Computational models and tools for analysis, prediction, and control of infectious diseases

by

Tanvir Ferdousi

B.S., Bangladesh University of Engineering and Technology, 2013

AN ABSTRACT OF A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Mike Wiegers Department of Electrical and Computer Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Abstract

Infectious disease modeling is used to examine pathogen transmission retrospectively and forecast outbreaks preemptively. Model results help public health authorities to optimize disease control measures, preventing catastrophic loss of lives in humans and animals. Yet, several fundamental challenges arise in infectious disease modeling. A critical problem involves modeling new and evolving pathogens for realistic simulations and reliable predictions of outcomes. Another concern is the lack of data related to infectious diseases. Epidemic modelers often face data inadequacy with host networks and disease incidence. This dissertation proposes remedies to challenges associated with infectious disease modeling, outbreak prediction, and host movement data.

In response to vector-borne disease modeling challenges, this dissertation first takes a mechanistic approach. To realistically model the infection process, a novel interconnected network model is designed for the mosquito-vectored Zika virus, which links homogeneous vector populations with heterogeneous human contact networks. The model incorporates seasonal variations in mosquito abundance and characterizes hosts based on age group and gender. The aim is to develop a detailed model for an accurate representation of pathogen dynamics while keeping it computationally tractable. An event-based simulation tool is developed based on the non-Markovian Gillespie algorithm. This work investigates effects of seasonal variations on outbreak size, the role of sexual transmission in sustaining the pathogen, and relative contributions of key model parameters using a sensitivity analysis.

A framework to improve machine learning performance for predicting dengue fever cases is developed in a data-driven approach. The goal is to fill in temporally limited human case data from spatially adjacent populations. The method ranks and sorts time-series data from peripheral locations around a target location as predictor variables commonly referred to as features. Metrics are computed from windowed time-shifted cross-correlation of incidence

data, spatial distance, and historical prevalence to rank feature variables. A window detection method presented in this work analyzes incidence data to identify time intervals with significant outbreaks. The framework achieves improved prediction performance and works well with recurrent neural network (RNN) architectures. Performance gains are compared using different time window allocation methods for three distinct prediction models: linear, long short-term memory (LSTM), and gated recurrent units (GRU).

Availability of data also affects applicability of mechanistic models. In the United States, farm animal movements are not tracked by a central authority. Lack of animal movement data is a significant obstacle in using network models to analyze infectious outbreaks in meat-producing industries. As an immediate solution, a novel method is presented to generate movement networks from limited data available in the public domain. A custom configuration model is developed for network generation that uses aggregate data from farm animal movement-related surveys and the U.S. agricultural census. A hypothetical spread of the African swine fever virus (ASFV) is simulated in a generated network to analyze how network structure affects pathogen dispersal. A node centrality-based analysis is performed to identify important farm operation types and evaluate how targeted control measures affect outbreaks.

The experience of working with infectious disease models for the U.S. meat-producing industry revealed fundamental problems linked to trust and business data sharing. The U.S. beef cattle industry lacks adequate traceability, as most farm owners consider such data confidential, possibly harming their businesses if exposed. Blockchains, also known as distributed ledgers, have gained popularity in industrial supply chains because of their unique features of data immutability and transparency. A smart contract-based supply chain framework is designed using a private blockchain network. This system supports anonymity for users to protect their identities and lets everyone store data locally while ensuring the blockchain records any change in data with cryptographic proofs. The framework presented contains functionalities to perform business transactions, transfer animal data, conduct anonymous surveys, and trace animals.

This work has original contributions in network epidemic models, data-driven prediction

tools, network generation algorithms, and data management frameworks. It combines knowledge from social network analysis, graph theory, epidemiology, machine learning, statistics, cryptography, computer networks, and computational science to improve infectious disease modeling, analysis, and control. The knowledge gained here is generalizable to applications beyond specific cases presented in this dissertation.

Computational models and tools for analysis, prediction, and control of infectious diseases

by

Tanvir Ferdousi

B.S., Bangladesh University of Engineering and Technology, 2013

A DISSERTATION

submitted in partial fulfillment of the
requirements for the degree

DOCTOR OF PHILOSOPHY

Mike Wiegers Department of Electrical and Computer Engineering
Carl R. Ice College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2021

Approved by:

Major Professor
Caterina Maria Scoglio

Copyright

Abstract

Infectious disease modeling is used to examine pathogen transmission retrospectively and forecast outbreaks preemptively. Model results help public health authorities to optimize disease control measures, preventing catastrophic loss of lives in humans and animals. Yet, several fundamental challenges arise in infectious disease modeling. A critical problem involves modeling new and evolving pathogens for realistic simulations and reliable predictions of outcomes. Another concern is the lack of data related to infectious diseases. Epidemic modelers often face data inadequacy with host networks and disease incidence. This dissertation proposes remedies to challenges associated with infectious disease modeling, outbreak prediction, and host movement data.

In response to vector-borne disease modeling challenges, this dissertation first takes a mechanistic approach. To realistically model the infection process, a novel interconnected network model is designed for the mosquito-vectored Zika virus, which links homogeneous vector populations with heterogeneous human contact networks. The model incorporates seasonal variations in mosquito abundance and characterizes hosts based on age group and gender. The aim is to develop a detailed model for an accurate representation of pathogen dynamics while keeping it computationally tractable. An event-based simulation tool is developed based on the non-Markovian Gillespie algorithm. This work investigates effects of seasonal variations on outbreak size, the role of sexual transmission in sustaining the pathogen, and relative contributions of key model parameters using a sensitivity analysis.

A framework to improve machine learning performance for predicting dengue fever cases is developed in a data-driven approach. The goal is to fill in temporally limited human case data from spatially adjacent populations. The method ranks and sorts time-series data from peripheral locations around a target location as predictor variables commonly referred to as features. Metrics are computed from windowed time-shifted cross-correlation of incidence

data, spatial distance, and historical prevalence to rank feature variables. A window detection method presented in this work analyzes incidence data to identify time intervals with significant outbreaks. The framework achieves improved prediction performance and works well with recurrent neural network (RNN) architectures. Performance gains are compared using different time window allocation methods for three distinct prediction models: linear, long short-term memory (LSTM), and gated recurrent units (GRU).

Availability of data also affects applicability of mechanistic models. In the United States, farm animal movements are not tracked by a central authority. Lack of animal movement data is a significant obstacle in using network models to analyze infectious outbreaks in meat-producing industries. As an immediate solution, a novel method is presented to generate movement networks from limited data available in the public domain. A custom configuration model is developed for network generation that uses aggregate data from farm animal movement-related surveys and the U.S. agricultural census. A hypothetical spread of the African swine fever virus (ASFV) is simulated in a generated network to analyze how network structure affects pathogen dispersal. A node centrality-based analysis is performed to identify important farm operation types and evaluate how targeted control measures affect outbreaks.

The experience of working with infectious disease models for the U.S. meat-producing industry revealed fundamental problems linked to trust and business data sharing. The U.S. beef cattle industry lacks adequate traceability, as most farm owners consider such data confidential, possibly harming their businesses if exposed. Blockchains, also known as distributed ledgers, have gained popularity in industrial supply chains because of their unique features of data immutability and transparency. A smart contract-based supply chain framework is designed using a private blockchain network. This system supports anonymity for users to protect their identities and lets everyone store data locally while ensuring the blockchain records any change in data with cryptographic proofs. The framework presented contains functionalities to perform business transactions, transfer animal data, conduct anonymous surveys, and trace animals.

This work has original contributions in network epidemic models, data-driven prediction

tools, network generation algorithms, and data management frameworks. It combines knowledge from social network analysis, graph theory, epidemiology, machine learning, statistics, cryptography, computer networks, and computational science to improve infectious disease modeling, analysis, and control. The knowledge gained here is generalizable to applications beyond specific cases presented in this dissertation.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I express my gratitude to Dr. Caterina Scoglio, my research supervisor, for her continuous support and guidance throughout this doctoral program. I have learned a great deal from her on how to delve into academic research. She has been patient and kind with my mistakes and has helped me grow as a researcher. She has guided me well through challenging times with her experience and wisdom. This dissertation has thoroughly improved from her editorial advice.

I thank Dr. Lee Cohnstaedt, who advised me for a significant part of my Ph.D. program. His expert opinions of mosquito-vectors and vector-borne diseases were essential to this work. I also appreciate Dr. Don Gruenbacher for his mentorship in the blockchain project. It was a great privilege to work with Mr. Adrian Self and Dr. David Amrine and learn from their disciplines.

I am grateful to my beloved wife, colleague, and friend, Sifat Afroj Moon, for her inspiration in reaching this milestone. She has stayed with me throughout this journey with tremendous patience and support.

I acknowledge the people of the Network Science and Engineering Group: Mahbubul Huq Riad, Qihui Yang, Aram Vajdi, and Chunlin Yi for their company and support. I also thank all the people I met at the ECE department for making this journey a pleasant one.

Finally, I recognize the unconditional love of my parents, without whom I would not be here.

Dedication

To the two people who make my world. My mother, Selina Nargis, and my daughter, Aurora Ferdousi.

Preface

With the title "Computational models and tools for analysis, prediction, and control of infectious diseases," this dissertation is submitted for the degree of Doctor of Philosophy in the Department of Electrical and Computer Engineering at Kansas State University. The research has been performed under the supervision of Dr. Caterina Scoglio.

This work is original to the best of my knowledge, except where acknowledgments and references are made to previous works. Parts of this work have been published in the following peer-reviewed journal articles.

- T. Ferdousi, L. W. Cohnstaedt, D. S. McVey, and C. M. Scoglio, "Understanding the survival of Zika virus in a vector interconnected sexual contact network," *Scientific Reports*, 9(1), 7253, 2019.[1]

- T. Ferdousi, S. A. Moon, A. Self, and C. M. Scoglio, "Generation of swine movement network and analysis of efficient mitigation strategies for African swine fever virus," *PLOS ONE*, 14(12), 2019.[2]

- T. Ferdousi, D. Gruenbacher, and C. M. Scoglio, "A Permissioned Distributed Ledger for the US Beef Cattle Supply Chain," *IEEE Access*, 8, 154833-154847, 2020.[3]

This research has been supported by the Department of the Army, U.S. Army Contracting Command, Aberdeen Proving Ground, Natick Contracting Division, Fort Detrick (DWFP Grant W911QY-19-1-0004), the NSF/NIH/USDA/BBSRC Ecology and Evolution of Infectious Diseases (EEID) Program through USDA-NIFA Award 2015-67013-23818, the National Science Foundation under Grant CMMI-1744812, and the Global Food Systems Seed Grant Program, Kansas State University. The findings, conclusions, or recommendations expressed in this dissertation do not necessarily reflect the views of mentioned funding agencies.

Chapter 1

Introduction

1.1 Background

Infectious diseases have threatened human societies since the early days of civilization. Despite scientific breakthroughs in medical interventions throughout history, new pathogens have emerged in almost every era, disrupting human lives and ravaging economies. We cannot overstate the importance of characterizing infections, projecting the risk of spread, and planning control measures. Epidemic models that aid such efforts can range from mechanistic to purely data-driven. Network models combine tools and techniques developed in social sciences, graph theory, and epidemiology to compute infection dynamics. This kind of modeling is mechanistic with assumptions about processes governed by the laws of natural sciences. Machine learning models learn from past examples to predict future values based on experience. These are data-driven approaches that require a considerable number of observations on prediction targets. Selection of one approach over another usually depends on our understanding of the pathogen, availability of infection-related data, and results we are trying to achieve.

Regardless of approach, fundamental challenges are associated with modeling and predicting infectious diseases. A mechanistic model needs to incorporate complex behaviors and interactions of hosts and pathogens while remaining computationally efficient. In addition to

that, the model parameters need to be estimated. Inadequacy of data is a common problem with epidemic modeling. Lack of disease incidence data can significantly limit use of machine learning models. A network model requires a host network structure to compute epidemic processes, which may not be readily available. In the United States, host movement data in the animal farming industry are difficult to obtain due to a lack of trust. Blockchain is a decentralized ledger technology that can create trust, improve traceability, and secure data in a supply chain. This dissertation adopts blockchain as a technological solution to a business problem that affects infectious disease modeling and mitigation strategies.

1.1.1 Network models for epidemic analysis

Network models compute epidemic dynamics at the population level using individual-level behaviors[4]. A network allocates contacts for an individual, which is a departure from classical epidemic models that assume random mixing. In a real-world scenario, an individual may not randomly interact with other people. For example, a person may only interact with his family members, relatives, friends, and colleagues within a finite time interval. Number of contacts will also vary from one person to another. These are some of the reasons why network models are useful tools in modern epidemiology. Using tools and techniques of network science[5], we can identify individuals' risky behaviors and analyze how a network topology contributes to the spreading process. These insights can aid control measures such as contact tracing based early interventions[6–8]. Using node centrality metrics[9], we can identify important nodes (e.g., hosts) and target interventions more effectively. Node degree is one such metric that counts the number of active contacts of a node. Node degree distribution in a network is commonly used to characterize the network. For example, studies have concluded that sexual contact networks have scale-free structures characterized by a degree distribution that decays according to a power law[10–12].

A key ingredient of a network-based transmission model is data on the network itself, described by nodes (hosts) and the edges (contacts) that connect them. One way to get this ingredient is to perform surveys or observe social behaviors to record interactions among

hosts in a target population. Such studies have been conducted before[13]. However, these are limited in quantity and pertain to small populations depending on those studies' objectives. A more workable method for epidemic modelers is to generate such networks using a graph generator. There are graph generation models that can produce networks of varying characteristics. Some popular models are Erdős–Rényi[14] for random graphs, Barabási–Albert[15] for scale-free graphs, and Watts-Strogatz[16] for small-world graphs. These graph generators have well-defined parameters that determine output graphs' properties (e.g., the number of nodes, edge density). Infectious spreading networks often require further constraints depending on hosts' geographic locations, commute patterns, and pathogen transmission characteristics. As a result, standard network models may not be readily usable without application-specific modifications.

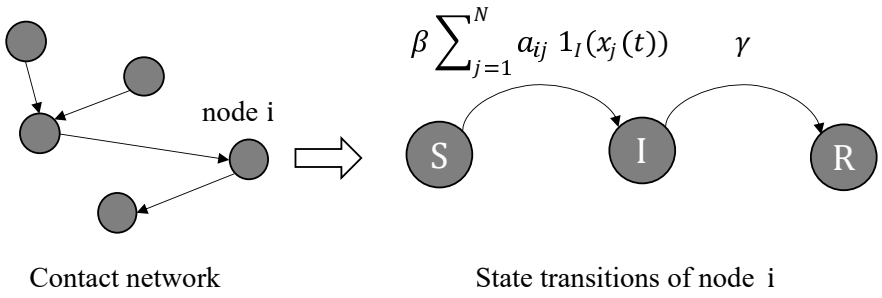


Figure 1.1: *A diagram of a network epidemic model. The circles on the left (contact network) indicate nodes with directional links connecting them. For each node i, the model defines three states: susceptible (S), infectious (I), and recovered (R). The curved arrows indicate state transitions along with the transition rates. The symbols β and γ indicate the infection and recovery rates. The indicator function $1_I(x_j(t))$ ensures that a susceptible host (S) can only be infected by an infectious host (I).*

To demonstrate how to model infectious spreading processes on networks, an example is shown in Figure 1.1. We start with a contact network of N host nodes. Let, $A = [a_{ij}]$ be an adjacency matrix of dimension $N \times N$ that describes the connections among nodes. An element a_{ij} of the matrix indicates the presence ($a_{ij} = 1$) or absence ($a_{ij} = 0$) of an edge/link from node j to node i. Figure 1.1 shows a network with directed edges. However, depending on the application, edges can also be undirected ($a_{ij} = a_{ji}$). For this example, we define three epidemic states for each node: susceptible (S), infectious (I), and recovered (R). Let, $x_i(t)$ be the state of node i at time t. We define the combined state of the

network with N nodes as $X(t) = [x_1(t), x_2(t), x_3(t), ..., x_N(t)]$. For this particular example, the networked system can have 3^N different states. In the state transition diagram of the SIR model, we define two transitions: infection ($S \rightarrow I$) and recovery ($I \rightarrow R$). A host i which is susceptible ($x_i(t) = S$) may get infected with some probability when it comes in contact ($a_{ij} = 1$) with an infected host j ($x_j(t) = I$) at time t. From a single infected host j, the susceptible host i may get infected at a rate β. Once infected, a host i may recover at a rate γ. The recovery process differs from the infection process as it is not affected by host i's neighbors (contacts) in the network. For Markovian processes, transition times are distributed exponentially with an average duration equal to inverse transition rates. The Gillespie[17] algorithm can be used to run stochastic simulations using node transitions as events occurring with their corresponding transition rates. The two transition probabilities are described in the following equations,

$$
\begin{aligned}
P(x_i(t + \Delta t) = I | x_i(t) = S, X(t)) &= \beta \sum_{j=1}^{N} a_{ij} 1_I(x_j(t)) \Delta t + o(\Delta t) \\
P(x_i(t + \Delta t) = R | x_i(t) = I, X(t)) &= \gamma \Delta t + o(\Delta t)
\end{aligned}
\tag{1.1}
$$

Here, Δt is the time step, $1_I(x_j(t))$ is an indicator function which is equal to 1 when $x_j(t) = I$ (node j is infected at time t) and 0 otherwise. The first equation computes a susceptible node's probability of getting infected ($S \rightarrow I$). The second equation computes an infected node's probability of recovering from the infection ($I \rightarrow R$).

1.1.2 Machine learning for time series prediction

Machine learning (ML) models for epidemics lie on the other end of the modeling spectrum when compared with mechanistic models (Section 1.1.1). In supervised learning, a model learns from 'example' data on how to predict an output from a given input while trying to improve prediction accuracy[18], where these 'examples' map inputs to their corresponding outputs. The set of examples used for model learning is known as the training data set. In

the ML domain, inputs and outputs are also referred to as 'features' and 'labels.' Features are predictor variables that may have some degree of influence on output values (labels). For example, to predict a child's height, its age could be used as a feature. Features can be single, few, or many depending on the application and available data. A part of ML research is dedicated to feature selection, which aims to select a subset of relevant features to improve learning performance while reducing processing and storage overheads[19]. A supervised model learns from training data and tries to generalize input-output relationships. Alternatively, this can be described as searching for a hypothesis h that approximates the true function f, which is not known. Multiple hypotheses h may fit a finite training data set[20]. However, the goal is to find one which can reasonably generalize beyond the training set (i.e., perform similarly in predicting unseen examples). Inductive bias is the set of assumptions used to choose one consistent hypothesis over others. One example of inductive bias is to choose the simplest hypothesis consistent with training examples (known as Occam's razor). In practice, available example data are split into train and test data sets. Test data are used to evaluate performance once a model is fitted with training data. When a model predicts training data with comparatively higher accuracy than test data, it is called overfitting[21] (i.e., failing to generalize). Regularization techniques are used to prevent or remedy such situations.

Time series prediction falls in the category of 'sequence prediction' in machine learning. A sequence is a set of data with some meaningful ordering among samples, where adjacent values are related. In other words, sampled data points are not i.i.d. (independent and identically distributed). A time series is a sequence where data points are ordered based on time. For example, during an outbreak, infections spread from infected people to healthy people. Depending on several factors (e.g., number of infectious people in a community), daily new infected cases may follow a pattern such as increasing, decreasing, or remaining unchanged. This is a temporal relationship we expect a model to learn. Conventional models that consider each data point independent of others will not be able to capture temporal relationships.

