated in the foundational idea of the SSATAN-X algorithm, which segregates complex dynamics into two distinct categories – population dynamics R^s and contact dynamics R^c – room for further enhancement remains. Such improvement could be achieved by carefully tracking dependencies between reactions. By updating reaction propensities exclusively when they are likely to be influenced by a specific triggered reaction, computational efficiency can be significantly improved. For a deeper exploration of the techniques and data structures that can be employed for this purpose, please refer to [97].

7.3.5 Incorporation of various agents characteristics

In the intricate domain of contact dynamics and disease spreading, behaviours – notably those related to the establishment and severance of contacts – along with alterations in susceptibility or contagiousness are subject to an array of influencing factors and processes, both internal and external to the agents involved. Examples abound, such as how medicating infected individuals can reduce their contagiousness or how the implementation of pandemic countermeasures (e.g., mask mandates in public transit and the enforcement of social distancing) can truncate the number of relevant contacts for disease propagation, thereby tempering overall infection rates [133–135]. These multifaceted impacts can be seamlessly incorporated at both the agent and population levels of the model, providing a refined perspective for navigating through the complex dynamics of disease dissemination and containment, as well as connecting pathogen dynamics within a host to the spread of infection at the population level. The design of the SSATAN-X algorithm allows for these modifications to be straightforwardly incorporated into the simulation.

7.3.6 Machine learning and AI application

The application of machine learning (ML) techniques is a notable and novel avenue to possibly enhance agent sophistication and more accurately model human behaviours within this context [157]. By leveraging ML, agents can ostensibly "learn" from historical data and modify their behaviours more realistically and adaptively. Such an approach not only anchors agent actions and decisions in a richer, data-driven context but also furnishes the model with a capacity for more nuanced and dynamic responses to evolving epidemiological landscapes, thereby enhancing the predictive and exploratory utility of the simulation frame-

work. Incorporating machine learning tools into the adaptiveSpreadX software package has been part of a sustained, ongoing development plan.

References

[1] Christof Schuette and Philipp Metzner. Markov chains and jump processes. *An Introduction to Markov and Jump Processes on Countable States Spaces. Freie Universit Berlin, Berlin*, 2009.

[2] University of Alabama in Huntsville Kyle Siegrist, Department of Mathematical Sciences. Random: Probability, mathematical statistics, stochastic processes. http://www.randomservices.org/random/. Accessed: 2023-12-18.

[3] P. Todorovic. *An Introduction to Stochastic Processes and Their Applications*. Springer Series in Statistics. Springer New York, 2012.

[4] Daniel T. Gillespie. *Markov Processes: An Introduction for Physical Scientists*. Academic Press Inc., 1992.

[5] Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.

[6] Donald A. McQuarrie. Stochastic approach to chemical kinetics. *Journal of Applied Probability*, 4(3):413–478, 1967.

[7] NG Van Kampen. Stochastic processes in chemistry and physics, 1981.

[8] William Ogilvy Kermack and A. G. McKendrick. A contribution to the mathematical theory of epidemics. *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, 115(772):700–721, 1927.

[9] David F Anderson. A modified next reaction method for simulating chemical systems with time dependent propensities and delays. *J Chem Phys*, 127(21):214107, Dec 2007.

[10] Daniel T. Gillespie, Andreas Hellander, and Linda R. Petzold. Perspective: Stochastic algorithms for chemical kinetics. *The Journal of Chemical Physics*, 138(17):170901, 2013.

[11] Daniel T Gillespie. Stochastic simulation of chemical kinetics. *Annu Rev Phys Chem*, 58:35–55, 2007.

[12] Chetan Gadgil, Chang Hyeong Lee, and Hans G Othmer. A stochastic analysis of first-order reaction networks. *Bulletin of Mathematical Biology*, 67(5):901–946, 2005.

[13] Samuel Mwalili, Mark Kimathi, Viona Ojiambo, Duncan Gathungu, and Rachel Mbogo. Seir model for covid-19 dynamics incorporating the environment and social distancing. *BMC Res Notes*, 13(1):352, Jul 2020.

[14] Douglas A. Luke and Katherine A. Stamatakis. Systems Science Methods in Public Health: Dynamics, Networks, and Agents. *Annual Review of Public Health*, 33(1):357–376, 2012.

[15] Abdulrahman M El-Sayed, Peter Scarborough, Lars Seemann, and Sandro Galea. Social network analysis and agent-based modeling in social epidemiology. *Epidemiologic Perspectives & Innovations : EP+I*, 9(1):1–1, 2012.