Artificial neurons are the building blocks (units) of neural networks that have revolution-

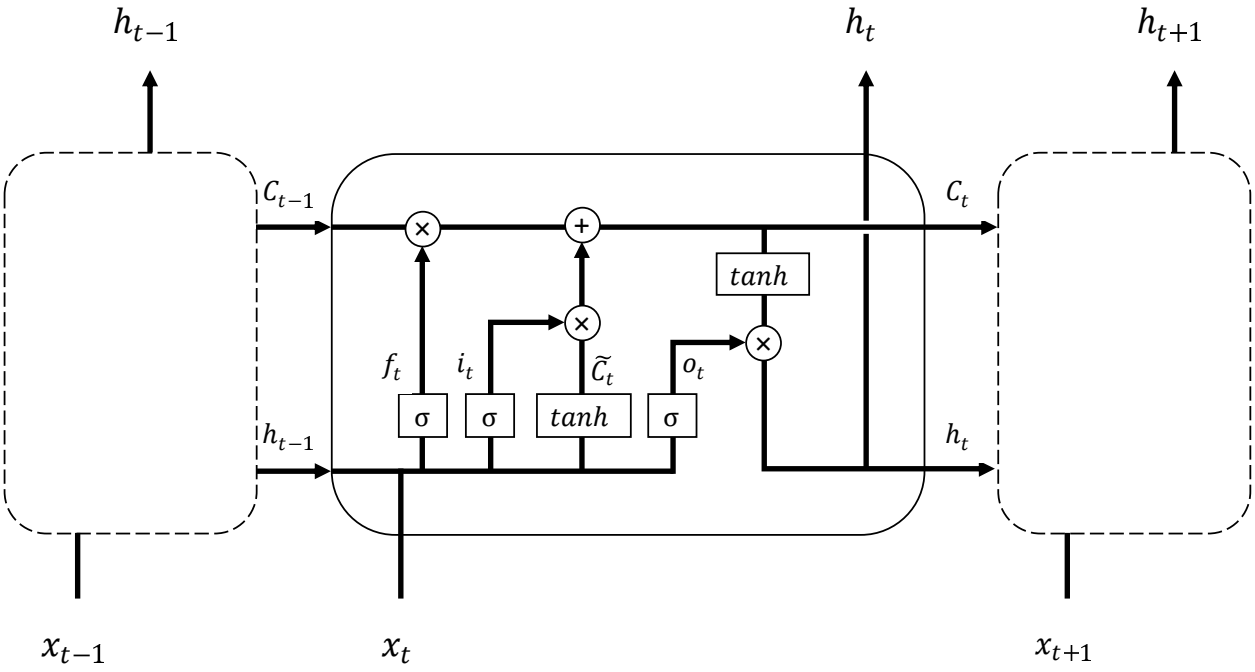


Figure 1.2: *A diagram*[22] *of a long short-term memory (LSTM) unit*[23;24]*. It contains three gates: forget gate (f_t), input gate (i_t), and output gate (o_t), and uses two distinct activation functions: sigmoid (σ) and hyperbolic tangent (tanh). The internal memory state (also called cell state) at time t is indicated by C_t. Inputs and outputs of the unit at time t are x_t and h_t respectively. The $\times$ and $+$ symbols indicate point-wise multiplication and point-wise addition respectively. This is a single LSTM unit unrolled with respect to time.*

ized machine learning[25]. Deep neural networks consist of multiple layers of artificial neurons with varying neural unit structures depending on the application. Some prominent applications of deep learning include image classification, speech recognition, and recommender systems[26–28]. Recurrent neural networks (RNN) are special types of artificial neural networks (ANN) that employ recurrent neural units which can learn sequential data[29]. These units have internal memory states that can store information about past inputs. Early RNN models had a vanishing gradient problem[30] that was solved with advanced architectures: LSTM (long short-term memory) and GRU (gated recurrent units)[23;24;31]. A diagram of an LSTM unit is shown in Figure 1.2. RNN units have self-loops that transfer values (for example, memory states (C_t) and outputs (h_t)) from one time-step to the next. The diagram we show here is unrolled with respect to time for better representation of its sequential nature. Gates in a recurrent unit control the flow of information using a sigmoid layer (σ)

and a point-wise multiplier ($\times$). Note, we use the term point-wise as the symbols shown in Figure 1.2 (e.g., x_t, h_t) are vectors. Size of these vectors depends on input data. Three gates that regulate the internal memory state or output are called input, output, and forget gates. The internal memory state (C_{t-1}) can be modified by the input gate (i_t) or the forget gate (f_t) before being passed to the next time step. These gates may add information to the memory state or remove it depending on the situation. The output is a filtered version of the internal memory state controlled by the output gate (o_t). Researchers have used RNNs successfully for machine translation, speech synthesis, handwriting recognition, and time series prediction[32–35].

1.1.3 Blockchains for distributed ledgers

Blockchains are linked-list structures where a block is linked to its previous block using cryptographic hashes[36]. They have been widely used for cryptocurrencies such as bitcoin[37], which are digital assets under decentralized control. A simplified diagram depicting a blockchain structure is shown in Figure 1.3. It does not show every component of a block, only the items related to this discussion. A block consists of a header and a list of transactions. A transaction is created when one account (e.g., user) in the blockchain network transfers some value (e.g., bitcoin) to another account. The transactions are ordered and added to a new block based on an agreed-upon consensus mechanism[38]. Using a Merkle tree[39] like structure, a hash tree is computed from all transactions in a block. The hash tree root (H_{TR}) is stored in the header. Bitcoin uses SHA-256 as the cryptographic hash function[40]. In addition to H_{TR}, a header contains the hash computed from the previous block's header (for example, H_{k-1} in Block k of Figure 1.3). This ensures any change in content of any previous block will break the chain. A modification in the content of one block would require all subsequent blocks to be updated. This makes it difficult for a perpetrator to change any past data. Most cryptocurrencies use a proof-of-work consensus mechanism. In proof-of-work, active participants (also called miners) try to solve a computationally difficult problem of finding a hash to match some predefined requirements[38]. Miners vary the nonce parameter(N_k) to

search for the desired hash. Successful miners are allowed to add new blocks, and they are rewarded with an amount of crypto-currency. A proof of work (matching hash) is difficult to compute but easy to verify. Blockchains operate using peer-to-peer (P2P) communication protocols. Redundancy is created by each participating node having a copy of the chain.

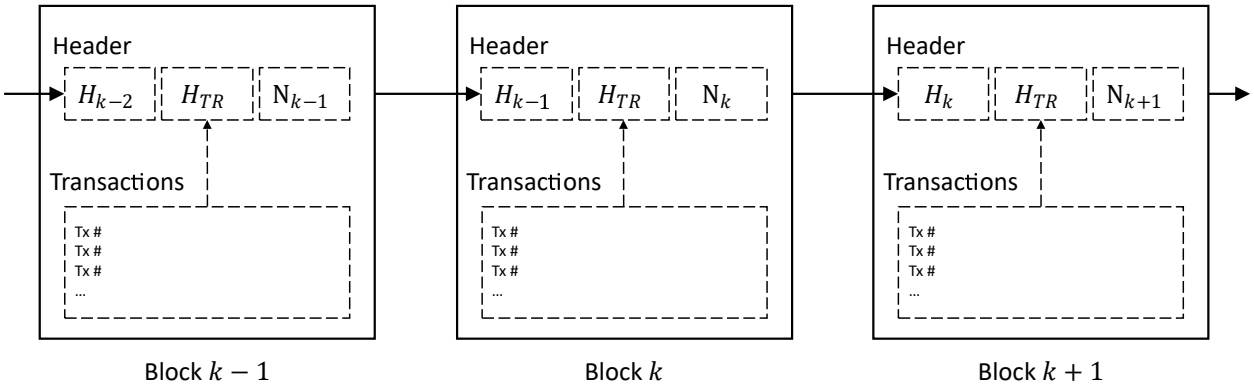


Figure 1.3: *A simplified blockchain architecture. Each block contains some header information and a list of transactions, along with several other parameters. The header of a block (k) contains a cryptographic hash computed from the header of the previous block (H_{k-1}), a cryptographic transaction root hash (H_{TR}) of transactions contained inside the current block (computed using a Merkle tree), and a nonce (N_k) parameter. Each block is linked to a previous block via hash values H_ks as shown using arrows[37]. Note, only a subset of a block's components is shown here.*

More advanced blockchain frameworks such as Ethereum[41], and Hyperledger[42] support smart contracts. A smart contract is a block of computer code stored in the blockchain (along with transaction data). Smart contracts have a 'class' like structure, commonly used in object-oriented programming. It supports inheritance and can have constructors, member variables, and member methods. Smart contracts can be designed to enforce business policies, privacy practices, and access control[43;44]. The blockchain ensures everyone has the same version of code and data, which creates trust.

1.2 Motivation

A general problem with classical epidemic models is their inability to capture complex dynamics precisely during transient stages of pathogen spread in a population. A network model with heterogeneous mixing patterns can remedy this problem, but it may not remain

computationally tractable with increasingly more nodes. It is important to find a middle ground where a model can approximate real-world dynamics with reasonable accuracy while remaining efficient in memory and processing power. Some approaches that deal with large populations are models that use mean-field approximations[45;46] and metapopulations[47;48]. The Zika virus spreading process has several modeling challenges. First, the number of mosquito vectors is several orders of magnitude higher than the number of human hosts in a locality. Zika viruses can also be transmitted sexually from male hosts. A hybrid model is needed which can analyze both modes of transmission.

An alternate approach to predicting vector-borne infections is to use machine learning when data are available on disease incidence and related covariates (e.g., temperature, rainfall). Disease surveillance programs have improved applicability of machine learning approaches. However, data quality varies across regions. Infectious disease processes are complex and continuously evolving. Historical data are often not sufficient to train a machine learning model that can predict with acceptable accuracy. However, modern surveillance programs can provide incidence data at a much finer spatial resolution than it was possible in the past. We need a feature selection method that can take advantage of incidence data from spatially adjacent locations to use them as features. When analyzing time series data among pairs of locations, we also need to quantify phase relationships, geographical distances, and historical prevalence to select the best possible features.

When working with farm animals' infectious diseases in the United States, a significant drawback is lack of animal movement data. No central authority records transfer of animals that happen at the farm level. Business owners fear sharing movement data will benefit competitors, which may cause a loss of revenue. Hence, it is difficult to predict and mitigate the spread of infections in U.S. farm networks. We can approach this problem from two directions. First, A network generator can be designed to use data available in the public domain. It can use the knowledge gained from animal movement surveys and county-level aggregate data to characterize a representative network structure. A second way is to address the lack of trust among businesses directly. A data management framework is needed which can meet the requirements of privacy, data ownership, and traceability while providing

incentives for businesses at the same time.

This dissertation deals with several open problems of infectious disease modeling, analysis, and control. The list of issues includes the network model of Zika virus transmission, data-driven prediction of dengue outbreaks, network generation of farm animal movements, and decentralized database management of meat-producing businesses.

1.3 Dissertation overview

To aid with analysis, prediction, and control of infectious diseases, this dissertation contributes in two distinct ways: i) Chapters 2 and 3 present epidemic modeling techniques for complex infectious processes and ii) Chapters 4 and 5 remedy problems created by missing data that hinders outbreak analysis and mitigation strategies.

Chapter 2 introduces an interconnected model that incorporates both insect-vectored and sexual transmission of the Zika virus. Using the interconnected model, we address computational overhead issues by modeling vectors as homogeneously mixed populations. The vector model incorporates seasonal variations of mosquito abundance in the form of time-varying mosquito birth rates. This chapter also presents a method of generating sexual contact networks from aggregate data on host behavior patterns. We implement an event-based stochastic simulator using the non-Markovian Gillespie algorithm, which interconnects the host and vector sub-models by updating co-dependent model parameters between events. In the results section, we perform a seasonal analysis to understand the effect of the pathogen introduction month on the outbreak. Later, we perform a survival analysis to investigate the impact of crucial model parameters on the vector-free pathogen survival period. We conclude with a sensitivity analysis to compare the relative effect of model parameters.

In Chapter 3, we take a machine learning approach to develop a time series forecast model to predict dengue fever cases. To address the issue of sparse data at a target location, we present a method to compute a windowed time-shifted correlation metric and combine it with distance and prevalence metrics to rank incidence data from nearby locations as potential predictor features. We present two techniques to allocate such windows on the time series

before computing correlation metrics: fixed-length windows and outbreak window detection. We compare results generated using these techniques across three regression models: linear, LSTM (long short-term memory), and GRU (gated recurrent units). We also compare performance improvements when applying the models on municipality-level and ecoregion-level data to predict Brazil's dengue cases.

To remedy the lack of farm animal movement data in the public domain, we present a graph generation method in Chapter 4 to synthesize directed swine movement networks for the U.S. pork industry. Using a generated network representative of the U.S. farm network structure, we simulate the spread of African swine fever virus. By introducing pathogens in different farm operation types, we observe how outbreak sizes are affected. We perform network centrality analysis to identify important node categories (farm operation types) in the disease-spreading process. We also evaluate which centrality measures (in-degree, out-degree, or betweenness) are more effective in controlling outbreaks under targeted interventions of hypothetical vaccines or movement restrictions.

To directly address farm business owners' reluctance in sharing animal movement data, we implement a blockchain-based decentralized data management framework in Chapter 5 for beef cattle supply chains in the United States. We design how the framework handles users, data, and communications using five goal-specific smart contracts. The framework specifies local data storage protocols to ensure security and ownership of data for businesses. We evaluate user privacy, data security, provenance tracking, reliability, and operations cost using integration tests. The tests are performed on a prototype running system of six blockchain nodes. The test results validate the framework's various operational characteristics to ensure that it runs nominally.

1.4 Contributions

Contributions of this dissertation are summarized below:

- Designed an interconnected host-vector model that links heterogeneously-mixed hosts

with homogeneously-mixed vectors to model insect-vectored and sexual transmission of Zika virus.

- Implemented a simulator tool for the interconnected host-vector model using a non-Markovian variant of the Gillespie algorithm.
- Quantified that sexual transmission alone can sustain Zika virus in a population for up to 75 days without vectors.
- Designed and implemented a network generation algorithm for farm animal movement networks in the United States.
- Analyzed the relative impact of node centrality-based targeted control measures and concluded that high in-degree nodes should be treated first for efficient control of African swine fever outbreaks.
- Conceptualized a framework for decentralized storage and management of farm animal data in the United States beef cattle industry using blockchains focusing on user privacy, data security, and trust.
- Designed and implemented a windowed time-shifted correlation-based feature selection method to improve performance in predicting dengue fever outbreaks with neural networks.

Chapter 2

Understanding the survival of Zika virus in a vector interconnected sexual contact network[1]

2.1 Background

Zika virus (ZIKV) is predominantly vector-borne, and the outbreaks are strongly dependent on the mosquito vector density, which is influenced by the climatic conditions[49]. There have been substantial evidence[50] and reports[51] of sexual transmission of ZIKV. Although ZIKV epidemic models mostly focus on mosquito-vectored transmission, multiple studies have quantified the contribution of sexual transmission[52–56]. Modeling analysis has helped to determine the relative importance of different transmission mechanisms. To enhance our understanding of how the combination of vectored and sexual transmission can help sustain the pathogen during the vector-free seasons (e.g., winter), we develop a novel epidemic model that interconnects a homogeneous mosquito vector population with a heterogeneous sexual contact network.

Zika virus is a positive-sense single-stranded RNA virus of the Flaviviridae family[57]. It is

[1]This chapter is a slightly modified version of a published article[1], Copyright 2019, Scientific Reports.

related to other flaviviruses such as dengue, yellow fever, West Nile, and Japanese encephalitis viruses. ZIKV infection symptoms include acute fever, maculopapular rash, arthralgia, and conjunctivitis. However, only about 20% of infected people suffer any significant symptoms[58]. A large proportion of asymptomatic cases poses a significant problem in determining the outbreaks to initiate early response measures. ZIKV infection in pregnant women can result in congenital microcephaly, a severe birth defect[59]. The transmission of ZIKV occurs primarily by infected mosquito vectors such as *Aedes aegypti* and *Aedes albopictus*. However, it can also be spread via infected semen, and blood[60]. The first autochthonous case of Zika in the mainland US was reported in July 2016 in Miami, Florida. As of April 2018, $5,676$ cases have been reported in the US States[61]. In 2016, 224 autochthonous cases were reported where the state of Florida itself is responsible for 97% of those cases, and the remaining 3% cases occurred in Texas[61]. Due to the complex nature and multiple pathways of pathogen transmission, several epidemic parameters are still unknown.

There have been numerous attempts to model the outbreaks of ZIKV. Several models only considered vectored transmission[49;62–64] and several others considered both sexual and vectored transmission routes[52;53]. Kuhlman et al. developed a hybrid agent-based model[64] to simulate Zika outbreaks in the central Miami region. While the work demonstrates interlinked host and vector populations, it does not address host-to-host sexual transmissions. Vertical transmission has also been modeled in a work of Agusto et al.[65]. Olawoyin et al. analyzed the effects of multiple transmission pathways and concluded that secondary transmission mechanisms increase the basic reproductive number and cause the outbreaks to occur sooner compared to the vector-only transmission[66]. The work of Gao et al. concluded that sexual transmission contributes about 3.044% of the overall transmission[52]. Moghadas et al. also found similar contributions from the sexual transmission, and they emphasized the importance of asymptomatic transmission[67]. Maxian et al. concluded that this contribution is too minor to sustain an outbreak, as they found that sexual transmission contributes about 4.8% to the basic reproductive number, R_0[54]. They also concluded that sexual transmission indirectly helps the vectored transmission by increasing the pool of infectious individuals. The relatively higher prevalence of infection in the female population than the male popu-

lation was analyzed by Pugliese et al.[68], who concluded that females' higher susceptibility compared to their male counterparts in the case of sexual transmission could be the reason. The work of Sasmal et al. estimated that sexual transmission contribution could be as high as 15.36% for sexual risk-stratified population[69]. Another study concludes that existing research could potentially underestimate the risk of sustained sexual transmission by an order of magnitude[70]. It has been found that Zika virus can sustain in semen for a long time after it disappears from blood[71]. Hence, even though it is clear that sexual transmission alone can't sustain an outbreak, we don't have clear information about the persistence of the pathogen in a human host network without vectors. Limited information on the reduction of transmissibility for both asymptomatically infected hosts and during the extended sexual transmission period adds to the complexity in understanding how the outcomes are affected.

In this chapter, we develop an individual-based (i.e., node-based) network model, which is different from basic compartmental models as it features heterogeneous mixing. Each individual (i.e., a node) is connected to a specific number of individuals, which can be very different. In the context of real-world sexual networks, this is realistic. Using survey data[72] on sexual behavior, we develop a network generator tool that can produce networks with properties that closely match the given data. As the vector population is much larger than the host population and vectors do not transmit between them, it is unnecessary to use node-based models for vectors. We employ a homogeneous population model for the vectors. We interconnect these two populations (hosts and vectors) via infection rate parameters where the infection of one population depends on the number of infected in the other population. There have been several works on epidemic spreading in interconnected networks[73]. However, due to the multi-host (one being the vector) scenario, our model is built differently from conventional interconnected networks. Our goal is to incorporate heterogeneity in the host population but reduce unnecessary computational overhead by interconnecting with a homogeneous vector population. Our model also differentiates between symptomatically and asymptomatically affected individuals by assigning them different states. We furthermore incorporate the extended sexual transmissibility in our model. In the host network sub-model, each host node can be in either one of the seven states: $SEI_sI_aJ_sJ_aR$ (Susceptible, Exposed,

Infectious (Symptomatic), Infectious (Asymptomatic), Convalescent (Symptomatic), Convalescent (Asymptomatic), and Recovered). The host population is also divided based on gender, sexual orientation, and age. In the vector population sub-model, the population is modeled into three compartments: $S_V E_V I_V$ (Susceptible, Exposed, and Infected). We only use birth-death demography for the vector model, and the climatic variation is incorporated into the vector birth rate, which is one of the variable parameters. The simulation model is based on Gillespie's Stochastic Simulation Algorithm (SSA)[17]. However, we have a few variable rate parameters in our model, which prompted us to use the non-Markovian Gillespie Algorithm (n-MGA)[74] to simulate our model, which we implement using a modified version of the GEMF[75] tool.

The contributions of this chapter are threefold: i) we propose a novel interconnected model to evaluate host-to-host and host-to-vector-to-host pathogen transmission, ii) we develop a sexual contact network generator based on aggregate data, and iii) using our implementation of the non-Markovian Gillespie Algorithm (n-MGA) we examine survival of the pathogen in the host network. We also perform sensitivity analyses on key model parameters to evaluate their role in disease outcomes. The interconnected model is presented in the model formulation section, three independent approaches in evaluating model outputs are presented in the results section, and the methods section contains the details on the simulation model.

2.2 Model formulation

2.2.1 Interconnected population model

We propose an interconnected population model to investigate the spread of ZIKV within the human host and the mosquito vector population during an outbreak. A basic diagram of the model is shown in Figure 2.1. It has two major components: the vector population marked by the cloud on top and the host contact network marked by the solid circle nodes and the solid edges. The vector population is assumed to be in the vicinity of the host

population such that each mosquito vector can bite any of the hosts (indicated by dashed lines).

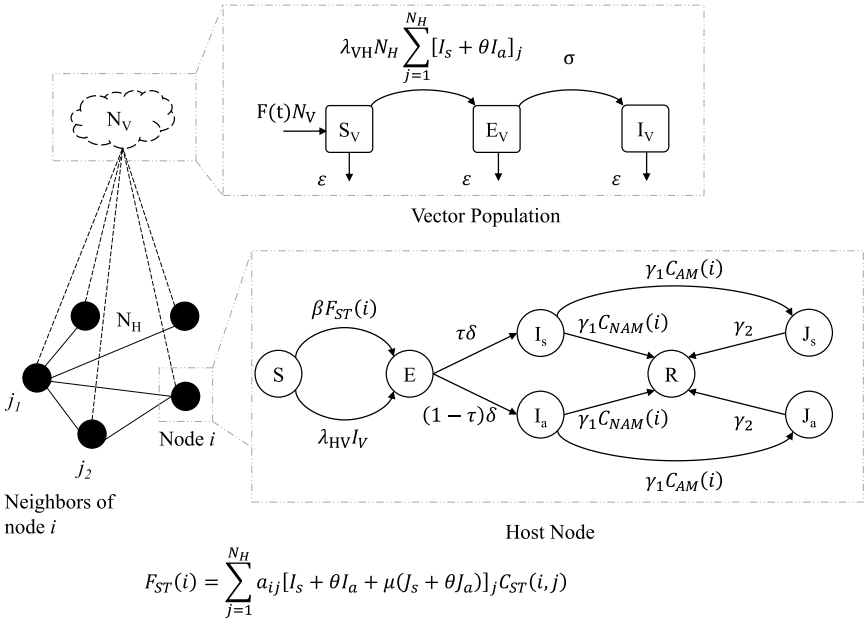


Figure 2.1: *Coupled population model for ZIKV. The black solid circles indicate host nodes (individuals) and the cloud shape above indicates the vector population. The solid edges connecting the nodes indicate the sexual contact network and the dashed edges indicate contacts of the vector population with the host nodes. A host node can be in any of the seven states ($SEI_sI_aJ_sJ_aR$) and the entire vector population is divided into three compartments ($S_VE_VI_V$).*

Each host node can be in either one of the seven states: Susceptible (S), Exposed (E), Infectious (Symptomatic) (I_s), Infectious (Asymptomatic) (I_a), Convalescent (Symptomatic) (J_s), Convalescent (Asymptomatic) (J_a), and Recovered (R). The convalescent states were used to model the extended presence of ZIKV in semen as reported by previous works[71]. People in J_s or J_a states cannot transmit infections via vectors; they can only transmit sexually. The vector population is divided into three compartments: Susceptible (S_V), Exposed (E_V), and Infected (I_V). The unidirectional arrows indicate transition from one state/compartment to another. The symbol adjacent to each arrow indicates the rate of that particular transition. The set of all these symbols constitute the model parameters, and they are listed in Table 2.1. This table also lists the nominal values and the range of values to be used for the

simulation experiments. Some of these values are not readily available in the literature, and we performed calibrations to determine acceptable ranges of values to conduct the experiments. We also performed sensitivity analyses with several critical parameters to evaluate model behavior. We have also used boolean condition functions marked by the symbol, $C_x(n)$ as shown in Figure 2.1. The subscript 'x' indicates which condition a node n has to satisfy. If a condition is met, the condition function returns True (= 1); otherwise, it returns False (= 0). For example, $C_{AM}(i)$ will return 1 if node i is an adult male (AM), it will return 0 otherwise. The conditions that we have used are: AM = Adult Male, and NAM = Not Adult Male. It should be obvious that for any i, $C_{AM}(i)$ is the complement of $C_{NAM}(i)$. $F_{ST}(i)$ is the multiplier component function of the force of infection for sexual transmission from other nodes to node i. The symbols I_s, I_a, J_s, and J_a in the formulation of $F_{ST}(i)$ are host state indicators for neighbors (j) of node i. $C_{ST}(i,j)$ is the sexual transmission condition function. It only returns True (1) if i is an adult and j is an adult male. Here, j is considered the transmitter, and i is considered the recipient of infection.

As indicated in Figure 2.1, a susceptible host node can get infected ($S \rightarrow E$) via two mechanisms: i) getting bitten by an infected mosquito (at the rate λ_{HV}) or ii) having sexual contact with an infected or convalescent (I_s, I_a, J_s or, J_a) host (at the rate β). We use the adjacency matrix representation of a graph, where $\mathcal{A}$ is the adjacency matrix. An element $a_{ij} \in \mathcal{A}$ indicates the connection/link between node i and node j. If a link exists, then $a_{ij} = 1$, otherwise $a_{ij} = 0$. The sexual transmission rate, β was computed using the formula $\beta = 1 - (1 - \alpha)^n$, which was previously used in the work of Agusto[53]. Here, α is the probability of transmission per coital act with an infected partner. As data specific to the Zika virus is not available, we assume a range of α between 0.001 and 0.1[76]. The exponent n is the average number of coital acts per unit of time. Some estimated data indicate that people have sex an average of 60 times per year[72]. Hence, we estimate $n = 60/365 = 0.1644$ per day. The range of β for the range α is 0.000164 to 0.0172. Once a human host is infected, he/she can either develop symptoms or not show any sign of infection at all. Hence, there are two possible transitions from the exposed (E) state: i) a proportion, τ of the people becomes symptomatically infected (I_s) and ii) a proportion, $1 - \tau$ of the people becomes

asymptomatically infected (I_a). The proportion of symptomatic ZIKV infections is claimed by previous works[58] to be about 20%. However, one recent work[77] claims that this proportion could be higher, ranging from $27\% - 50\%$. For our analysis, we take this into consideration as shown in Table 2.1. Once someone is infected, they will stay infected for an average duration of $1/\gamma_1$. However, once the virus disappears from blood, the adult male hosts will go through some extended period of infectiousness which is modeled by the convalescent states (J_s and J_a). An adult male in the convalescent state may only transmit sexually and would recover after an average duration of $1/\gamma_2$. The infectiousness of the four states (I_s, I_a, J_s, and J_a) are not considered the same. The relative transmissibility of the asymptomatic states is represented by θ. The relative transmissibility for the convalescent states is denoted by μ. Although symptomatically infected people could be spreading less if they are seeking medical attention, we do not reduce the transmissibility for the symptomatic cases in this model. Historically, it has been seen that possible negligence associated with ZIKV has contributed to sexual transmissions[78]. A healthy mosquito vector can get infected with some nonzero probability only if it bites an infectious human (I_s or I_a) (at the rate λ_{VH}). The time-dependent per-capita mosquito recruitment rate is represented by $F(t)$ where, $F(t) = (1/12)A(t)$. Here, $1/12$ is a fixed birth rate assumed to be the same as the mortality rate (ϵ). The mosquito abundance factor $A(t)$ is the seasonality parameter that incorporates seasonal variations into the model.

To generate a realistic sexual network, we develop a network generator tool based on the 'configuration model'[5]. Our tool takes as input data of sexual behavior, age, gender, and produces representative random networks. Details are featured in the Methods section. An example network generated by our tool is given in Figure 2.2. In our model, we also intend to incorporate the temporal variations of the climate in different seasons and the consequences of such variation on the mosquito vector population. To simplify this task, we use a vector abundance factor ($A(t)$) expressed as a fraction in the range $[0, 1]$. This parameter was derived from data points used in the work of Monaghan et al.[82], which they originally extracted from the works of Reiskind et al.[80]. For our simulations, we use vector abundance data from Miami, Florida. The mosquito abundance factor over the course of 12

months is plotted in Figure 2.3. The data originally had 12 sample points (monthly values). As the model requires the abundance factor on a daily basis, we use linear interpolations between data points of adjacent months.

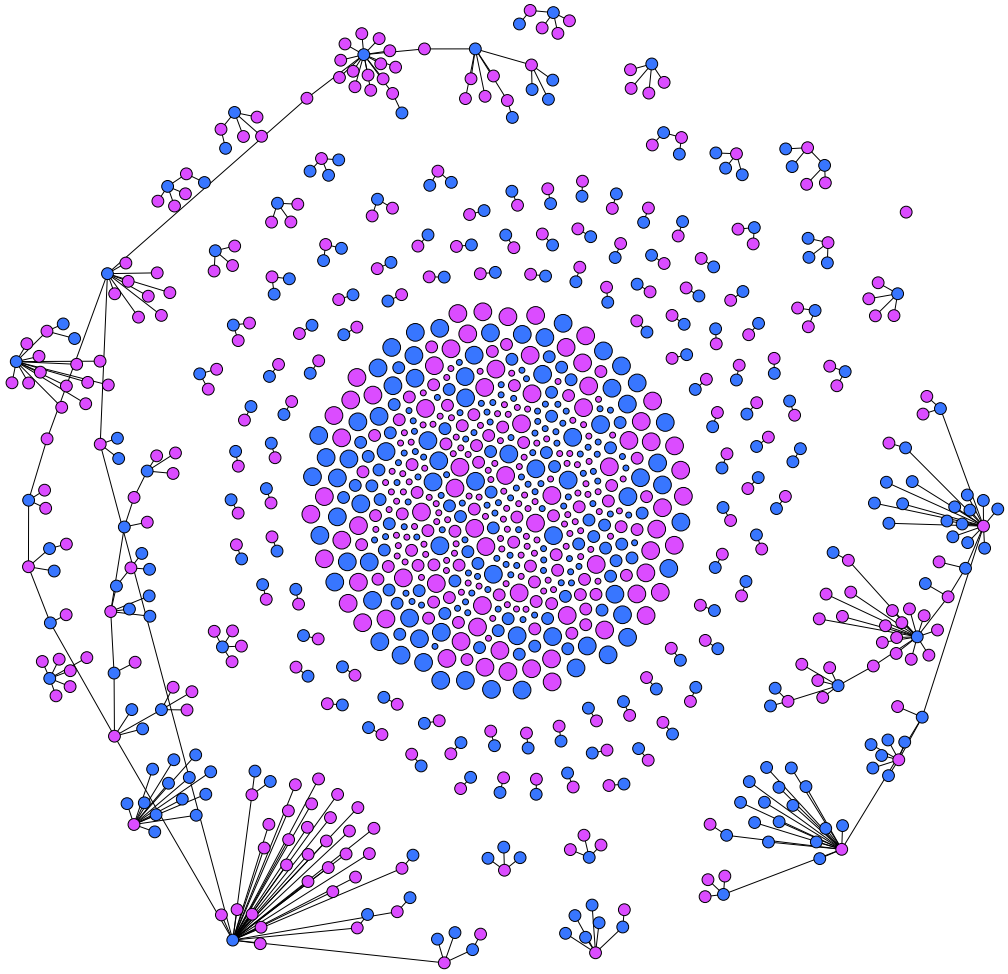


Figure 2.2: *Generated sexual contact network based on sexual behavior*[72]*. The given network has* $N = 1000$ *nodes. The Methods section contain the details on how it is generated. The nodes are presented in two distinct colors and three distinct sizes. The males are marked as blue and the females are marked as pink. The three node sizes correspond to Children (small), Adults (medium), and Elderly (large) people. This particular instance of the network has an average node degree of* 0.803 *whereas the* 64% *adult population have an average degree of 1.26. This particular instance of network has 463 male and 537 female host nodes. The edge density of this network is 0.0008038. There are 600 connected components and the largest one consists of 117 nodes.*

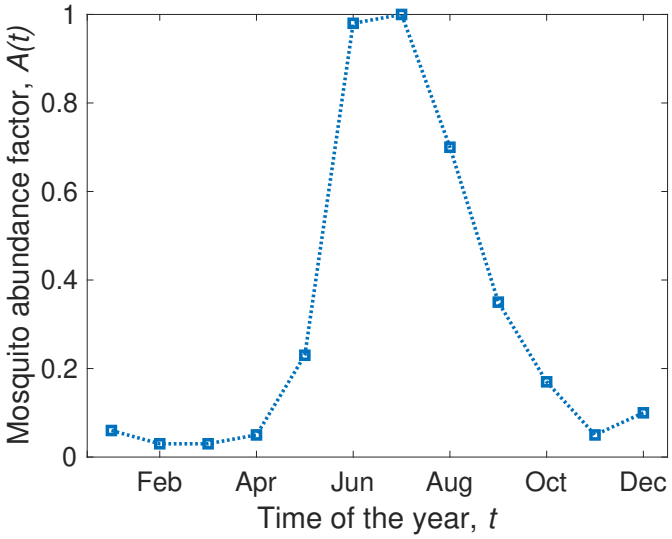


Figure 2.3: *Variation in mosquito vector abundance with time. Due to changing temperature, rainfall, and other climatic factors, mosquito abundance varies throughout the year. The squares show the data points and the dotted lines are linear interpolations. The data represent Palm Beach, Florida where the vector abundance peaks during June-July. The plot was constructed from data observed in Palm Beach, FL over 27 four-week periods from 2006-2008*[80;82].

2.2.2 Simulation tool

If the transition rate parameters are constant, the overall stochastic model is a Markov process with Poisson arrivals and exponential inter-event times. A stochastic algorithm such as the Gillespie SSA[17] can be used to perform simulations in most of these cases. The well-developed GEMFsim[75] has been used previously[83] to solve such stochastic spreading processes in the networks. However, our model requires a few changes before such simulations can be performed.

In our model, the transition rate parameters are not constant. The mosquito birth rate is a seasonally dependent parameter that varies according to the given vector abundance input data. The vector population compartments are changing with time during the outbreak simulation, which changes the host infection rate $\lambda_{HV} I_V$. Similarly, the host infected populations are also changing with time which modify the vector infection rate $\lambda_{VH} N_H \sum_{j=1}^{N_H} [I_s + \theta I_a]_j$. These constitute a set of exogenously varying parameters (i.e., varying due to external forces/catalysts). As a consequence, the processes are no longer Pois-

son; rather, these are called non-homogeneous Poisson. To cope with this issue, we use the non-Markovian generalized Gillespie Algorithm (nMGA)[74] which assumes general renewal processes (which is more generic) that allow exogenous variations of parameters. We modified the existing GEMFsim[75] accordingly and added a vector compartmental model solver on top of it. The vector compartmental solver is effectively coupled to the modified network-based nMGA solver via the infection rate parameters ($\lambda_{HV} I_V$ and $\lambda_{VH} N_H \sum_{j=1}^{N_H} [I_s + \theta I_a]_j$), where a parameter in one population depends on some quantity in the other population.

2.3 Results

The simulation tool uses the parameters listed in Table 2.1, the initial conditions, the seasonal variation data, the population data, and the contact network representation. The initial condition is a single infected vector ($I_V(0) = 1$) while keeping the remaining vector population and the entire host population susceptible (S). We specify the pathogen introduction month (M_{start}) and the max run time (T_{end}). For example, if we set $M_{start} = 5$ and $T_{end} = 100$, the simulation will start from 1$^{\text{st}}$ of May and run up to the 2$^{\text{nd}}$ week of August. A simulation can terminate before T_{end} if the outbreak dies out. As we are running stochastic simulations, we smooth out our results by averaging over 500 repetitions. The population data contains the distribution of genders, sexual orientations, and age groups used to generate the contact network. For the survival analysis, we introduce the pathogen in November ($M_{start} = 11$) to indicate the start of winter. For the sensitivity analysis, we introduce the pathogen into the population in April ($M_{start} = 4$), which is the most likely scenario for the 2016 Miami-Dade outbreak[81].

2.3.1 Seasonal analysis

First, we explore the effect of seasonal variations on the epidemic progression. We use the nominal parameter values defined in Table 2.1 and run the simulations for three distinct pathogen introduction months, M_{start}. Taking some cues from the seasonal patterns of

Miami, FL shown in the Figure 2.3, we choose to run independent scenarios starting on 1st of April ($M_{start} = 4$), 1st of August ($M_{start} = 8$), and 1st of October ($M_{start} = 10$). The averaged results of 500 simulations are shown in Figure 2.4.

Without any intervention, we observe a large outbreak (more than 50% of the population affected) if the pathogen is introduced on 1st of August. For the remaining two cases, the outbreak sizes are comparatively much smaller. This can be explained using the seasonal variations of Figure 2.3. In Miami, the mosquito abundance is high during the beginning of August. This contributes to a boost in the infected vector population compared to the other two dates.

We also observe some interesting behaviors in the outbreak dynamics. Despite the vector abundance having a large positive slope during the month of April and a large negative slope during the month of August, Miami suffers a larger outbreak in the August introduction compared to the April introduction. Figure 2.4b shows that the infected vectors die out soon in the April outbreak, although the upcoming summer (positive slope in abundance) sustains healthy vectors for a long time. In the case of the August introduction, the infected vectors rise rapidly during the first 50 days (Figure 2.4d). Although the vector population declines rapidly after 50 days due to the upcoming winter's unsuitable climates, the initial surge in infected mosquitoes causes a large outbreak. The results indicate that *the suitability of the climate during pathogen introduction plays a significant role in determining the size of the outbreak*. The impacts of climatic changes in the following months are minimal.

2.3.2 Survival analysis

As a part of the survival analysis, we will be observing several measured quantities: the epidemic/infection lengths, the pathogen survival period, and the epidemic attack rates. Here, we define them first before discussing the results.

Host infection length (T_{HL})

The host infection length is defined as,

$$T_{HL} = \textit{The last time instant between 0 and } T_{end} \textit{ where there is at least one infected host left} \tag{2.1}$$

It is the last day on the outbreak time-line an infected host can be found. For this property, the infected hosts who are asymptomatic or in the convalescent state are also considered infected. In this chapter, when we use the term "*epidemic length*", we imply host infection length T_{HL} unless otherwise mentioned.

Vector infection length (T_{VL})

The vector infection length is defined as,

$$T_{VL} = \textit{The last time instant between 0 and } T_{end} \textit{ where there is at least one infected vector left} \tag{2.2}$$

It is the last day on the outbreak time-line an infected vector can be found. In typical outbreak situations, infected vectors die out before all the infected hosts recover.

Pathogen survival period, (T_{PS})

The vector free pathogen survival period is defined as,

$$T_{PS} = T_{HL} - T_{VL} \tag{2.3}$$

It measures how long pathogen can survive in the host population without the presence of vectors.

Epidemic attack rate (AR)

The epidemic attack rate (AR) is defined as,

$$AR = \frac{Number\ of\ hosts\ who\ experience\ infection\ throughout\ the\ outbreak}{Total\ number\ of\ hosts,\ N_H} \tag{2.4}$$

The value of AR is in the range $[0, 1]$. We sometimes express this quantity as a percentage instead of a fraction. For example, an AR of 0.3 means 30% of the population were infected during the outbreak and the remaining 70% never experienced any infection.

To see how some of the above-mentioned quantities relate to sexual transmission, we vary the sexual transmission rate, β in the range mentioned in Table 2.1. We find that varying β does not noticeably increase or decrease the epidemic length (Figure 2.5a), which remains between 2-4 days for the range of β in consideration. Outbreaks in most of these cases last for about 158 days. The effect of increased sexual transmission in modifying pathogen survival is also minor (Figure 2.5b), being within 74-78 days with an average of 76.75 days. A similar situation arises for the attack rate (Figure 2.5c), with the mean attack rate being 15.12%. If there is no sexual transmission ($\beta = 0$), the attack rate is 14.48%. With a sexual transmission rate of 0.0175 (corresponding to about 10% probability of transmission per coital act), the attack rate increases to 15.97%. This accounts for about a 10.29% increase in the epidemic size that a high rate of sexual transmission can cause. We also have an unknown factor, the relative sexual transmissibility (μ) when recovering hosts are in the convalescent phases. For all other simulations, we assume the infectiousness will reduce by 50% in convalescent stages. However, we also vary this parameter to determine how it affects the outcomes. Just like the effects we saw for β, the epidemic lengths remain within 1-2 days of the mean of 157.6 days (Figure 2.5d). The pathogen survival also show similar characteristics as before and stays within 73-76 days (Figure 2.5e). The effect on the epidemic size is barely noticeable (Figure 2.5f), with minor fluctuations.