[16] Eric Silverman, Umberto Gostoli, Stefano Picascia, Jonatan Almagor, Mark McCann, Richard Shaw, and Claudio Angione. Situating agent-based modelling in population health research. *Emerging Themes in Epidemiology*, 18(1):10, 2021.

[17] M. E. J. Newman. Spread of epidemic disease on networks. *Physical Review E*, 66(1):016128, 2002.

[18] Marc Barthélemy, Alain Barrat, Romualdo Pastor-Satorras, and Alessandro Vespignani. Velocity and hierarchical spread of epidemic outbreaks in scale-free networks. *Phys. Rev. Lett.*, 92:178701, Apr 2004.

[19] Lauren Ancel Meyers, Babak Pourbohloul, M.E.J. Newman, Danuta M. Skowronski, and Robert C. Brunham. Network theory and sars: predicting outbreak diversity. *Journal of Theoretical Biology*, 232(1):71–81, 2005.

[20] Petter Holme and Jari Saramäki. *Temporal network theory*, volume 2. Springer, 2019.

[21] Joel C. Miller. Spread of infectious disease through clustered populations. *Journal of The Royal Society Interface*, 6(41):1121–1134, 2009.

[22] Matt J Keeling and Ken T.D Eames. Networks and epidemic models. *Journal of The Royal Society Interface*, 2(4):295–307, 2005.

[23] Marcel Salathé and James H. Jones. Dynamics and Control of Diseases in Networks with Community Structure. *PLoS Computational Biology*, 6(4):e1000736, 2010.

[24] J. O. Lloyd-Smith, S. J. Schreiber, P. E. Kopp, and W. M. Getz. Superspreading and the effect of individual variation on disease emergence. *Nature*, 438(7066):355–359, 2005.

[25] Jonathan M Read, Ken T.D Eames, and W. John Edmunds. Dynamic social networks and the implications for the spread of infectious disease. *Journal of The Royal Society Interface*, 5(26):1001–1007, 2008.

[26] Petter Holme and Jari Saramäki. Temporal networks. *Physics Reports*, 519(3):97–125, 2012.

[27] Jessica Enright and Rowland Raymond Kao. Epidemics on dynamic networks. *Epidemics*, 24:88–97, 2018.

[28] Thilo Gross, Carlos J. Dommar D'Lima, and Bernd Blasius. Epidemic Dynamics on an Adaptive Network. *Physical Review Letters*, 96(20):208701, 2006.

[29] Shweta Bansal, Bryan T Grenfell, and Lauren Ancel Meyers. When individual behaviour matters: homogeneous and network models in epidemiology. *Journal of The Royal Society Interface*, 4(16):879–891, 2007.

[30] I.Z. Kiss, J.C. Miller, and P.L. Simon. *Mathematics of Epidemics on Networks: From Exact to Approximate Models*. Interdisciplinary Applied Mathematics. Springer International Publishing, 2017.

[31] R.M. Anderson and R.M. May. *Infectious Diseases of Humans: Dynamics and Control*. Infectious Diseases of Humans: Dynamics and Control. OUP Oxford, 1991.

[32] Romualdo Pastor-Satorras, Claudio Castellano, Piet Van Mieghem, and Alessandro Vespignani. Epidemic processes in complex networks. *Reviews of Modern Physics*, 87(3):925–979, 2015.

[33] Deven T. Hamilton, Mark S. Handcock, and Martina Morris. Degree Distributions in Sexual Networks: A Framework for Evaluating Evidence. *Sexually Transmitted Diseases*, 35(1):30–40, 2008.

[34] Fredrik Liljeros, Christofer R. Edling, Luís A. Nunes Amaral, H. Eugene Stanley, and Yvonne Åberg. The web of human sexual contacts. *Nature*, 411(6840):907–908, 2001.

[35] M Kretzschmar and M Morris. Measures of concurrency in networks and the spread of infectious disease. *Math Biosci*, 133(2):165–95, Apr 1996.

[36] P. Erdös and A. Rényi. On random graphs i. *Publicationes Mathematicae Debrecen*, 6:290, 1959.

[37] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[38] Paul Erdős and Alfréd Rényi. On the strength of connectedness of a random graph. *Acta Mathematica Hungarica*, 12(1):261–267, 1961.

[39] M. E. J. Newman. The Structure and Function of Complex Networks. *SIAM Review*, 45(2):167–256, 2003.

[40] Béla Bollobás and Oliver Riordan. Handbook of Large-Scale Random Networks. *Bolyai Society Mathematical Studies*, pages 15–115, 2008.

[41] Béla Bollobás, Svante Janson, and Oliver Riordan. The phase transition in inhomogeneous random graphs. *Random Structures and Algorithms - RSA*, 31, 08 2007.

[42] Bernard M Waxman. Routing of multipoint connections. *IEEE journal on selected areas in communications*, 6(9):1617–1622, 1988.

[43] Sergey Dorogovtsev. *Lectures on Complex Networks*. Oxford University Press, 02 2010.

[44] Mark EJ Newman and Juyong Park. Why social networks are different from other types of networks. *Physical review E*, 68(3):036122, 2003.

[45] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440–442, 1998.

[46] Remi Monasson. Diffusion, localization and dispersion relations on "small-world" lattices. *The European Physical Journal B-Condensed Matter and Complex Systems*, 12:555–567, 1999.

[47] Mark EJ Newman and Duncan J Watts. Scaling and percolation in the small-world network model. *Physical review E*, 60(6):7332, 1999.

[48] J.-P. Onnela, J. Saramäki, J. Hyvönen, G. Szabó, D. Lazer, K. Kaski, J. Kertész, and A.-L. Barabási. Structure and tie strengths in mobile communication networks. *Proceedings of the National Academy of Sciences*, 104(18):7332–7336, 2007.

[49] Edward A Bender and E Rodney Canfield. The asymptotic number of labeled graphs with given degree sequences. *Journal of Combinatorial Theory, Series A*, 24(3):296–307, 1978.

[50] Michele Catanzaro, Marián Boguñá, and Romualdo Pastor-Satorras. Generation of uncorrelated random scale-free networks. *Physical Review E*, 71(2):027103, 2005.

[51] Guilherme Ferraz de Arruda, Francisco A. Rodrigues, and Yamir Moreno. Fundamentals of spreading processes in single and multilayer complex networks. *Physics Reports*, 756:1–59, 2018.

[52] Mark EJ Newman. Random graphs with clustering. *Physical review letters*, 103(5):058701, 2009.

[53] Václav Havel. Poznámka o existenci konečných grafů. *Časopis pro pěstování matematiky*, 080(4):477–480, 1955.

[54] S Louis Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. i. *Journal of the Society for Industrial and Applied Mathematics*, 10(3):496–506, 1962.

[55] Fan Chung and Linyuan Lu. Connected components in random graphs with given expected degree sequences. *Annals of combinatorics*, 6(2):125–145, 2002.

[56] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[57] Derek J. de Solla Price. Networks of scientific papers. *Science*, 149(3683):510–515, 1965.

[58] Petter Holme and Beom Jun Kim. Growing scale-free networks with tunable clustering. *Physical review E*, 65(2):026107, 2002.

[59] Marc Barthélemy. Crossover from scale-free to spatial networks. *Europhysics letters*, 63(6):915, 2003.

[60] Sergey N Dorogovtsev, Alexander V Goltsev, and José Ferreira F Mendes. Pseudofractal scale-free web. *Physical review E*, 65(6):066122, 2002.

[61] Iaroslav Ispolatov, Pavel L Krapivsky, and Anton Yuryev. Duplication-divergence model of protein interaction network. *Physical review E*, 71(6):061911, 2005.

[62] Ove Frank and David Strauss. Markov graphs. *Journal of the american Statistical association*, 81(395):832–842, 1986.

[63] David Strauss. On a General Class of Models for Interaction. *SIAM Review*, 28(4):513–527, 1986.

[64] Stanley Wasserman and Philippa Pattison. Logit models and logistic regressions for social networks: I. an introduction to markov graphs and p. *Psychometrika*, 61(3):401–425, 1996.

[65] Carolyn J Anderson, Stanley Wasserman, and Bradley Crouch. A p* primer: logit models for social networks. *Social Networks*, 21(1):37–66, 1999.

[66] Garry Robins, Pip Pattison, Yuval Kalish, and Dean Lusher. An introduction to exponential random graph (p*) models for social networks. *Social Networks*, 29(2):173–191, 2007.

[67] Dean Lusher, Johan Koskinen, and Garry Robins. *Exponential random graph models for social networks: Theory, methods, and applications*. Cambridge University Press, 2013.