ZIKV disease outcomes are also affected by a large proportion of asymptomatic hosts. We vary the two parameters (τ and θ) that we use to model the asymptomatically infected

individuals. Although in most works we found that about 20% of the cases were symptomatic, one recent work estimated that this proportion could be higher[77] (27% - 50%). To evaluate the effect of such variations, we vary the symptomatic proportion parameter, τ from 10% to 60%. The epidemic length increases gradually with τ starting from 155 days for 10% symptomatic to 172 days for 60% symptomatic population (Figure 2.5g). A 50% increase in the symptomatic proportion causes a 11% increase in the length of the outbreak. The pathogen survival is affected much less compared to epidemic length. However, we find a small fluctuating decrease followed by a gradual increase (Figure 2.5h). The vector free pathogen survival remains within 2-3 days of the average length of 75.6 days. The epidemic size has a clear increasing trend as shown in Figure 2.5i. This is expected because the symptomatically infected individuals are subject to higher transmission rates. As an example, the outbreaks will be about 45% larger if 50% of the people are symptomatic compared to the usual proportion of 20%. The relative transmissibility has a significantly larger effect on the disease outcomes, and all three properties increase (Figure 2.5 bottom row). In most other simulations, it is assumed by default that infectiousness reduces to half ($\theta = 0.5$) when someone is asymptomatic. Compared to our default situation, if we make infectiousness of both symptomatic and asymptomatic individuals the same, we see 17.5% increase in the epidemic length (Figure 2.5j), 14.45% increase in the pathogen survival period (Figure 2.5k), and a massive 123.5% increase in the epidemic size (Figure 2.5l).

2.3.3 Sensitivity analysis

A sensitivity analysis is performed in order to evaluate the model response with respect to several key parameters. For a vector-borne disease such as ZIKV, the ratio of vector and host population is expected to play a prominent role in the disease outcomes. In our previous analysis, we have also found that the proportion of asymptomatic individuals and their relative transmission rate affect the disease outcomes. To evaluate how these parameters interplay, we perform a pairwise sensitivity analysis of the three parameters: the vector-to-host ratio (N_V/N_H), the proportion of asymptomatic hosts (τ), and the relative transmissibility

of asymptomatic hosts (θ). The results are depicted in Figure 2.6.

The epidemic length (T_{HL}) is affected by the three parameters in question. However, we can see that the effect of symptomatic proportion depends on the relative transmissibility. The symptomatic proportion can regulate epidemic length if relative transmissibility is low, as seen in Figure 2.6b. If θ is above 0.4, the capability of τ is greatly reduced. The vector-to-host ratio (N_V/N_H) always increases the epidemic length (Figures 2.6a and 2.6c), which is also true for the relative transmissibility of asymptomatic individuals (θ) (Figures 2.6b and 2.6c).

We obtain some interesting results when we analyze the sensitivity of pathogen survival, T_{PS}. Pathogens can survive longer for an intermediate range of host to vector ratio if the proportion of symptomatic individuals is low, as shown in Figure 2.6d. On the other hand, despite having some large epidemics on the upper right corner of Figure 2.6g, we see that pathogen survival is relatively lower (as low as 67 days) (Figure 2.6d). In Figure 2.6e, we see that depending on the value of τ, the survival can be higher for certain intermediate values of θ. For example, When $\tau < 0.2$ and $0.3 < \theta < 0.7$, there is a region where pathogen can survive more compared to the other situations. Survival reduces for both high and low values of asymptomatic relative transmissibility (θ). When comparing vector-to-host ratio along with the asymptomatic relative transmissibility, we see that a thick band or a region appears where the pathogen survival is high (Figure 2.6f). There are slight fluctuations in that peak region; however, they are relatively minor to indicate any particular phenomena. If we compare this plot with the attack rate plot (Figure 2.6i), we see that pathogen survival is relatively long for intermediate values of attack rates and relatively short for small or large attack rates. Small outbreaks naturally end sooner. Substantial outbreaks also can end soon due to faster spreading dynamics and herd immunity of the recovered population.

The epidemic attack rate (AR), as expected, is highly sensitive to vector-to-host ratio, N_V/N_H for the entire range of asymptomatic proportion, τ. For the nominal value of $\tau = 0.2$, doubling the N_V/N_H ratio increases epidemic size by 249.78% (Figure 2.6g). However, if the asymptomatic individuals have very low transmission capabilities, it can limit the effect of vectors on the epidemic size as seen in Figure 2.6i. The fact that most (80%) of the hosts are

asymptomatic in ZIKV infections indicates that they hold a critical role in spreading. The parameters τ and θ both demonstrate the importance in determining epidemic size (Figure 2.6h), and among them, the effect of θ is more radical.

2.4 Discussion

In this study, we have proposed an individual-based interconnected network model for ZIKV that can also be used to simulate any vector-borne disease that features contact-based direct transmissions. We employ heterogeneous mixing based on the host contact network, which is generated based on real-world data on human sexual behavior, sexual orientation, gender, and age structure. We utilize the approximations of the non-Markovian Gillespie algorithm to run stochastic simulations. In the beginning, we explore how the seasons can affect this predominantly vector-borne disease. Later, we focus on the survival of the pathogen in climates similar to Florida if an outbreak starts prior to the colder months. In this step, we examine how some of the important model parameters affect the outcomes. Finally, we perform a sensitivity analysis to evaluate our model behavior in response to changes in some key parameters.

Our seasonal analysis results indicate that outbreak size is strongly related to the environmental conditions during the pathogen introduction. The first few weeks are crucial in determining how much the pathogen would spread. After that period, environmental variations have a much weaker effect in reducing or increasing the outbreak size. If the pathogen is introduced during the peak mosquito season, there is a high probability that we will see a large outbreak. Even if the climatic suitability of mosquito vectors declines rapidly after the first few weeks, the vectors manage to spread the pathogen in the host population rapidly and cause large outbreaks. This suggests that a ZIKV outbreak can spread rapidly out of control if it is not effectively contained during the initial stage. Early interventions are crucial even though it may be challenging to identify outbreaks due to a large asymptomatic group of hosts. The ratio of vector to host, as expected, has shown its prominence in determining outbreak size and the length (Figure 2.6).

We have analyzed how sexual transmission affects the outbreaks. Our results in Figures 2.5c and 2.5f align with the conclusions drawn by some of the earlier works[52–54] that sexual transmission is a small component in the overall force of infection, which is dominated by the vectored-transmission. However, the pathogen still survives up to 75 days in the host network without the help of infected vectors. This prolonged survival can be attributed to a small amount of sexual transmission, but it is mainly due to the extended infectiousness (convalescence) of the hosts as it is assumed that pathogen can survive up to a month in the semen of recovering males. The use of specific birth controls (e.g., condoms) could effectively combat pathogens' spread during this period. These conclusions should inspire further clinical studies in this area to test the efficacy of control measures.

A large number of asymptomatically infected individuals play an important role in outbreak dynamics. The proportion of symptomatic individuals do not affect the survival of the pathogen noticeably. However, it is positively correlated to outbreak size. The relative transmissibility of asymptomatic states is one of the key factors in determining disease outcomes which can extend the vector-free pathogen survival up to 3 months. It is one of the most important parameters that need to be properly estimated in order to obtain informative outbreak predictions.

The conclusions drawn from our results can be useful in evaluating potential endemic scenarios for Zika virus disease in temperate regions. Although the contribution of sexual transmission is small, the pathogen's ability to survive in a human sexual contact network for extended periods can have consequences in sustaining a Zika outbreak in a region and spreading to other regions due to long-distance travels. In addition to that, a large asymptomatic proportion could be the most critical hurdle in controlling ZIKV outbreaks. Although we provide conclusions on the relative importance of key parameters, data unavailability on some of those warrants further estimations. This model should be applicable to other vector-borne diseases which have the potential of being transmitted sexually. Our model is also mesoscale (medium-sized population) in terms of hosts due to the limitations imposed by computational complexities. This work can be extended in the future by the use of activity-driven networks (ADN), vertical transmission, and larger populations. Those studies would provide

more insights into the study of Zika virus epidemics.

2.5 Implementation details

2.5.1 Host network characterization

For the host population, we assume an equal number of males and females. We divide the population into three age groups: Children (0-14), Adults (15-64), and Elderly (65+). Based on the World Development Indicators (WDI) data published for 2017[84], we find that for the USA population, the three age groups are distributed as 19% Children, 66% adults, and 15% elderly people. The adults are the only ones assumed to be capable of transmitting the disease sexually; hence the remaining population is only affected by vectors. Based on the data provided in a sexual behavior study[72], we find that the sexual orientation of men consisted of approximately 97.2% heterosexuals, 2.5% homosexuals, and 0.3% bisexuals. For women, the study revealed 98.9% heterosexuals, 0.9% homosexuals, and 0.2% bisexuals. The same work[72] also compiles a population distribution based on the number of sexual partners listed in Table 2.2. The data indicate that majority of the population has a single partner in a period of 12 months. The average number of partners for the adult population was 1.28.

The network generator tool is designed using the configuration model[5]. It takes all the above-mentioned population and sexual behavior properties as inputs and generates a graph whose statistical properties closely match the given partner distribution of Table 2.2 and the distribution of sexual orientation. Studies on sexual networks also indicate that sexual contact networks have node degree distributions that follow the scale-free structure[10]. There are large variations in the number of sexual partners while a small group of highly active people forms the core[10]. Core groups are essential in sustaining pathogen transmission, especially if the duration of infection is short[85]. To maintain these features, the network generator is designed so that high degree nodes have higher probabilities of connecting to low degree nodes. An example of a network generated by our generation tool is shown in Figure 2.2.

2.5.2 Vector characterization

The mosquito vectors are modeled as homogeneous population and we use the classical Ross-Macdonald approach used by Keeling et al. in their book[48]. We divide the vector population into three compartments, Susceptible (S_V), Exposed (E_V), and Infected (I_V). The transitions between the three compartments are showed in Figure 2.1. Table 2.1 describes the different parameters that were used for the model. The equations for disease dynamics in mosquito vectors are given below,

$$\begin{aligned} \dot{S_V} &= F(t)N_V - \lambda_{VH}N_H(\sum_{j=1}^{N_H}[I_s + \theta I_a]_j)S_V - \epsilon S_V \\ \dot{E_V} &= \lambda_{VH}N_H(\sum_{j=1}^{N_H}[I_s + \theta I_a]_j)S_V - \sigma E_V - \epsilon E_V \\ \dot{I_V} &= \sigma E_V - \epsilon I_V \end{aligned} \tag{2.5}$$

We incorporate seasonality into this model using a time-dependent mosquito recruitment rate, $F(t)$. This rate depends on the time (day) of the year. The transmission parameters, the λ's are computed from the mosquito bite rate, r and transmission probability, T. The formula is given in Table 2.1.

2.5.3 Non-Markovian Gillespie algorithm

The nMGA (Non-Markovian Gillespie algorithm) was proposed by Boguná et al[74]. The following derivation was also described in the work of Masuda et al.[86].

We first consider N renewal processes running in parallel. Let t_i be The time elapsed since the last event of the ith process. We denote $\psi_i(\tau)$ as the probability density function of inter-event times for the ith process. The survival function of the ith process (i.e., the probability that the inter-event time is larger than t_i) is,

$$\Psi_i(t_i) = \int_{t_i}^{\infty} \psi_i(\tau)d\tau \tag{2.6}$$

Now, the probability that no process generates an event for time Δt is,

$$\Phi(\Delta t|\{t_j\}) = \prod_{j=1}^{N} \frac{\Psi_j(t_j + \Delta t)}{\Psi_j(t_j)} \tag{2.7}$$

To determine the time until the next event, Δt, we take a sample u from uniform distribution over $[0, 1]$ and solve $\Phi(\Delta t|\{t_j\}) = u$. This step is computationally expensive when N is large. To improve performance, we approximate this step as proposed by Boguná et al[74]. This approximation is exact as $N \to \infty$. When Δt is small (N is large), equation (2.7) becomes[86],

$$\Phi(\Delta t|\{t_j\}) \approx exp\Big[- \Delta t\Big(\sum_{j=1}^{N} \lambda_j(t_j) \Big) \Big] \tag{2.8}$$

Now, the instantaneous (hazard) rate of the ith process, which is generally assumed to be a function of time since the last event is determined by,

$$\lambda_i(t_i) \equiv \frac{\psi_i(t_i)}{\Psi_i(t_i)} \tag{2.9}$$

With the above equations in hand, we can run the Non-Markovian Gillespie algorithm as follows,

1. Initialize t_j for all $(1 \leq j \leq N)$.

2. Determine the time until next event from,

$$\Delta t = \frac{-\ln u}{\sum_{j=1}^{N} \lambda_j(t_j)} \tag{2.10}$$

3. Select the process i that has generated the event with probability,

$$\Pi_i \equiv \frac{\lambda_i(t_i)}{\sum_{j=1}^{N} \lambda_j(t_j)} \tag{2.11}$$

4. Update the time since the last event, $t_j = t_j + \Delta t$ for all $j \neq i$. Set $t_i = 0$.

5. Repeat steps 2-4.

The original Gillespie Algorithm can be recovered from this nMGA using $\lambda_i(t_i) = \lambda_i$. This is the case for all the parameters that are constant.

2.5.4 Numerical simulation

We use the non-Markovian Gillespie Algorithm (nMGA) to simulate the processes related to host nodes. The vector population is simulated using a deterministic ordinary differential equation (ODE) solver. We combine the hosts and the vector simulations together by calling the ODE solver at the end of each event. The GEMFsim[75] tool, which already supports the Gillespie algorithm, was modified in order to include the vector population model and time-varying parameters. We use the variables X and Y to describe the host and vector populations in different states/compartments, respectively. Here, X contains information about the state of each host node, and Y contains the population count of each vector compartment. X_0 and Y_0 are the initial states/compartments at the start of the simulation. $\mathcal{A}$ is the host network representation. The term $PARAM$ is used to denote the set of model parameters listed in Table 2.1. T_H and T_V in the output are the host and vector indexing arrays that contain information about the time where the data points (X and Y) were generated. We denote $x_n \in X$ as the state of node n. $\lambda_n(x_n \to j)$ is the transition rate of node n from its current state x_n to state j. We use M to denote the total number of host states ($= 4$ in our case). For a node n, Λ_n denotes the sum of transition rates from its current state to all other possible states. The $VECTSOLVER$ is an ordinary differential equation (ODE) solver that takes as input the current situation of the vector population Y and solves them from the current instance to the time δt. The ODE equations that are being solved are given in equation (2.5). This solver is invoked once after each event (with the updated parameters), and the time indexing vectors are updated with respect to the global time t. The simplified pseudo-code of the simulator is given in the next page.

As the simulation is event-based and parameters are updated following each event, the inter-event times should be short to keep the variable parameters up to date in both popula-

input : $X_0, Y_0, PARAM, T_{end}, \mathcal{A}$
output: X, Y, T_H, T_V

$X \leftarrow X_0$ $Y \leftarrow Y_0$ **while** $t \leq T_{end}$ **do**
  for $n \leftarrow 1$ **to** N **do**
    $\Lambda_n \leftarrow \sum_{j=1}^{M} \lambda_n(x_n \rightarrow j)$
  end
  $\Lambda_{tot} \leftarrow \sum_{n=1}^{N} \Lambda_n$
  $u \sim Uniform(0,1)$ // sample a value from uniform distribution
  $\delta t \leftarrow -\dfrac{ln(u)}{\Lambda_{tot}}$ // sample time until the next event
  $P_1(n) \leftarrow \dfrac{\Lambda_n}{\Lambda_{tot}}$
  $k \sim P_1$ // sample a node from activity rate distribution
  $i_k \leftarrow x_k$ // read the current state of the selected node k
  $P_2(j) \leftarrow \dfrac{\lambda_k(i_k \rightarrow j)}{\Lambda_k}$
  $m_k \sim P_2$ // sample the future state of the selected node k
  event $\leftarrow (\delta t, k, i_k, m_k)$ // the event quartet is defined as (*time until the event, participating node, old state, new state*)
  $x_k \leftarrow m_k$
  $I_H \leftarrow$ *count the infected hosts*
  $[Y, t_v] \leftarrow$ *VECTSOLVER* $(Y, PARAM, [0, \delta t])$ // invoke the ODE solver for vector eqns for a duration of δt
  $t_v \leftarrow t_v + t$ $t \leftarrow t + \delta t$
  $T_H \leftarrow [T_H, t]$ $T_V \leftarrow [T_V, t_v]$
  $I_V \leftarrow$ *count the infected vectors*
end

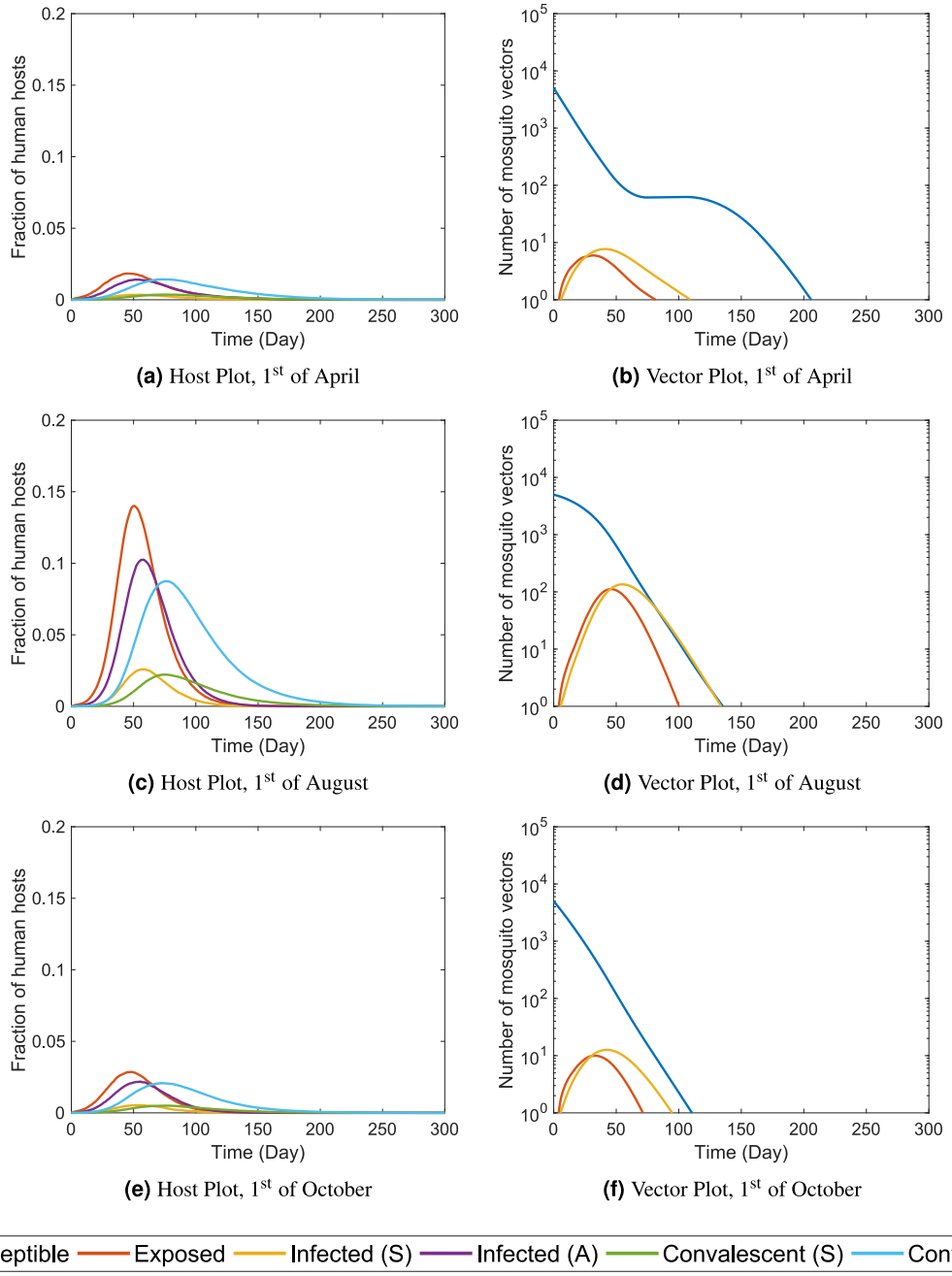
tions. Longer inter-event times in a simulation can increase parameter discrepancy between the host and the vector sub-models. These sub-models are coupled by inter-dependent parameters that vary with time and update after each event. Hence, it is important that events occur at shorter intervals to keep the model outcomes accurate. Fortunately, due to the nature of the Gillespie algorithm, it is indeed the case when a pathogen is present in the host population. We have demonstrated this fact in Figure 2.7 that ensures parameter accuracy. The inter-event times are also exponentially distributed, which is shown in Figure 2.7b. We have a small number of outliers that indicate long inter-event times. Most of these occur when the infection is either very low in the population or about to die out. Hence, the simulation results remain unaffected.