[68] Sourav Chatterjee and Persi Diaconis. Estimating and understanding exponential random graph models. *The Annals of Statistics*, 41(5):2428–2461, 2013.

[69] Jon M Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[70] Sanjay Jain and Sandeep Krishna. Autocatalytic sets and the growth of complexity in an evolutionary model. *Physical Review Letters*, 81(25):5684, 1998.

[71] Michael Knudsen and Carsten Wiuf. A markov chain approach to randomly grown graphs. *Journal of Applied Mathematics*, 2008, 2008.

[72] Lauren Ancel Meyers, M.E.J. Newman, and Babak Pourbohloul. Predicting epidemics on directed contact networks. *Journal of Theoretical Biology*, 240(3):400–418, 2006.

[73] Amy C. Kinsley, Gianluigi Rossi, Matthew J. Silk, and Kimberly VanderWaal. Multilayer and multiplex networks: An introduction to their use in veterinary epidemiology. *Frontiers in Veterinary Science*, 7, 2020.

[74] Cliff C. Kerr, Robyn M. Stuart, Dina Mistry, Romesh G. Abeysuriya, Katherine Rosenfeld, Gregory R. Hart, Rafael C. Núñez, Jamie A. Cohen, Prashanth Selvaraj, Brittany Hagedorn, Lauren George, Michał Jastrzębski, Amanda S. Izzo, Greer Fowler, Anna Palmer, Dominic Delport, Nick Scott, Sherrie L. Kelly, Caroline S. Bennette, Bradley G. Wagner, Stewart T. Chang, Assaf P. Oron, Edward A. Wenger, Jasmina

Panovska-Griffiths, Michael Famulare, and Daniel J. Klein. Covasim: An agent-based model of covid-19 dynamics and interventions. *PLOS Computational Biology*, 17(7):1–32, 07 2021.

[75] Matthieu Nadini, Lorenzo Zino, Alessandro Rizzo, and Maurizio Porfiri. A multi-agent model to study epidemic spreading and vaccination strategies in an urban-like environment. *Applied Network Science*, 5(1):68, 2020.

[76] Petter Holme. Modern temporal network theory: a colloquium. *The European Physical Journal B*, 88(9):234, 2015.

[77] Mohammad Mehdi Hosseinzadeh, Mario Cannataro, Pietro Hiram Guzzi, and Riccardo Dondi. Temporal networks in biology and medicine: a survey on models, algorithms, and tools. *Network Modeling Analysis in Health Informatics and Bioinformatics*, 12(1):10, 2022.

[78] Naoki Masuda and Petter Holme. *Temporal network epidemiology*. Springer, 2017.

[79] Petter Holme. Epidemiologically optimal static networks from temporal network data. *PLoS computational biology*, 9(7):e1003142, 2013.

[80] Luis EC Rocha and Vincent D Blondel. Bursts of vertex activation and epidemics in evolving networks. *PLoS computational biology*, 9(3):e1002974, 2013.

[81] Nicola Perra, Bruno Gonçalves, Romualdo Pastor-Satorras, and Alessandro Vespignani. Activity driven modeling of time varying networks. *Scientific reports*, 2(1):469, 2012.

[82] Christian L. Vestergaard, Mathieu Génois, and Alain Barrat. How memory generates heterogeneous dynamics in temporal networks. *Phys. Rev. E*, 90:042805, Oct 2014.

[83] Michele Starnini, Andrea Baronchelli, and Romualdo Pastor-Satorras. Modeling human dynamics of face-to-face interaction networks. *Physical review letters*, 110(16):168701, 2013.

[84] Nataša Djurdjevac Conrad, Luzie Helfmann, Johannes Zonker, Stefanie Winkelmann, and Christof Schütte. Human mobility and innovation spreading in ancient times: a stochastic agent-based simulation approach. *EPJ Data Science*, 7(1):24, 2018.

[85] Matthieu Nadini, Alessandro Rizzo, and Maurizio Porfiri. Epidemic Spreading in Temporal and Adaptive Networks with Static Backbone. *IEEE Transactions on Network Science and Engineering*, 7(1):549–561, 2018.

[86] Thomas E Gorochowski. Agent-based modelling in synthetic biology. *Essays in biochemistry*, 60(4):325–336, 2016.

[87] Albert-Laszlo Barabasi. The origin of bursts and heavy tails in human dynamics. *Nature*, 435(7039):207–211, 2005.