2.6 Code availability

The MATLAB implementation of our model, which was used to generate host networks, run outbreak simulations, and process simulation results, is available for the interested reader at the Network Science and Engineering (NetSE) group website[87] of Kansas State University.

Symbol	Parameter Description	Range	Nominal	Reference
N_H	Total human host population	-	$1,000$	-
N_V	Total mosquito vector population	$1,000$ - $10,000$	$5,000$	Calibrated
λ_{HV}	Vector to host pathogen transmission rate	-	rT_{HV}	As defined
λ_{VH}	Host to vector pathogen transmission rate	-	rT_{VH}	As defined
β	Host to host sexual transmission rate (day^{-1})	0.000164 - 0.0172	0.0087	Agusto[53]
δ	Intrinsic incubation rate in hosts (day^{-1})	1/2 - 1/9	1/7	Zhang[49]
τ	Proportion of symptomatically infected individuals	0.27 - 0.5	0.2	Mitchell[77]
θ	Relative transmissibility of asymptomatic states	0.0 - 1.0	0.5	-
μ	Relative transmissibility of convalescent states	0.0 - 1.0	0.5	-
σ	Extrinsic incubation rate in vectors (day^{-1})	1/7 - 1/10	1/8	Zhang[49]
γ_1	Host recovery rate from infectiousness (day^{-1})	1/3 - 1/7	1/7	Caminade[79]
γ_2	Host recovery rate from convalescence (day^{-1})	-	1/30	Turmel[71]
$A(t)$	Mosquito abundance factor	0 - 1	-	Reiskind[80]
$F(t)$	Mosquito birth rate ($day^{-1}vector^{-1}$)	-	$(1/12) \times A(t)$	As defined
ϵ	Mosquito mortality rate (day^{-1})	1/4 - 1/35	1/12	Gao[52]
T_{HV}	Vector to host pathogen transmission probability	0.214 - 0.8	0.634	Castro[62]
T_{VH}	Host to vector pathogen transmission probability	0.6 - 0.95	0.770	Castro[62]
r	Mosquito bite rate ($host^{-1}vector^{-1}day^{-1}$)	-	b/N_H	As defined
b	Mosquito bite rate ($vector^{-1}day^{-1}$)	0.4 - 0.8	0.63	Castro[62]
$C_{AM}(i)$	Condition: Host i is an adult male	$\{0, 1\}$	-	-
$C_{NAM}(i)$	Condition: Host i is not an adult male	$\{0, 1\}$	-	-
$C_{ST}(i, j)$	Condition: Host i is an adult and j is an adult male	$\{0, 1\}$	-	-
T_{end}	Simulation termination time / Max runtime (day)	-	250	-
M_{start}	Simulation start month / Pathogen introduction month	1-12	4 and 11	Marini[81]

Table 2.1: *Coupled epidemic model parameters and functions.*



(a) Host Plot, 1st of April

(b) Vector Plot, 1st of April

(c) Host Plot, 1st of August

(d) Vector Plot, 1st of August

(e) Host Plot, 1st of October

(f) Vector Plot, 1st of October

Figure 2.4: *The time series plots of a ZIKV outbreaks for three different pathogen introduction months, M_{start} in Miami, FL. The left column contains the host plots and the right column contains the corresponding vector plots. The three rows indicate three different pathogen introduction times: 1^{st} of April, 1^{st} of August, and 1^{st} of October respectively. The hosts in different states are expressed as fractions of the total population. The number of vectors in different compartments are expressed in log scaled axes. Here, we present 5 out of the 7 host states (E, I_s, I_a, J_s, and J_a) and the all three vector compartments (S_V, E_V, and I_V) which are marked with distinct colors described by the legends at the bottom. All plots are averages of 500 independent stochastic simulations.*

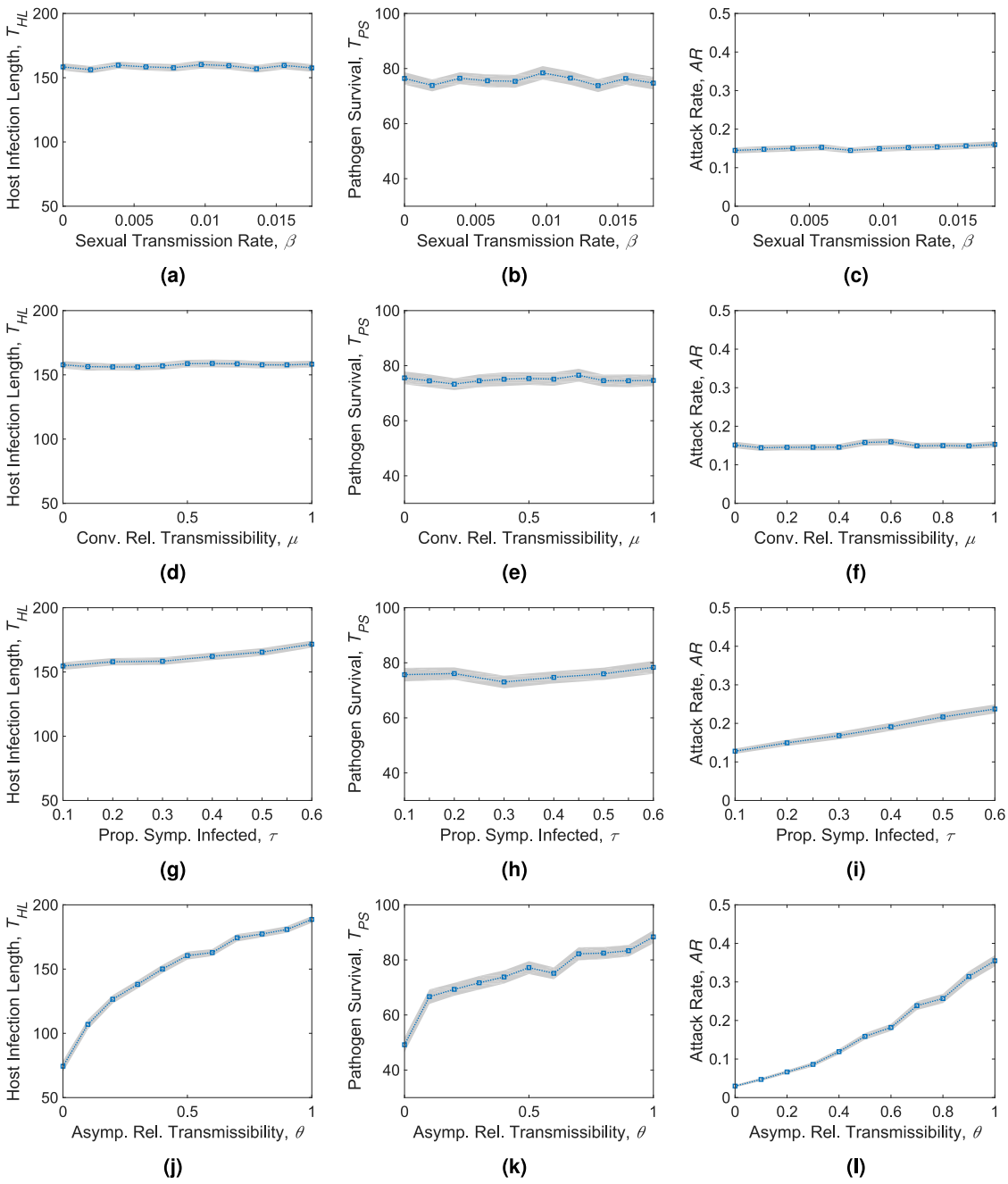


Figure 2.5: *The plots depicting the results of pathogen survival analysis. The results were obtained by varying the sexual transmission rate β (1^{st} row), the relative transmissibility of the convalescent states μ (2^{nd} row), the proportion of symptomatically infected individuals τ (3^{rd} row), and the relative transmissibility of the asymptomatic states θ (4^{th} row). The plots demonstrate the host infection length T_{HL} (1^{st} Col), the pathogen survival T_{PS} (2^{nd} col), and the attack rate AR (3^{rd} col). Each data point (blue square) in the above plots is an average of 500 independent stochastic simulations. The shaded regions indicate 95% confidence intervals.*

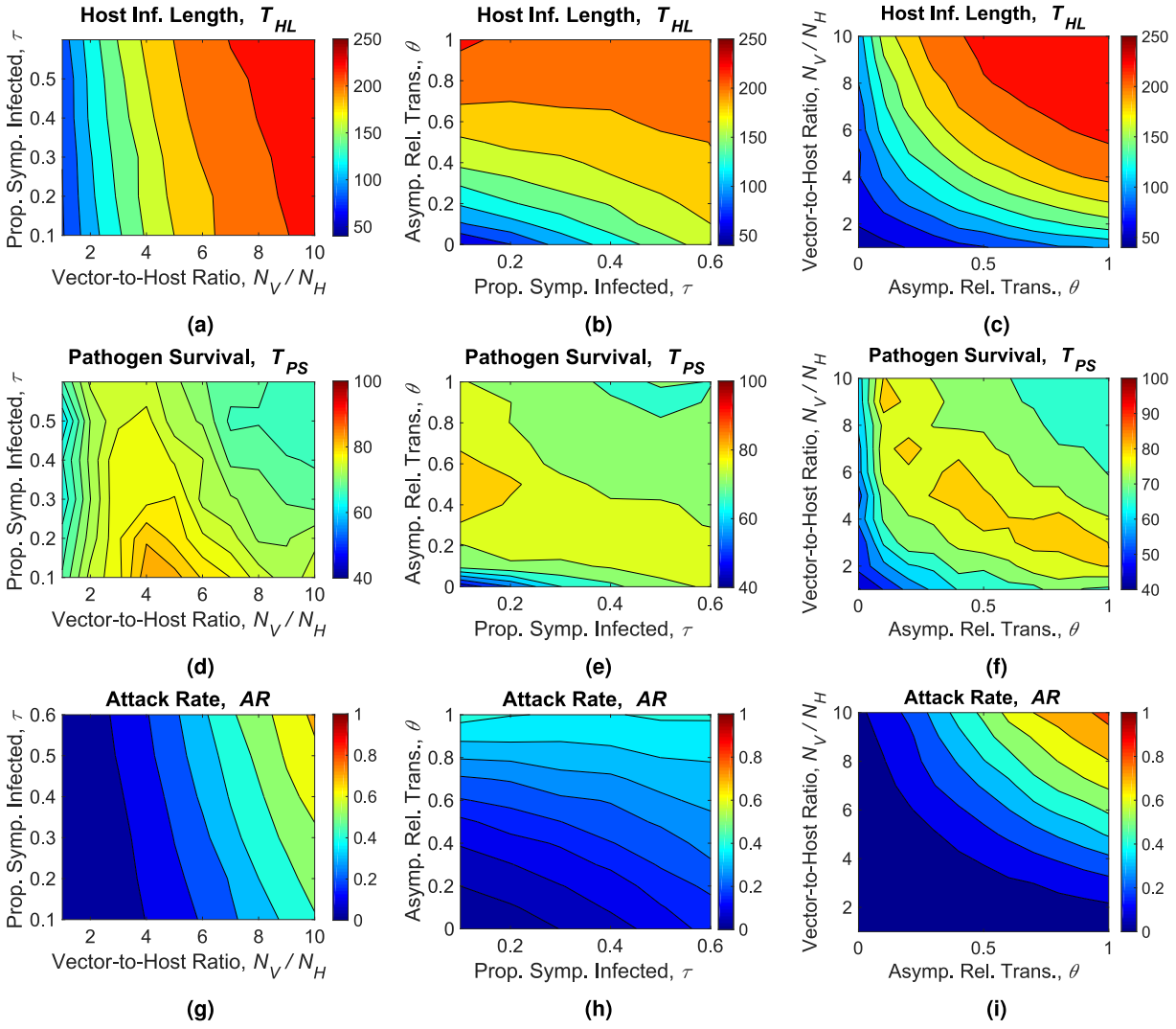


Figure 2.6: *Contour plots depicting the results of the sensitivity analysis. The parameters varied in each plot are marked as axes labels. The quantity for which the contours are being displayed is mentioned on top of each plot. The contours are color mapped and a color-bar on the side of each plot indicates the range of values represented by the plot. The 1^{st} row shows the sensitivity of the epidemic length (T_{HL}) on the parameters, the 2^{nd} row shows the sensitivity of the pathogen survival (T_{PS}) on the parameters, and the 3^{rd} row shows the sensitivity of the epidemic attack rate (AR) on the parameters. The parameters that were varied here are: the vector-to-host ratio (N_V/N_H), the proportion of symptomatically infected individuals (τ), and the relative transmissibility of the asymptomatic states (θ). Each data point shown in the above plots is an average of 500 independent stochastic simulations.*

No. of Partners	% of population
0	15.7
1	71.8
2	5.5
3	2.8
4	1.6
5-9	1.5
10-19	0.4
20+	0.2

Table 2.2: *Average number of sexual partners in the last 12 months from the survey*[72]

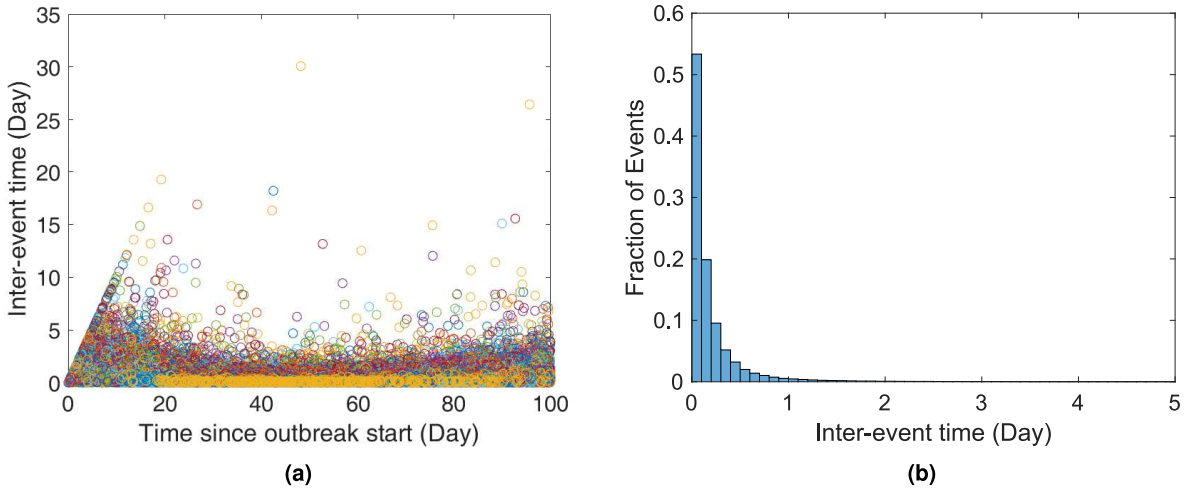


Figure 2.7: *Inter-event times in different stages for all of the 500 simulations during the first 100 days (left) and their distribution (truncated inter-event times > 5) (right). For this case, the average inter-event time is 0.2074 day (95% CI [0.2056 to 0.2092]). About 96.87% of the events that occurred had intervals shorter than a day. There are a few outliers though, which are mostly due to slowing down of the events at the end of the epidemics (when pathogen is low in the host population or have been wiped out).*

Chapter 3

A windowed correlation based feature selection method to improve time series prediction of dengue fever cases

3.1 Background

Accurate time series prediction of dengue fever outbreaks can be useful in planning mitigation strategies for hundreds of tropical and subtropical regions around the world. Data-driven models such as neural networks are flexible in design and can ease the difficulties of estimating unknown parameters that mechanistic models often require[88]. However, the prediction accuracy of such models depends on the quality and the quantity of training data. For dengue fever outbreaks, the availability of incidence data varies across regions. A data aggregation center in a region may not have adequate data to achieve an acceptable level of accuracy in out-of-sample projections. In such cases, selected incidence data from adjacent centers in the same region could improve model performance as additional features. We propose a framework with quantitative methodologies to rank and select nearby case data as supplementary features. Our method uses windowed time-lagged cross-correlation combined with distance and prevalence metrics to identify relevances and potential causal relationships.

Dengue virus is primarily spread by several species of mosquito vectors (*Aedes aegypti* and *Aedes albopictus*), and the outbreaks infect 390 million people every year[89]. The viral strains also cause about 40,000 annual deaths with hemorrhagic fever, and dengue shock syndrome[90]. Dengue virus transmission is prevalent in regions where competent vector mosquitoes are present. In those regions, the mosquito abundance varies throughout the year and depends on factors including air temperature, precipitation, vegetation, and urbanization[91–93]. The availability of extensive data on these factors makes statistical and machine learning analyses feasible. However, researchers experience missing data, uneven reporting intervals, lack of granularity, inadequacy, and inaccuracy with dengue case counts for many regions around the globe. These factors limit the prediction performance of data-driven models. To complicate the situation further, time series outbreak data for most diseases are non-stationary (e.g., the underlying processes evolve with time). In many regions, climatic variations affect *Aedes* vector populations[94]. In addition to that, outbreak start times and sizes may vary because of imported cases caused by short and long-range mobility. Co-circulation of multiple viral strains adds to the complexity. Despite those issues, correlations exist between outbreaks in adjacent human populations (county, municipality, district, etc.) for most communicable diseases, including dengue[95]. While a correlation may not always imply causation, the use of incidence data from adjacent regions can improve the model training because of similarities in meteorological factors, host population density, and a high probability of mobility-based viral transmissions.

For sequential data (e.g., multivariate time series), different methods of feature selection have been used in the past including correlation-based filters[96;97], Granger causality tests[98;99], genetic algorithm[100], and several other methods[101;102]. The applications include forecasts of electrical energy consumption, meteorological variables, financial markets, etc. Correlation-based methods are widely used in machine learning problems[103;104]. Few works extend feature sets for disease outbreak prediction using incidence data from spatially adjacent locations. Such geospatial clustering techniques rely on similarity metrics to improve model performance. A recent work[105] uses pair-wise correlation to cluster location data to train models for COVID-19 outbreak projection in Chinese provinces. Another work tar-

geted towards dengue also uses correlation-based similarity measures to extend the training feature set[106]. However, these implementations assume an instantaneous correlation of incidence data between regions and do not consider the temporal order of outbreaks (whether one location is leading or lagging the other). There can be a considerable amount of time delay between outbreaks occurring in adjacent regions. Time lagged cross-correlation can help identify such phase relations[107]. The phase information may help quantify relationships between outbreaks of adjacent locations. We hypothesize that *if the temporal incidence dynamics of one outbreak lead the incidence dynamics of another, the former outbreak might have a causal influence on the latter*, given that these places lie spatially close enough to affect one another, and the outbreaks are of a reasonable size. We weigh the available features (incidence data) based on a combination of these factors: leading phase correlation, distance, and prevalence. Because of the population's dynamically changing immunity patterns caused by the co-circulation of multiple dengue virus strains, the regions experiencing large outbreaks may also evolve. A single computation of the correlation coefficient over the entire timeline of data may mislead the analysis. Hence, splitting the time series into multiple windows and comparing sequences at each time window for correlation and phase analyses can provide better insights into the dengue outbreak's seasonal patterns. A limitation with some existing methods is that clustering large data sets for model training can make the process complex and inefficient, while unrelated features may deteriorate prediction performance. This phenomenon is known as the curse of dimensionality[19;108]. This work aims to reach optimal model performance with the smallest subset of highly relevant features based on their ranks. There are two broad categories of feature selection methods: filters and wrappers[109]. Wrapper approaches[110] are computationally expensive as the search space for optimal feature subsets increases exponentially with the number of available features. Our method is primarily a filter approach to rank features.

In this work, we present a framework of feature enhancement for data-driven prediction of dengue outbreaks. To achieve that goal, this chapter details: i) a windowed time-shifted cross-correlation method to compute correlation weights, ii) two correlation-window allocation methods, iii) a procedure of ranking feature using metrics based on correlation, distance,

and prevalence, and iv) analysis of prediction performance across windowing schemes, prediction models, and spatial aggregations. This work's novel contribution lies in how we interpret and process incidence data to compute correlation metrics using our knowledge of how outbreaks spread.