[88] Anders Johansen. Probing human response times. *Physica A: Statistical Mechanics and its Applications*, 338(1-2):286–291, 2004.

[89] Márton Karsai, Mikko Kivelä, Raj Kumar Pan, Kimmo Kaski, János Kertész, A-L Barabási, and Jari Saramäki. Small but slow world: How network topology and burstiness slow down spreading. *Physical Review E*, 83(2):025102, 2011.

[90] Dirk Brockmann. Human mobility, networks and disease dynamics on a global scale. In *Diffusive Spreading in Nature, Technology and Society*, pages 375–396. Springer, Cham, 2018.

[91] Juan Pablo Gutiérrez-Jara, Katia Vogt-Geisse, Maritza Cabrera, Fernando Córdova-Lepe, and María Teresa Muñoz-Quezada. Effects of human mobility and behavior on disease transmission in a covid-19 mathematical model. *Scientific Reports*, 12(1):10840, 2022.

[92] Diogo H Silva, Celia Anteneodo, and Silvio C Ferreira. Epidemic outbreaks with adaptive prevention on complex networks. *Communications in Nonlinear Science and Numerical Simulation*, 116:106877, 2023.

[93] Meili Li, Manting Wang, Shuyang Xue, and Junling Ma. The influence of awareness on epidemic spreading on random networks. *Journal of Theoretical Biology*, 486:110090, 2020.

[94] Thilo Gross and Bernd Blasius. Adaptive coevolutionary networks: a review. *Journal of The Royal Society Interface*, 5(20):259–271, 2008.

[95] Trystan Leng and Matt J Keeling. Concurrency of partnerships, consistency with data, and control of sexually transmitted infections. *Epidemics*, 25:35–46, 12 2018.

[96] Giulia Simoni, Federico Reali, Corrado Priami, and Luca Marchetti. Stochastic simulation algorithms for computational systems biology: Exact, approximate, and hybrid methods. *WIREs Systems Biology and Medicine*, 11(6):e1459, 2019.

[97] Naoki Masuda and Christian L Vestergaard. Gillespie algorithms for stochastic multiagent dynamics in populations and networks. *Elements in Structure and Dynamics of Complex Networks*, 2022.

[98] Tamás Székely and Kevin Burrage. Stochastic simulation in systems biology. *Computational and Structural Biotechnology Journal*, 12(20-21):14–25, 2014.

[99] Luca Marchetti, Corrado Priami, and Vo Hong Thanh. *Simulation algorithms for computational systems biology*, volume 1. Springer, 2017.

[100] Elohim Fonseca Dos Reis, Aming Li, and Naoki Masuda. Generative models of simultaneously heavy-tailed distributions of interevent times on nodes and edges. *Physical Review E*, 102(5):052303, 2020.

[101] TP Schulze. Kinetic monte carlo simulations with minimal searching. *Physical Review E*, 65(3):036704, 2002.

[102] Christian A Yates and Guido Klingbeil. Recycling random numbers in the stochastic simulation algorithm. *The Journal of chemical physics*, 138(9):094103, 2013.

[103] Michael A Gibson and Jehoshua Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The journal of physical chemistry A*, 104(9):1876–1889, 2000.

[104] Vo Hong Thanh, Corrado Priami, and Roberto Zunino. Efficient rejection-based simulation of biochemical reactions with stochastic noise and delays. *The Journal of chemical physics*, 141(13):10B602_1, 2014.

[105] Vo Hong Thanh, Luca Marchetti, Federico Reali, and Corrado Priami. Incorporating extrinsic noise into the stochastic simulation of biochemical reactions: A comparison of approaches. *The Journal of chemical physics*, 148(6):064111, 2018.

[106] Daniel T. Gillespie. Approximate accelerated stochastic simulation of chemically reacting systems. *The Journal of Chemical Physics*, 115(4):1716–1733, 2001.

[107] Yang Cao, Daniel T. Gillespie, and Linda R. Petzold. Avoiding negative populations in explicit Poisson tau-leaping. *The Journal of Chemical Physics*, 123(5):054104, 2005.

[108] Abhijit Chatterjee, Dionisios G. Vlachos, and Markos A. Katsoulakis. Binomial distribution based τ-leap accelerated stochastic simulation. *The Journal of Chemical Physics*, 122(2):024112, 2005.

[109] Tianhai Tian and Kevin Burrage. Binomial leap methods for simulating stochastic chemical kinetics. *The Journal of chemical physics*, 121(21):10356–10364, 2004.