3.2 Preliminaries

3.2.1 Definitions

In supervised learning problems, we collect data on multiple variables. A *target* variable (*label*) is the designated output of a machine learning model for prediction. A *feature* is an input variable that is expected to influence the target variable. Each *instance* of data (e.g., a point in time) contains several feature values and usually a single label value. A data set comprises many such instances. For a supervised learning problem, a data set is split into 3 subsets in order to: *train*, *evaluate*, and *test* the models. With the training subset fed in batches (collection of instances), a supervised model learns to predict targets based on features in an iterative process. It optimizes parameters by minimizing a loss function. A neural network comprises artificial neurons (cells) in one or multiple layers. Each neural cell is a node in the network with connections to other nodes across layers, inputs, or outputs. The recurrent neural networks (RNN) are special neural cells that can store internal states in their memory, which helps them predict sequence data (data points that are temporally related) better. Model training aims to get optimal parameter values to generalize beyond the training data and perform well with test data. Sometimes, a model can over-fit the training data and perform poorly with unseen test examples. To prevent such scenarios, we use regularization techniques.

3.2.2 Time series prediction of outbreaks

Time series forecasting is a popular research area because of its applicability in many disciplines, such as forecasting weather patterns, stock prices, market trends, and resource

allocation. In epidemiology, these methods enable the prediction of future outbreaks by fitting models with past disease incidence data and carefully chosen covariates. Such predictions come at varying degrees of accuracy and depend on the characteristics of the target variable, quality of sample data available for fitting, the covariates (predictors) being used, and the models themselves. There is rarely a single model that works best for every application. Classical forecasting methods such as exponential smoothing, autoregressive integrated moving average (ARIMA), and seasonal autoregressive integrated moving average (seasonal ARIMA) have been widely used to predict time series data[111]. The ARIMA model can handle non-stationary data, which is an important advantage. Besides that, the seasonal ARIMA model can incorporate repeating patterns in the data to enable forecasting of diseases that show seasonal patterns[112]. However, these models have tendencies to follow the mean values of past data, and it is not easy to associate these with rapidly changing processes[113]. In addition to that, many classical models require manual tuning of their parameters and may fail to capture complex nonlinear interactions. Data-driven forecasting of vector-borne diseases such as dengue fever is complicated due to complex interactions of several factors with disease dynamics. A list of these factors include but is not limited to seasonally dependent *Aedes* mosquito growth and feeding patterns[114], co-circulations of multiple viral strains[115], environmental (e.g., temperature) effects on dengue virus transmission[116], and human mobility patterns[117]. Neural networks can automatically interpret features from observable variables and can model complex nonlinear phenomena. Hybrid methodologies that combine neural networks with classical models (e.g., ARIMA) are also popular and have been used for dengue outbreak forecasting[118]. In recent times, long-short term memory (LSTM) networks[23] (a type of recurrent neural networks), and its derivatives[119;120] have shown a considerable amount of success in predicting sequential data and has frequently outperformed other classical and machine learning methods[121]. These recurrent architectures have also performed well to predict disease outbreaks[122–124]. Hence, we consider these viable candidates to test the performance of the proposed feature enhancement framework in this chapter.

3.2.3 Factors that affect dengue disease dynamics

Dengue virus is primarily spread by female mosquito vector species: (*Aedes aegypti* and *Aedes albopictus*). Hence, the outbreaks depend on the abundance of such vectors. The relationships between *Aedes* mosquitoes and environmental variables (i.e., temperature, rainfall) are already well characterized by many researchers. Environmental temperature affects the growth, host-seeking, blood-meal intakes of mosquitoes. *Aedes aegypti* cannot develop below 16° C or above 34° C[125]. Within that range, the development from larva to adult was found to be faster at higher temperatures (30° C) compared to lower temperatures (21° C)[126]. *Aedes albopictus* can develop in wider temperature ranges and can survive better in lower temperatures[127] compared to *Aedes aegypti*. The optimum flight temperature for *Aedes aegypti* females was found to be 21° C[128]. Studies have observed that a large diurnal temperature range decreases female fecundity[129]. Temperature fluctuations also affect extrinsic incubation periods (EIP) of dengue viruses. An experiment with DEN-2 strain found that EIP was 12 days at 30° C and reduced to 7 days for 32° C and 35° C[130]. Besides temperature, rainfall has a significant role in dengue outbreaks. Rainwater stuck in different places creates breeding spaces for *Aedes* mosquitoes. Previous works have studied the association of dengue transmission with rainfall[91;131]. In this work, we use several variables, including observed and reanalyzed temperatures, precipitations, relative humidity, and surface-level pressure. These can directly or indirectly affect mosquito vector suitability and dengue outbreak dynamics.

3.2.4 Data collection and processing

Data acquisition

To test the proposed framework for dengue outbreak predictions, we collect data for several regions of Brazil. The InfoDengue project[132] monitors outbreak data on over 700 municipalities of Brazil. Their server contains weekly dengue fever case counts with a surveillance period starting from 2010. Besides dengue incidence data, we collect weather observation

data from NOAA (National Oceanic and Atmospheric Administration) ground weather station database and reanalysis data from the NCEP /NCAR Reanalysis 1 dataset published by the NOAA physical sciences laboratory (PSL) database (NCEP and NCAR stand for National Centers for Environmental Prediction and National Center for Atmospheric Research, respectively). We list all the variables in Table 3.1. We use the weekly case counts as labels and further process the remaining variables to use those as baseline features.

Variable Name	Time Resolution	Space Resolution	Source
Reported dengue fever cases	Weekly	Municipality level	INFO Dengue[132]
Observed Temperatures (min, max, and avg)	Daily	Ground station dependent	NCEI-NOAA
Observed Precipitation	Daily	Ground station dependent	NCEI-NOAA
Avg surface air temperature	Daily	$2.5° \times 2.5°$	NCEP/NCAR Reanalysis 1
Avg surface relative humidity	Daily	$2.5° \times 2.5°$	NCEP/NCAR Reanalysis 1
Avg surface pressure	Daily	$2.5° \times 2.5°$	NCEP/NCAR Reanalysis 1
Avg precipitable water	Daily	$2.5° \times 2.5°$	NCEP/NCAR Reanalysis 1

Table 3.1: *Data collection sources for weather and environmental variables*

Re-sampling and feature engineering

The available raw data cannot be readily used in the machine learning methods. All the features and labels are matched and aligned in both spatial and temporal dimensions. We align data based on available labels (i.e., case data). For each municipality of Brazil (smallest spatial unit available), we search for the nearest ground weather station to collect weather data. We extract reanalysis data from NCEP/NCAR Reanalysis 1 data sets using each municipality's centroid's coordinates. Once the spatial granularity is taken care of, we fix the temporal dimension mismatches by converting all data to match incidence data resolution. As incidence data are available in weekly intervals and the remaining variables are available daily, this process involves context-aware down-sampling of those remaining variables

(temperature, precipitation, humidity, etc.). During this process, we derive 4 additional features from observed weather data of ground stations: average diurnal temperature range of the week, minimum diurnal temperature range of the week, maximum diurnal temperature range of the week, and the number of rainy days in the week. We compute these from daily observed temperature and precipitation data. In total, we have 12 feature variables related to weather and environment: i) 4 observed variables: temperature (average, minimum, and maximum) and precipitation, ii) 4 derived variables based on the time interval (week): diurnal temperature range (average, minimum, and maximum) and the number of rainy days, iii) 4 reanalysis variables: temperature, relative humidity, pressure, precipitable water (all are averages and at earth surface level).

Data splitting and normalization

The complete set of data is a two-dimensional array X with *features* on one dimension and *time* on the other. The set of features consists of: i) the variables listed in section 3.2.4, ii) dengue incidence data of the target location, and iii) dengue incidence data of locations selected as predictors by the methods presented in this chapter. We split X along its time dimension into three parts: training (X_{train}), validation (X_{val}), and test (X_{test}). The validation set is required during the training phase as we implement *early stopping*[133] as a regularization mechanism. We normalize all three sets of data before model training and evaluation. The normalization formula is the following,

$$
\begin{aligned}
\mu_{train} &= MEAN(X_{train}) \\
\sigma_{train} &= SD(X_{train}) \\
\hat{X_{train}} &= (X_{train} - \mu_{train})/\sigma_{train} \\
\hat{X_{val}} &= (X_{val} - \mu_{train})/\sigma_{train} \\
\hat{X_{test}} &= (X_{test} - \mu_{train})/\sigma_{train}
\end{aligned}
\tag{3.1}
$$

All three data sets (training, validation, and test) are normalized using the mean and the standard deviation computed from training data. The complete data set's summary statistics are not used here to prevent the machine learning models from gaining statistical insights about validation and test data sets.

3.2.5 Sequence model specifications

Whether it is training or evaluation, the time series data are split into smaller batches and fed into the recurrent neural network models. From a macroscopic perspective, a sliding window moves over batches of data. Because of the sequential nature of dengue case data, the batches are fed according to the time order without randomization. The sliding window is configured with two integer parameters: input length (t_{in}), and output length (t_{out}). The window is depicted in Figure 3.1. A trainable model would take t_{in} time steps of feature data as input and predict t_{out} time steps of target/label data (e.g., dengue case counts) as outputs every iteration. In the configurations used in this chapter, there is no temporal overlap between input and output sequences. In this configuration, the models make single shot projections (all the t_{out} data points are predicted at once every iteration).

Each batch (window) of data is $(t_{in} + t_{out})$ steps long in the time dimension. A total of 32 batches are stacked together for model training and evaluation. One batch differs from another by a single time-step (hence, there are temporal overlaps between batches). Each batch is further split in time and data (variable) dimensions to separate inputs (t_{in} of features) and outputs (t_{out} of labels).

This work focuses on performance gains obtainable using recurrent neural network (RNN) models due to their proven track record in predicting time series data[121]. A recurrent unit's temporal behavior is illustrated in the lower part of Figure 3.1. We can see that information is passed through time (also called cell state), enabling the model to predict values based on insights gained from past inputs. We use two popular recurrent neural network cell types: LSTM (long-short term memory)[23] with forget gates[24] and GRU (gated recurrent unit)[31]. We also test with a simple linear neural network model as a trivial baseline. Using the

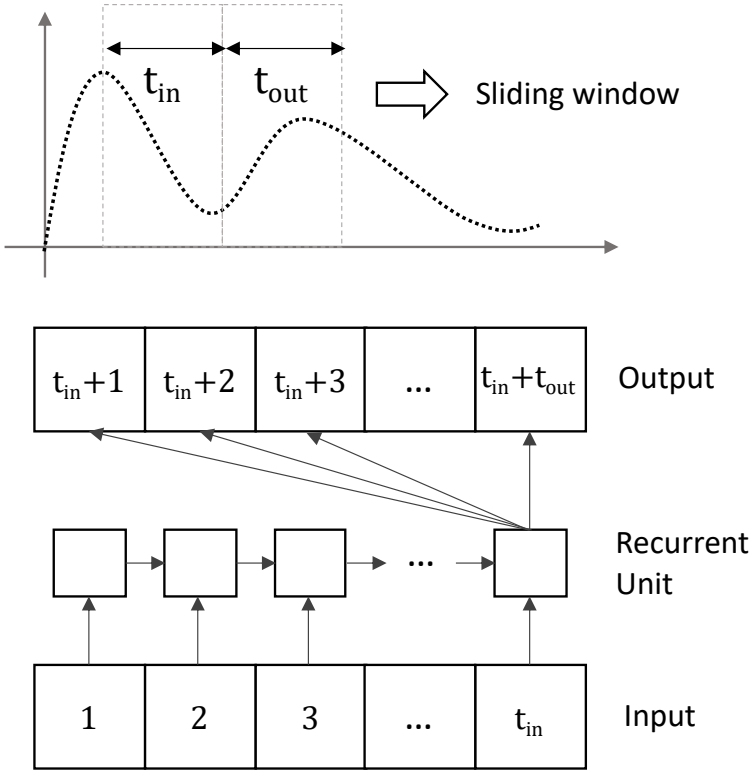


Figure 3.1: *A sliding window defined for feeding of input data and extraction of output case counts. The window slides along the horizontal axis and feeds t_{in} time steps of feature data into the model and extracts t_{out} time steps of predictions.*

TensorFlow[134] package, we configure the models as described below to produce the results:

- **Linear**: The model consists of a single layer of artificial neurons (*Dense* units in TensorFlow) without any nonlinear activation functions. The size of the layer (number of neural units) is equal to the output prediction steps, t_{out}.
- **LSTM**: The model consists of an input layer of 32 long short-term memory (LSTM) units. The output layer consists of a layer similar to the Linear model (described above).
- **GRU**: The model consists of an input layer of 32 gated recurrent units (GRU). The output layer consists of a layer similar to the Linear model (described above).

We initialize the weight metrics of the models as zeros in the beginning. Only the recurrent models (LSTM and GRU) can train and predict based on entire input sequences.

The Linear model predicts based on the last input time step. While training, we use the mean squared error (MSE) as the loss function to optimize the model using Adam optimizer[135]. For predictions, we use the mean absolute error (MAE) metric to evaluate model performance. We regularize our training process with the early stopping[133] mechanism, which monitors loss within validation data and stops training if performance does not improve. Based on our tests on different data sets, we configure the training to run for 120 iterations (epochs).

3.3 Methods

To describe the methodology, first, we define our spatial units. We designate the term *infection center* (IC) to indicate a spatial building block of the model. An IC is a point in the space (regional map) where observed or estimated incidence data on disease outbreaks are available. The spatial granularity of an IC is not fixed for the model. It can be a country, a state, a city, a suburb, or an administrative unit with some resolution in space based on available disease incidence data. The basic structure of our proposed framework is shown in Figure 3.2. The circles inside the shaded region are the infection centers. One of the infection centers, marked as IC, indicates the target infection center where we intend to make predictions of a designated label (e.g., dengue cases). The map's remaining infection centers are marked as *peripheral infection centers* (PIC). These are locations where similar observations on the designated label of the target IC are available. The temporal resolution of the IC and the PICs are aligned before performing any comparative analysis of the data. Some locations may have daily, weekly, or monthly observations. Some data may need re-sampling in the time domain before they can be compared (e.g., convert daily weather data to weekly values).

Assuming that there are N peripheral infection centers (PIC) on the map. Once we match the spatial and the temporal dimensions of the label data (e.g., weekly dengue cases in a city), we use a windowed-time shifted cross-correlation analysis on each $IC - PIC_k$ pair (for all $k \in N$) and compute a correlation weight, $\gamma_C(k)$. We also consider the cumulative cases of each PIC and compute a prevalence weight, $\gamma_P(k)$. Finally, the geodesic distance

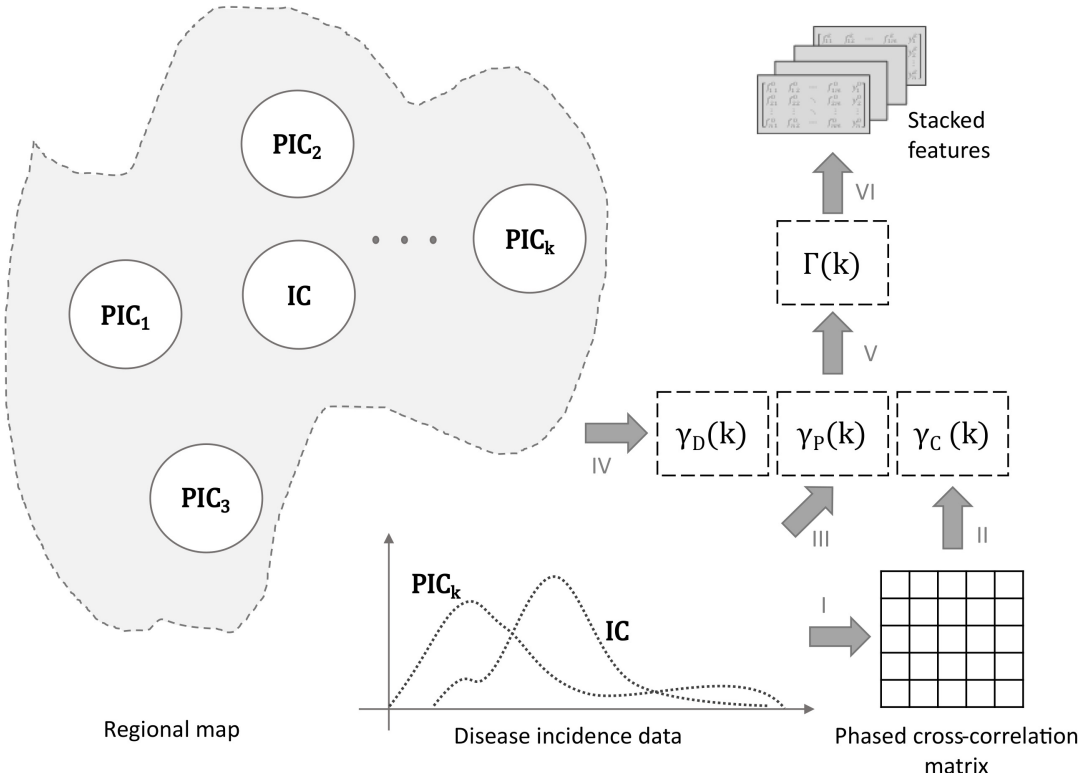


Figure 3.2: *A framework depicting the method to expand trainable and testable data set on dengue incidence. The shaded region enclosed by dashed borders on the left depicts a map of the region of interest. The circles inside the map indicate multiple infection centers (IC) across the region. The target infection center is marked as IC, while the k^{th} peripheral infection center is PIC_k. Using the proposed windowed cross-correlation method (section 3.3.2), a phased cross-correlation matrix is computed (I), which is eventually reduced to a correlation weight, $\gamma_C(k)$ (II). We also compute a prevalence weight, $\gamma_P(k)$, using cumulative case data (III) and a geographic distance weight, $\gamma_D(k)$, using location data (IV). The three weight metrics are combined to compute (V) the predictor metric of PIC_k, $\Gamma(k)$. The PICs are ranked using these Γ values, and their incidence data are selected accordingly to be stacked together with the IC feature set (VI).*

of each $IC - PIC_k$ pair is taken into account in the form of a distance metric, $\gamma_D(k)$. All three metrics are normalized and lie within the range $[0, 1]$ for the selected region. These are combined as following to compute a predictor strength metric for each PIC_k,

$$\Gamma(k) = \gamma_C(k)[\gamma_P(k) + \gamma_D(k)], \forall k \in N \tag{3.2}$$

3.3.1 Windowing incidence data

We compare the time series of disease incidence data (e.g., weekly cases per 100k people) to find correlations. The comparisons are made for all $IC - PIC_k$ pairs with available data for a given region. The key intuition behind this approach is, if a PIC in the region has an infectious outbreak at some point in time, $t = t_0$, this may initiate or affect the course of an outbreak for the target IC at $t \geq t_0$. A leading PIC outbreak may not always imply causal influence depending on the geographic location and population behavior. Despite that, a PIC having an outbreak will influence adjacent ICs, as it acts as an infection source. This also assumes that a strict isolation measure is not in place and the control measures are not 100% effective due to the vector-borne nature of the infection. It is also important to note that, despite seasonal patterns, outbreaks can occur irregularly. A PIC may lead the target IC in one season and lag in other seasons due to complex interactions of multiple viral strains. Hence, we divide the time series into smaller time windows. We propose two methods for windowing incidence data: i) fixed-length window allocation and ii) variable-length window detection. Both of these methods are depicted in Figure 3.3.

A straightforward approach is to divide the entire time series into a fixed number of intervals, M^f. If the time series is T units (day, week, or month) long in total, then a fixed window, w_m^f (where $m \in [0, M^f - 1]$) will have T/M^f units of data. While this is the simplest way to divide the series for correlation analysis, it may not be the most efficient. Setting the appropriate value of M_f remains an open problem, although we apply some intuitions from the seasonality patterns of dengue outbreaks in our case. The fixed windows might not be appropriately placed to contain the time series curve's meaningful dynamics, and some windows may even cover regions without an outbreak.

A second approach is to detect windows based on the time series itself. To do this, we normalize the incidence rate of the target IC to be ranged between $[0, 1]$. A typical outbreak curve has irregularities in its shape that make the analysis cumbersome. We use a Savitzky-Golay filter[136] to smooth out the irregularities while preserving the shape of the outbreaks. Our window detection method has two parameters: the incidence threshold (i_{MIN}) and the

minimum window size (Δ_{MIN}). An window is detected between two time points, t_{START} and t_{END} (where, $0 \leq t_{START} \leq t_{END} \leq T$), if the normalized incidence rate, $i_N(t) > i_{MIN}$ for all $t_{START} \leq t \leq t_{END}$ and $t_{END} - t_{START} \geq \Delta_{MIN}$. A detected window, w_m^d will have a length (greater than or equal to Δ_{MIN}) that depends on the time series curve characteristics. The number of detected windows, M^d, will also vary for the same reason. Figure 3.3 shows both methods in action using time series data for Brazil's municipality between 2010-2015. For the assigned value of $M^f = 5$, we get equally sized windows, each of which is 1 year in length. With our detection method, we find 4 windows ($M^d = 4$) that indicate 4 separate outbreaks ($\Delta_{MIN} = 10$ weeks, $i_{MIN} = 0.05$).

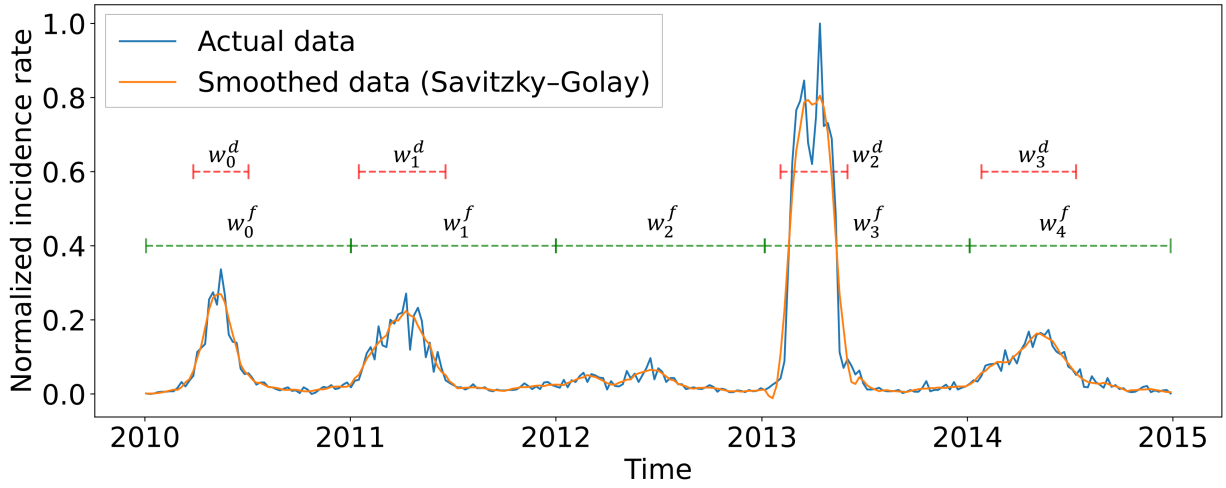


Figure 3.3: *Windowing methods used for data segmentation before computing time-shifted correlation coefficients. We present the normalized incidence data from the Cachoeiro de Itapemirim municipality of Brazil*[132]*. The fixed windows (w_m^f) are marked in green, and the detected windows (w_m^d) are marked in red. We set $M^f = 5$ to get 5 fixed windows, each comprising 1 year of data. For the detected windows, we configure $\Delta_{MIN} = 10$ weeks and $i_{MIN} = 0.05$. A Savitzky-Golay*[136] *smoothing is applied to the data before window detection takes place.*

Once the windows are selected (either by assignment of fixed number or detection), the following procedures are identical. Hence, we will ignore the superscripts (f and d) in this chapter's next sections for brevity. M will depict the total number of windows. w_m (where $m \in [0, M-1]$) will depict the $(m+1)^{th}$ window.

3.3.2 Time-shifted cross correlation

Let $i_0(t)$ and $i_k(t)$ be the disease incidence (new cases at time step t) of the target IC and the k^{th} PIC respectively. We perform bivariate computations of time shifted Pearson's correlation coefficients[137] using windowed (w_m) incidence data of the target IC ($i_0^{w_m}(t)$) and the k^{th} PIC ($i_k^{w_m}(t)$). The formula used to compute the coefficients is shown in Equation 3.3. This measure is also known as time lagged (or phased) cross-correlation[138] in statistics and signal processing.

$$R_k^{w_m}(\theta) = r(i_0^{w_m}(t), i_k^{w_m}(t-\theta)) \tag{3.3}$$

Here, $R_k^{w_m}(\theta)$ is the correlation coefficient computed for subsets of the time series $i_0(t)$ and $i_k(t)$ selected by the time window w_m when one series is shifted by an amount θ with respect to another. The Pearson's correlation coefficient function is indicated by $r()$ in Equation 3.3.

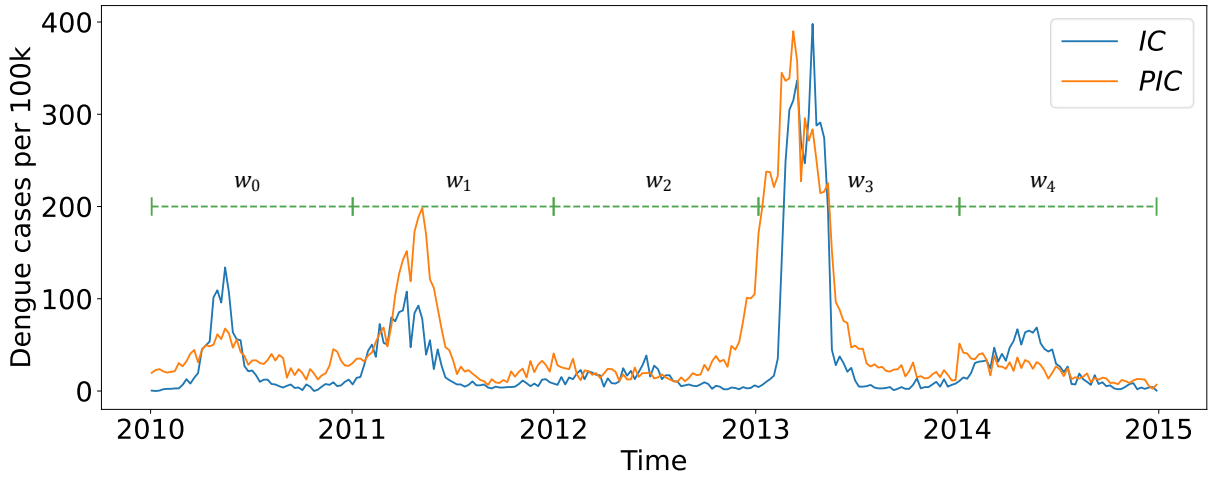


Figure 3.4: *Windowing of IC and PIC_k incidence data for computing time-shifted cross-correlation coefficients. We allocate a fixed number of windows ($M^f = M = 5$) for the time range 2010-2015, making each window 1 year long (52 weeks approximately). The two curves shown here correspond to the target IC ($i_0(t)$) and the k^{th} PIC ($i_k(t)$) and these are from the municipalities: Cachoeiro de Itapemirim and Vitória respectively[132]. The m^{th} time window is marked by the symbol w_m. Note, in the first window (w_0), the outbreaks of IC and PIC are almost in phase, whereas in the second window (w_1), the PIC outbreak is lagging in time compared to the IC.*

For the two time series curves shown in Figure 3.4, the time-lagged correlation matrix (obtained by computing $R_k^{w_m}(\theta)\ \forall\ m \in [0, M-1]$ and $\theta \in [-8, 8]$) is plotted as a heatmap in Figure 3.5. We use the location depicted in Figure 3.3 as the target IC ($i_0(t)$) and another location from the same state (Espirito Santo) of Brazil as the k^{th} PIC ($i_k(t)$). For this demonstration, the time series curves were split using fixed length windows ($M^f = M = 5$). The heatmap depicted in Figure 3.5 can be visually verified by comparing with Figure 3.4. As expected, the two curves are almost in phase in w_0 , negatively correlated in w_2, IC leads in w_1, and PIC leads in w_3 and w_4.

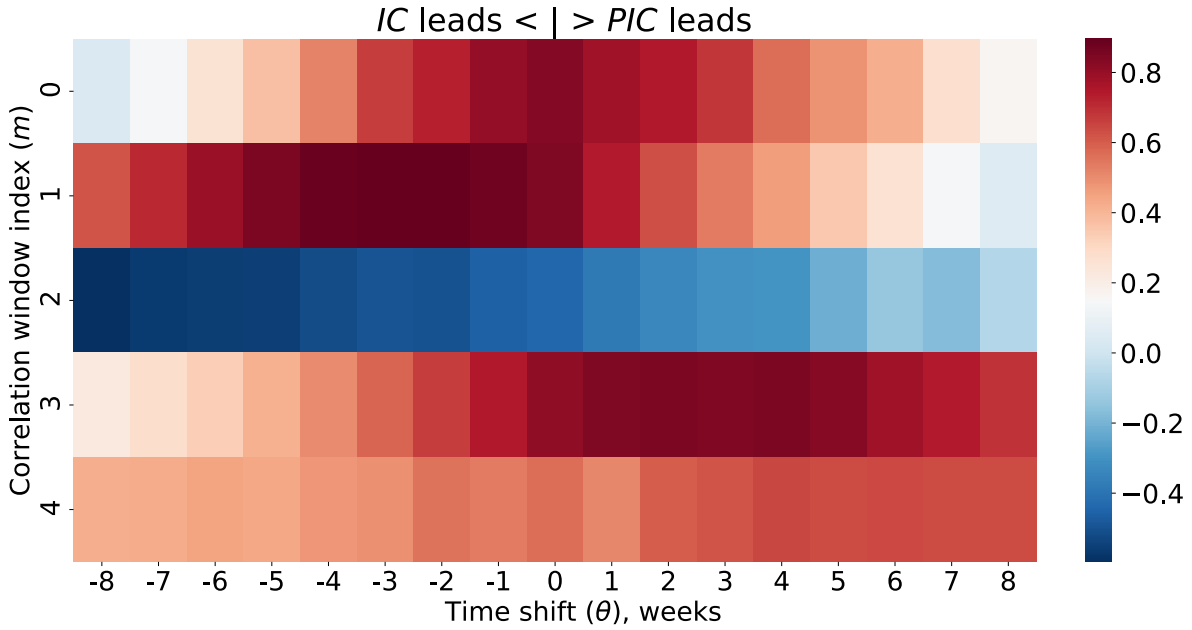


Figure 3.5: *Heatmap of the computed time-shifted cross-correlation matrix (Equation 3.3) for the IC and the PIC in Figure 3.4). The vertical axis depicts the window indices ($m \in M$) and the horizontal axis depicts time shift (phase), θ that ranges from -8 to $+8$ weeks. The color shades of the heatmap depict the correlation values demonstrated by the gradient bar on the right. Higher correlation values on the left of the midpoint ($\theta < 0$) indicate that IC is leading in the outbreak curve (Figure 3.4) compared to the PIC. Higher correlation values on the right of the midpoint ($\theta > 0$) indicate the opposite (PIC leads IC).*

Heatmaps like Figure 3.5 are computed for all PIC_k with $k \in [1, N]$. To determine if a PIC_k is leading in a window (w_m), we find the location (θ) of the peak correlation as shown in Equation 3.4.

$$\theta_k^{w_m} = \underset{\theta}{\operatorname{argmax}}\, R_k^{w_m}(\theta) \tag{3.4}$$

We define the correlation strength $S_k^{w_m}$ to be the mean correlation measure computed around the peak ($\theta_k^{w_m}$), extending by the amount θ_E in both directions (Equation 3.5).

$$S_k^{w_m} = \frac{1}{\sum_{\theta=-\theta_E}^{\theta_E} 1} \sum_{\theta=-\theta_E}^{\theta_E} R_k^{w_m}(\theta_k^{w_m} + \theta) \tag{3.5}$$

Equation 3.5 has an additional condition on the values θ such that $R_k^{w_m}(\theta_k^{w_m} + \theta)$ exists for the given parameters. A PIC_k leads the IC if the peak of correlation lies on the right half of the heatmap shown in Figure 3.5, which translates to $\theta_k^{w_m} > 0$. We only consider if a PIC_k is at least in phase with the IC and discard the cases where any PIC_k lags the IC. This is how we compute the predictor probability matrix P with dimensions $M \times N$. The individual predictor probabilities ($\forall m \in [0, M-1]$, $\forall k \in [1, N]$) are are computed as shown in Equation 3.6.

$$P_{m,k} = \begin{cases} S_k^{w_m}, & \text{if } \theta_k^{w_m} \geq 0 \\ 0, & \text{otherwise} \end{cases} \tag{3.6}$$

The predictor probabilities are averaged across all windows (Equation 3.7) to compute overall predictive abilities of all PIC_k. For a particular region ($k \in [1, N]$), the predictive ability metrics are normalized between [0,1]. The final measure, $\gamma_C(k)$, is defined as the *correlation weight* of PIC_k as shown in Equation 3.8.

$$\hat{\gamma_C}(k) = \frac{1}{M} \sum_{m=0}^{M-1} P_{m,k} \tag{3.7}$$

$$\gamma_C(k) = \frac{\hat{\gamma_C}(k) - \min_k \hat{\gamma_C}(k)}{\max_k \hat{\gamma_C}(k) - \min_k \hat{\gamma_C}(k)} \tag{3.8}$$

3.3.3 Distance and prevalence metrics

With increasing distance, the impact of a PIC on the target IC is likely to be reduced due to decreased travel between the locations, increasing differences in environmental conditions (e.g., temperature, rainfall, vegetation), etc. It is intuitive to use a metric proportional to the inverse distance for strengthening the predictive ability measures of PICs. Let, d_k be the geodesic distance[139] (shortest path on the surface of the earth, assuming earth to be an ellipsoid) between the target IC and PIC_k. The normalized $[0, 1]$ distance of a PIC_k in the region is,

$$\hat{d_k} = \frac{d_k - \min_k d_k}{\max_k d_k - \min_k d_k} \tag{3.9}$$

We want the metric to be inversely proportional to the distance. Hence, the distance metric of PIC_k is defined as,

$$\gamma_D(k) = 1 - \hat{d_k} \tag{3.10}$$

The outbreak history of a location is an important criterion that indicates the viral pathogen and endemic scenarios' persistence. For a PIC_k, we compute the prevalence I_k by taking a sum of the incidence data $i_k(t)$ for the entire timeline ($\forall t \in [0, T]$).

$$I_k = \sum_{t=0}^{T} i_k(t) \tag{3.11}$$

The prevalence metric is normalized $[0, 1]$ across the region.

$$\gamma_P(k) = \frac{I_k - \min_k I_k}{\max_k I_k - \min_k I_k} \tag{3.12}$$

3.4 Results

We present here the results in several stages. The outcomes of the feature analysis are presented first. This is followed by an analysis of prediction performance using the proposed

methods. The results are generated using municipality-wise weekly dengue case data between 2010-2019 from Brazil's Espírito Santo state. We obtained data for 78 municipalities of Espírito Santo and ranked them based on the total number of cases recorded for the entire time period. The top 5 municipalities based on prevalence are listed in Table 3.2. The location IDs shown in the table are the IBGE (Instituto Brasileiro de Geografia e Estatística) codes for Brazil[140].

Loc. ID	Name	Cases	Cases per 100k
3205309	Vitória	71,348	19,501.72
3205002	Serra	58,424	11,081.10
3201209	Cachoeiro de Itapemirim	45,319	21,520.12
3205200	Vila Velha	36,743	7,329.18
3201308	Cariacica	27,103	7,059.60

Table 3.2: *Top 5 locations of Espírito Santo ranked by total reported cases of dengue during 2010-2019*[132].

3.4.1 Feature selection and analysis

The PICs are sorted and ranked for each IC, based on the predictor strength metric, Γ (Equation 3.2), which is computed from the three individual metrics: correlation weight (γ_C), prevalence weight (γ_P), and distance weight (γ_D). We choose the municipality of Vitória in Espirito Santo, Brazil, as the target IC, which had the highest total number of cases in the state during 2010-2019, to generate the results. We compute the correlation weight using 20 fixed-length windows ($M^f = M = 20$) for the time range 2010-2019, making each window approximately 26 weeks (6 months) long. The top 5 ranked PICs based on Γ are listed in Table 3.3 with the corresponding weights. While our method prioritizes the correlation weight more than others, PICs with relatively lower correlation weight can still be favored because of the following factors: i) having a significant number of cases or ii) being in proximity of the target IC. This is evident in Table 3.3 as 3201209 (PIC #1) is chosen over 3205200 (PIC #2) and 3205101 (PIC #4) is chosen over 3200607 (PIC #5). *Note, the numbers (#) used in 'PIC #' indicate rank. This should not be confused with arbitrary indices (k) used to compute metrics (PIC_k).* This effect is also illustrated in Figure 3.6,

which shows the ranked PICs listed in Table 3.3. Although PIC #1 lies farthest from the IC among the five (lowest γ_D), it is ranked at the top due to significantly higher values in the other two factors (γ_C and γ_P).

Rank #	Loc. ID	Corr. γ_C	Prev. γ_P	Dist. γ_D	Γ
1	3201209	0.912	0.812	0.614	1.301
2	3205200	1.000	0.270	1.000	1.270
3	3201308	0.956	0.260	0.996	1.202
4	3205101	0.621	0.770	0.937	1.060
5	3200607	0.772	0.524	0.809	1.029

Table 3.3: *Top 5 predictor PICs for Vitória. The weights based on our defined predictability metrics (correlation, prevalence, and distance) are shown in columns 3-5. The combined weights (Γ) are shown in the last column.*

A time series plot in Figure 3.7 shows that the top PICs are mostly correlated with the IC, Vitória. Among the PICs displayed here, PIC #4 (3205101) shows the weakest correlation with the IC. It will be clear in the upcoming results, the proximity and the high incidence fraction of this location help with prediction performance. The variability of the incidence curves prevents our method from classifying a single PIC as the optimal predictor for all time ranges. However, the combination of top-ranked PICs will improve prediction accuracy.

3.4.2 Prediction performance

After selecting features with the proposed methods, we train and evaluate the prediction performances using the three models (Linear, LSTM, and GRU) described in section 3.2.5. The time series data are split into 3 distinct sets with ratios of 50:30:20 for model training, evaluation, and testing. A sliding window with input length (t_{in}) of 8 and output length (t_{out}) 4 is used for a single shot prediction of the next 4 weeks using data of the past 8 weeks every step.