[110] Muruhan Rathinam, Linda R Petzold, Yang Cao, and Daniel T Gillespie. Stiffness in stochastic chemically reacting systems: The implicit tau-leaping method. *The Journal of Chemical Physics*, 119(24):12784–12794, 2003.

[111] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Adaptive explicit-implicit tau-leaping method with automatic tau selection. *The Journal of chemical physics*, 126(22):224101, 2007.

[112] André Leier, Tatiana T Marquez-Lago, and Kevin Burrage. Generalized binomial τ-leap method for biochemical kinetics incorporating both delay and intrinsic noise. *The Journal of chemical physics*, 128(20):05B623, 2008.

[113] Xiaodong Cai and Zhouyi Xu. K-leap method for accelerating stochastic simulation of coupled chemical reactions. *The Journal of chemical physics*, 126(7):074102, 2007.

[114] Jana Lipková, Georgios Arampatzis, Philippe Chatelain, Bjoern Menze, and Petros Koumoutsakos. S-leaping: An adaptive, accelerated stochastic simulation algorithm, bridging τ-leaping and r-leaping. *Bulletin of mathematical biology*, 81(8):3074–3096, 2019.

[115] David F. Anderson. Incorporating postleap checks in tau-leaping. *The Journal of Chemical Physics*, 128(5):054103, 2008.

[116] Luca Marchetti, Corrado Priami, and Vo Hong Thanh. Hrssa–efficient hybrid stochastic simulation for spatially homogeneous biochemical reaction networks. *Journal of Computational Physics*, 317:301–317, 2016.

[117] S. Gómez, A. Arenas, J. Borge-Holthoefer, S. Meloni, and Y. Moreno. Discrete-time Markov chain approach to contact-based disease spreading in complex networks. *EPL (Europhysics Letters)*, 89(3):38009, 2010.

[118] Daniel Smilkov and Ljupco Kocarev. Influence of the network topology on epidemic spreading. *Physical Review E*, 85(1):016114, 2012.

[119] Brandon DL Marshall, William C Goedel, Maximilian RF King, Alyson Singleton, David P Durham, Philip A Chan, Jeffrey P Townsend, and Alison P Galvani. Potential effectiveness of long-acting injectable pre-exposure prophylaxis for hiv prevention in men who have sex with men: a modelling study. *The lancet HIV*, 5(9):e498–e505, 2018.

[120] Peter G. Fennell, Sergey Melnik, and James P. Gleeson. Limitations of discrete-time approaches to continuous-time contagion dynamics. *Physical Review E*, 94(5):052125, 2016.

[121] Nadezhda Malysheva, Junyu Wang, and Max von Kleist. Stochastic simulation algorithm for effective spreading dynamics on time-evolving adaptive networx (ssatan-x). *Mathematical Modelling of Natural Phenomena*, 17:35, 2022.

[122] Christian L. Vestergaard and Mathieu Génois. Temporal Gillespie Algorithm: Fast Simulation of Contagion Processes on Time-Varying Networks. *PLoS Computational Biology*, 11(10):e1004579, 2015.

[123] P. E. Kloeden and E. Platen. Higher-order implicit strong numerical schemes for stochastic differential equations. *Journal of Statistical Physics*, 66(1):283–314, 1992.

[124] Margaritis Voliotis, Philipp Thomas, Ramon Grima, and Clive G Bowsher. Stochastic simulation of biomolecular networks in dynamic environments. *PLoS Comput Biol*, 12(6):e1004923, 06 2016.

[125] Petter Holme. Fast and principled simulations of the SIR model on temporal networks. *PLoS ONE*, 16(2):e0246961, 2021.

[126] Daniel T. Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[127] Tobias Jahnke and Wilhelm Huisinga. Solving the chemical master equation for monomolecular reaction systems analytically. *Journal of Mathematical Biology*, 54(1):1–26, 2007.

[128] Gerold Alsmeyer. *Chebyshev's Inequality*, pages 239–240. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[129] DF Vysochanskij and Yu I Petunin. Justification of the 3σ rule for unimodal distributions. *Theory of Probability and Mathematical Statistics*, 21(25-36), 1980.

[130] Yang Cao, Daniel T Gillespie, and Linda R Petzold. Efficient step size selection for the tau-leaping simulation method. *The Journal of chemical physics*, 124(4):044109, 2006.

[131] Ira M Longini, Jr, M Elizabeth Halloran, Azhar Nizam, and Yang Yang. Containing pandemic influenza with antiviral agents. *Am J Epidemiol*, 159(7):623–33, Apr 2004.

[132] Marie-Claude Boily, Rebecca F Baggaley, Lei Wang, Benoit Masse, Richard G White, Richard J Hayes, and Michel Alary. Heterosexual risk of hiv-1 infection per sexual act: systematic review and meta-analysis of observational studies. *Lancet Infect Dis*, 9(2):118–29, Feb 2009.

[133] Sulav Duwal, Daniel Seeler, Laura Dickinson, Saye Khoo, and Max von Kleist. The utility of efavirenz-based prophylaxis against hiv infection. a systems pharmacological analysis. *Front Pharmacol*, 10:199, 2019.

[134] Wiep van der Toorn, Djin-Ye Oh, Daniel Bourquain, Janine Michel, Eva Krause, Andreas Nitsche, Max von Kleist, and Working Group on SARS-CoV-2 Diagnostics at RKI. An intra-host sars-cov-2 dynamics model to assess testing and quarantine strategies for incoming travelers, contact management, and de-isolation. *Patterns (N Y)*, 2(6):100262, Jun 2021.

[135] Wiep van der Toorn, Djin-Ye Oh, Max von Kleist, and Working Group on SARS-CoV-2 Diagnostics at RKI. Covidstrategycalculator: A software to assess testing and quarantine strategies for incoming travelers, contact management, and de-isolation. *Patterns (N Y)*, 2(6):100264, Jun 2021.

[136] Fernando P Polack, Stephen J Thomas, Nicholas Kitchin, Judith Absalon, Alejandra Gurtman, Stephen Lockhart, John L Perez, Gonzalo Pérez Marc, Edson D Moreira, Cristiano Zerbini, Ruth Bailey, Kena A Swanson, Satrajit Roychoudhury, Kenneth Koury, Ping Li, Warren V Kalina, David Cooper, Robert W Frenck, Jr, Laura L Hammitt, Özlem Türeci, Haylene Nell, Axel Schaefer, Serhat Ünal, Dina B Tresnan, Susan Mather, Philip R Dormitzer, Uğur Şahin, Kathrin U Jansen, William C Gruber, and C4591001 Clinical Trial Group. Safety and efficacy of the bnt162b2 mrna covid-19 vaccine. *N Engl J Med*, 383(27):2603–2615, 12 2020.

[137] Robert M Grant, Javier R Lama, Peter L Anderson, Vanessa McMahan, Albert Y Liu, Lorena Vargas, Pedro Goicochea, Martín Casapía, Juan Vicente Guanira-Carranza, Maria E Ramirez-Cardich, Orlando Montoya-Herrera, Telmo Fernández, Valdilea G Veloso, Susan P Buchbinder, Suwat Chariyalertsak, Mauro Schechter, Linda-Gail Bekker, Kenneth H Mayer, Esper Georges Kallás, K Rivet Amico, Kathleen Mulligan, Lane R Bushman, Robert J Hance, Carmela Ganoza, Patricia Defechereux, Brian Postle, Furong Wang, J Jeff McConnell, Jia-Hua Zheng, Jeanny Lee, James F Rooney, Howard S Jaffe, Ana I Martinez, David N Burns, David V Glidden, and iPrEx Study Team. Preexposure chemoprophylaxis for hiv prevention in men who have sex with men. *N Engl J Med*, 363(27):2587–99, Dec 2010.

[138] Myron S Cohen, Ying Q Chen, Marybeth McCauley, Theresa Gamble, Mina C Hosseinipour, Nagalingeswaran Kumarasamy, James G Hakim, Johnstone Kumwenda, Beatriz Grinsztejn, Jose H S Pilotto, Sheela V Godbole, Sanjay Mehendale, Suwat Chariyalertsak, Breno R Santos, Kenneth H Mayer, Irving F Hoffman, Susan H Eshleman, Estelle Piwowar-Manning, Lei Wang, Joseph Makhema, Lisa A Mills, Guy de Bruyn, Ian Sanne, Joseph Eron, Joel Gallant, Diane Havlir, Susan Swindells, Heather Ribaudo, Vanessa Elharrar, David Burns, Taha E Taha, Karin Nielsen-Saines,

David Celentano, Max Essex, Thomas R Fleming, and HPTN 052 Study Team. Prevention of hiv-1 infection with early antiretroviral therapy. *N Engl J Med*, 365(6):493–505, Aug 2011.

[139] Nancy H L Leung, Daniel K W Chu, Eunice Y C Shiu, Kwok-Hung Chan, James J McDevitt, Benien J P Hau, Hui-Ling Yen, Yuguo Li, Dennis K M Ip, J S Malik Peiris, Wing-Hong Seto, Gabriel M Leung, Donald K Milton, and Benjamin J Cowling. Respiratory virus shedding in exhaled breath and efficacy of face masks. *Nat Med*, 26(5):676–680, 05 2020.

[140] S Weller and K Davis. Condom effectiveness in reducing heterosexual hiv transmission. *Cochrane Database Syst Rev*, (1):CD003255, 2002.

[141] W.F. Tichy. A catalogue of general-purpose software design patterns. In *Proceedings of TOOLS USA 97. International Conference on Technology of Object Oriented Systems and Languages*, pages 330–339, 1997.

[142] S Duwal, V Sunkara, and M von Kleist. Multiscale systems-pharmacology pipeline to assess the prophylactic efficacy of nrtis against hiv-1. *CPT Pharmacometrics Syst Pharmacol*, 5(7):377–87, 2016.

[143] S Duwal, L Dickinson, S Khoo, and M von Kleist. Hybrid stochastic framework predicts efficacy of prophylaxis against HIV. *PLoS Comput Biol*, 14(6):e1006155, 2018.

[144] Lanxin Zhang, Junyu Wang, and Max von Kleist. Numerical approaches for the rapid analysis of prophylactic efficacy against hiv with arbitrary drug-dosing schemes. *PLoS Comput Biol*, 17(12):1009295, Dec 2021.

[145] R.I.M. Dunbar. *Grooming, gossip, and the evolution of language*. Harvard University Press, 1998.

[146] R. I. M. Dunbar. Neocortex size as a constraint on group size in primates. *Journal of Human Evolution.*, 22(6):469–493, 1992.

[147] Patrik Lindenfors, Andreas Wartel, and Johan Lind. 'dunbar's number' deconstructed. *Biol Lett*, 17(5):20210158, 05 2021.

[148] S. L. Hakimi. On realizability of a set of integers as degrees of the vertices of a linear graph. I. *J. Soc. Indust. Appl. Math.*, 10:496–506, 1962.

[149] JC Butcher. *Numerical Methods for Ordinary Differential Equations, Second Edition*. Wiley, 2008.

[150] Pau Rué, Jordi Villà-Freixa, and Kevin Burrage. Simulation methods with extended stability for stiff biochemical kinetics. *BMC Syst Biol*, 4:110, Aug 2010.

[151] E. Fehlberg. Classical fifth-, sixth-, seventh-, and eighth-order runge-kutta formulas with stepsize control. *NASA Technical Report (TR)*, 1968.

[152] David F Anderson, Arnab Ganguly, and Thomas G Kurtz. Error analysis of tau-leap simulation methods. *The Annals of Applied Probability*, 21(6):2226–2262, 2011.

[153] Juliette Stehlé, Nicolas Voirin, Alain Barrat, Ciro Cattuto, Lorenzo Isella, Jean-François Pinton, Marco Quaggiotto, Wouter Van den Broeck, Corinne Régis, Bruno Lina, et al. High-resolution measurements of face-to-face contact patterns in a primary school. *PloS one*, 6(8):e23176, 2011.

[154] Xuan Zhou, Zhifeng Zhao, Rongpeng Li, Yifan Zhou, Jacques Palicot, and Honggang Zhang. Human mobility patterns in cellular networks. *IEEE Communications Letters*, 17(10):1877–1880, 2013.

[155] Elohim Fonseca dos Reis, Aming Li, and Naoki Masuda. Generative models of simultaneously heavy-tailed distributions of interevent times on nodes and edges. *Physical Review E*, 102(5):052303, 2020.

[156] Naoki Masuda and Luis EC Rocha. A gillespie algorithm for non-markovian stochastic processes. *Siam Review*, 60(1):95–115, 2018.

[157] Molood Ale Ebrahim Dehkordi, Jonas Lechner, Amineh Ghorbani, Igor Nikolic, Émile Chappin, and Paulien Herder. Using Machine Learning for Agent Specifications in Agent-Based Models and Simulations: A Critical Review and Guidelines. *Journal of Artificial Societies and Social Simulation*, 26(1), 2023.