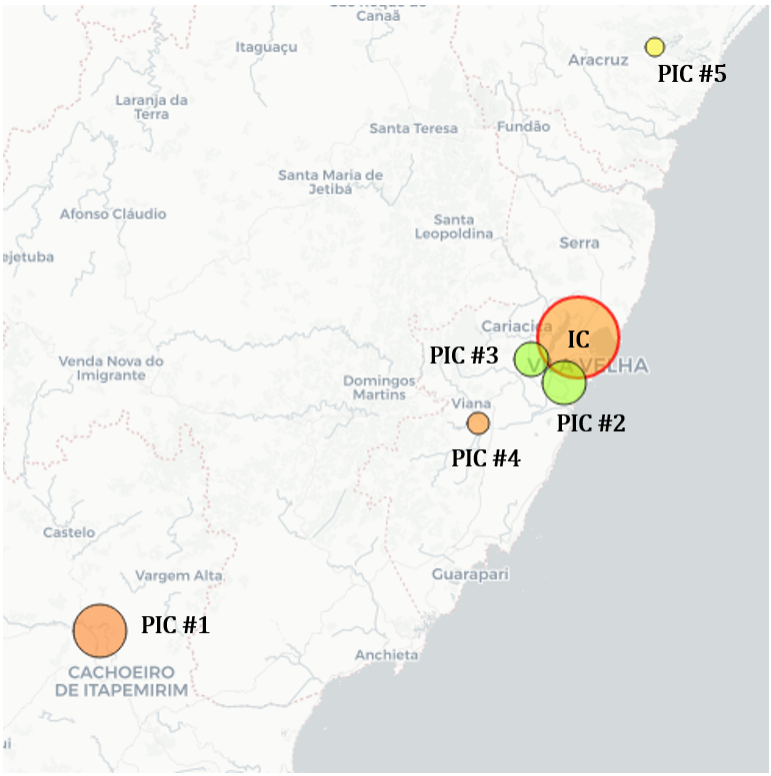


Figure 3.6: *A geospatial map showing an incidence center (IC) and 5 ranked peripheral incidence centers (PIC). The circles' radii are proportional to the total number of cases reported during 2010-2019*[132]*. The shades of the fill color are generated from a color gradient (green-yellow-red) which is proportional to the fraction of cases with respect to the local population of each location (IC or PIC) during 2010-2019*[132]*. The greenish shades indicate smaller infected fractions, while the reddish shades indicate larger fractions. Map generated using Folium*[141] *with OpenStreetMap*[142]*. Basemap tiles provided by CartoDB*[143]*.*

Individual *IC* prediction

For the *IC* of Vitória, the *PIC*s are added gradually according to their computed ranks (section 3.4.1), and the models' prediction performances are evaluated. The mean absolute error (MAE) values on the normalized test data are plotted in Figure 3.8 for varying number of additional features, N_{PIC}. For both LSTM and GRU models, the addition of the first two PICs deteriorates the model performance. However, the subsequent additions keep improving the outcomes. The plots quickly reach their minima, after which errors increase. The first few additions increase error due to high variability in the incidence data, as evident in Figure 3.7. Further additions of *PIC* create averaging effects on the predictor data set and cause performance improvements. The GRU model eventually reaches a minimum

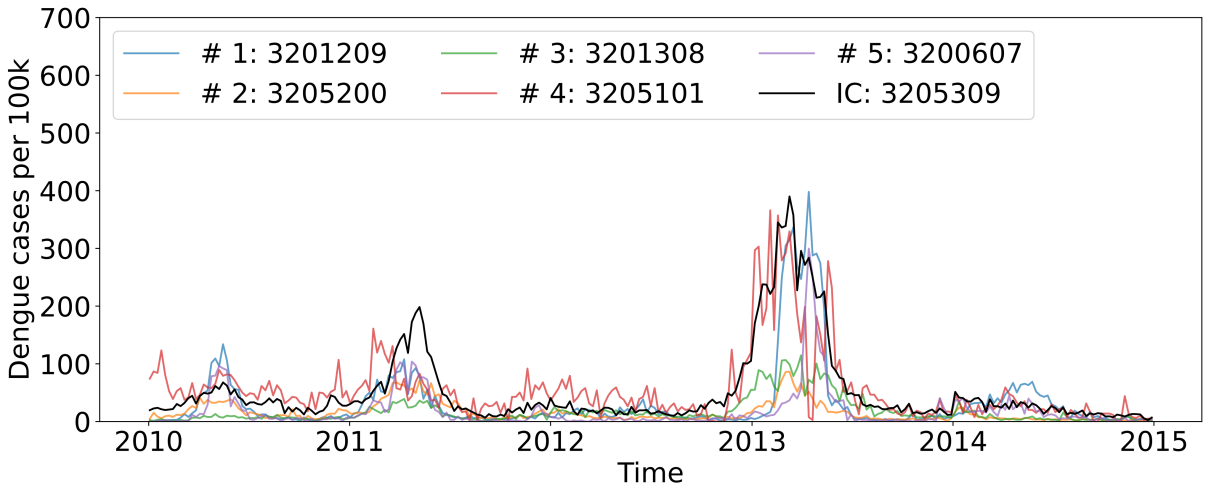


Figure 3.7: *Time series plots of weekly dengue cases per 100,000 people for the IC and top 5 ranked PICs (Table 3.3). The plots depict cases only between 2010-2015 for improved clarity, but the metrics were computed based on the entire series (2010-2019).*

MAE value of 0.128, which is lower than the best optimal LSTM prediction (0.1415) by about 9.54%. The Linear model reaches an optimum MAE value of 0.3384, which is nowhere close to the recurrent models. After reaching the minima, all three error curves rise again. This increase in MAE with larger feature sets (N_{PIC}) can be attributed to the curse of dimensionality[19;108]. LSTM and GRU models perform optimally with 4 and 6 additional *PIC*s, respectively. Predicting based on present input and historical context (internal states of LSTM and GRU) of data certainly puts recurrent models ahead in performance, which is evident even without a *PIC* ($N_{PIC} = 0$ in Figure 3.8) in the feature set. However, the linear model significantly benefits from feature addition as case data from *PIC*s strengthen inductive bias.

After determining the optimum number of features (N_{PIC}) to be added to the predictor data set for an *IC*, recurrent models are trained with the extended feature set and are used to predict dengue cases for the test data subset of the time series. Figure 3.9 compares the predictions with actual incidence data for both LSTM and GRU models using Vitória as the *IC* and choosing the optimum number of *PIC*s for the two models ($N_{PIC} = 4$ for LSTM and 6 for GRU).

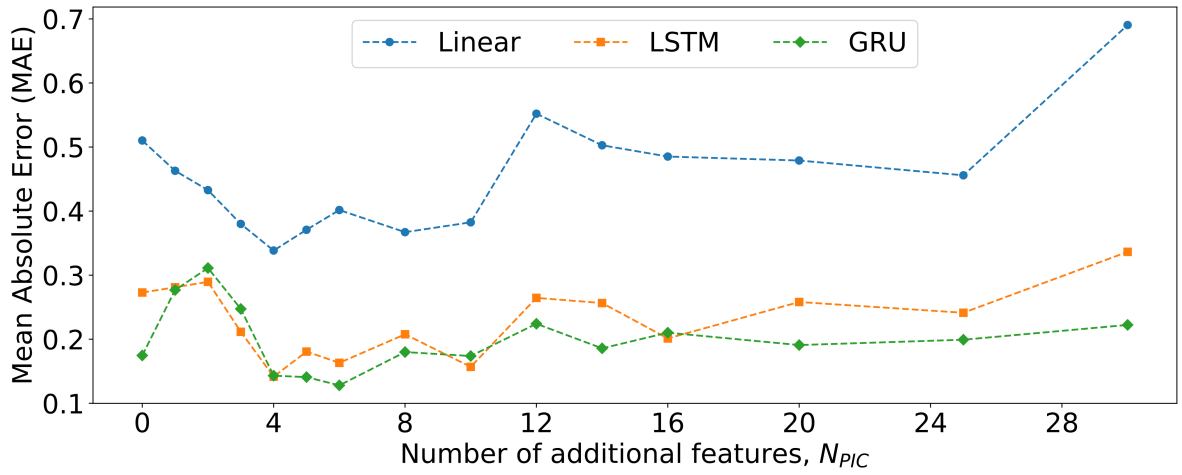


Figure 3.8: *Prediction performances of the Linear, LSTM, and GRU models in predicting normalized test data for varying number of features (N_{PIC}). PIC data are added to the feature set based on ranks dictated by computed predictor strengths (Table 3.3), after which models are trained and evaluated over the test data to compute the mean absolute error (MAE) metrics (lower is better).*

Effect of window selection methods

The impacts of various correlation window configurations are analyzed under the two proposed windowing methods: i) fixed-length window assignment and ii) variable-length window detection. For the first method (fixed), we vary the number of windows (M^f) between 5, 10, 20, and 40. For the second method (detection), we vary the minimum window size (Δ_{MIN}) between 5, 10, 20, and 30 weeks. In both methods, we are consequently varying the number of windows, window lengths, and where the windows are located. In total, we run tests under 8 different scenarios and compare the prediction performances using the 3 models (Linear, LSTM, and GRU). Instead of working with a single IC, we run the tests over multiple locations and report the average performance metrics (e.g., mean of MAE). We take the top 20 ICs based on total cases per 100,000 people and average the performance metrics across ICs. Among the 20 ICs, there were 4 ICs where none of the models could predict with reasonable accuracy (MAE < 1) with or without additional features. In those locations, either the data were too limited or the outbreaks were too random for our models to generalize beyond training data. We filter out these 4 locations and take the remaining 16 locations to evaluate our methods.

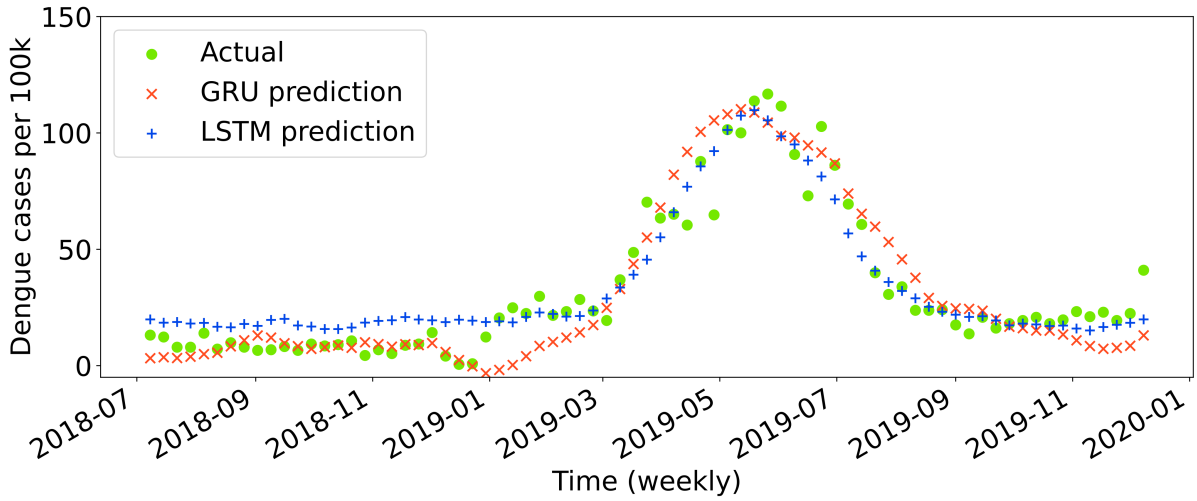


Figure 3.9: *Predicted weekly dengue cases per 100,000 people using test data with the recurrent models (LSTM and GRU). These results were obtained for the IC, Vitória, with 4 and 6 additional PIC data joined with LSTM and GRU's feature set, respectively.*

For a model, predicting on a given IC, the optimal MAE is defined as the minimum mean absolute error (MAE) obtained by varying the number of additional features (N_{PIC}) from the PIC ranked list produced by a given windowing method (i.e., minima of the curves in Figure 3.8 are the optimal MAEs of three models). The average optimal MAE (across 16 ICs) are plotted in Figure 3.10 for all 8 windowing schemes. The recurrent models perform significantly better than the Linear model: on average, LSTM and GRU outperform the Linear model by 60% and 61%, respectively. LSTM and GRU outperform the best-performing linear model by 41% and 43%, respectively. We expect this kind of advantage, given the importance of considering historical data in predicting future dengue cases. The performances across different windowing schemes with LSTM and GRU models are comparable (both models have an approximate standard deviation of 0.02 around their means of 0.3569 and 0.3435, respectively), with a small trend of increasing mean error values with windowing method configurations. In the case of linear models, there are some stark contrasts when using window detection methods. Using the window detection method with a large Δ_{MIN} value increases performance sharply. For example, using a Δ_{MIN} of 30 weeks shows 41.9% improvement over fixed window schemes' average performance. This indicates that features selected with a small number of large correlation windows detected based on

outbreak locations in the time series are more effective than the remaining schemes that use a relatively large number of smaller windows. A similar trend is also visible with fixed window methods, although it is not statistically conclusive. The lower values of M^f, which translates to having a lower number of large windows, show slightly better performance over higher values of M^f.

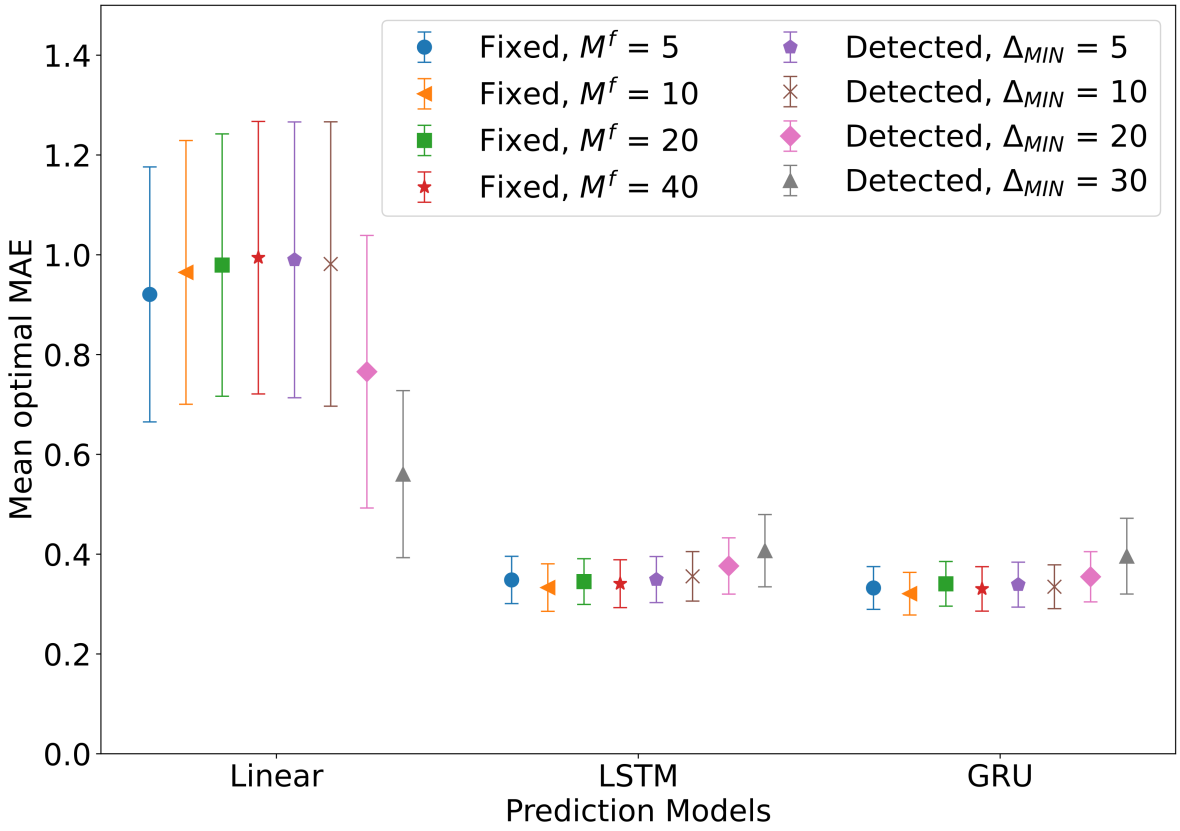


Figure 3.10: *A comparison chart of the lowest prediction error obtainable using different fixed and detected windowing schemes. Four fixed windowing schemes with the number of windows, M^f = 5, 10, 20, and 40, along with four window detection schemes having minimum window lengths, Δ_{MIN} = 5, 10, 20, and 30 weeks are compared for three models. The vertical axis denotes the average of the optimal MAE values. The markers denote the mean values, with the bars representing standard errors of the mean. The results are the averages of 16 top ICs based on prevalence.*

To understand how beneficial our proposed methods are over the baseline (without feature enhancement, $N_{PIC} = 0$), we analyze the improvements in prediction accuracy. The term, *improvement in accuracy* is defined as the relative decrease in MAE (i.e., increase in prediction performance) with optimum PIC selection (Figure 3.10) compared to MAE with-

out additional features ($N_{PIC} = 0$). We compute the accuracy improvements across all 16 ICs and plot the *mean improvement in accuracy* as percentages in Figure 3.11. The linear model demonstrates average improvements ranging from 18.97% to 27.13% depending on the window selection schemes. The LSTM model improvements range from 22.13% to 33.6% and the GRU model range from 10.79% to 31.92% depending on the window selection schemes. We should be careful in interpreting these values; greater improvements do not translate to better optimal performance. These values are relative to their baselines ($N_{PIC} = 0$). With some incidence centers (IC), a model can already predict with higher accuracy than other incidence centers. Our methods help minimize those gaps and still provide some improvements over the baselines (ranging from 10.79% to 33.6% depending on the model and the scheme). Figure 3.10 demonstrates that, in general, GRU and LSTM both perform well when we deal with average values. For individual ICs, however, a conclusive determination of the best performing model can be done (either GRU or LSTM).

Performance on aggregated data

It is sometimes reasonable to aggregate data to larger scales based on geographic adjacency and environmental similarities (e.g., weather). From a macroscopic point of view, predicting for an ecoregion may be more meaningful for policymakers to interpret the outcomes. According to Omernik (2004), ecoregions are defined as areas within which there is a spatial coincidence in characteristics of geographical phenomena (e.g., geology, physiography, vegetation, land use, climate, hydrology, terrestrial and aquatic fauna, etc.) associated with differences in the quality, health, and integrity of ecosystems[144]. We use the terrestrial ecoregions defined by The Nature Conservancy(TNC) in this work[145]. The following analysis is performed for Brazilian locations with available data in the Bahia Coastal Forest ecoregion.

Comparing the prediction performance on aggregated data shows that the advantages of recurrent models compared to the Linear model (which we had for individual ICs) are diminished. The optimal performances across prediction models become more comparable, as shown in Figure 3.12. The mean optimal MAE values obtained for 4 fixed window schemes

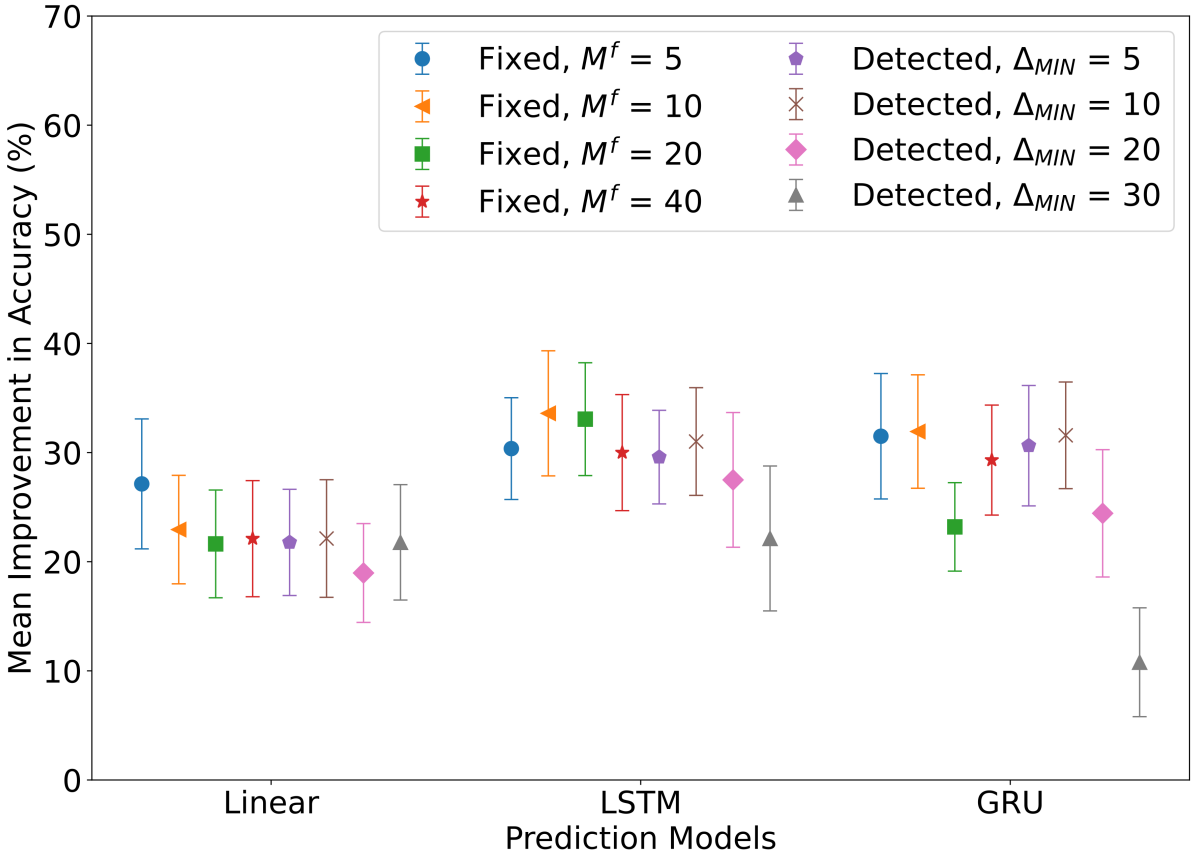


Figure 3.11: *A comparison chart of average improvement in accuracy (reduction in MAE) with respect to performance without feature enhancements ($N_{PIC} = 0$) for different fixed and detected windowing schemes. Four fixed windowing schemes with the number of windows, M^f = 5, 10, 20, and 40, along with four window detection schemes having minimum window lengths, Δ_{MIN} = 5, 10, 20, and 30 weeks are compared for three models. The markers denote the mean values, with the bars representing standard errors of the mean. The results are the averages of 16 top ICs based on prevalence.*

are 0.38, 0.40, and 0.36 when using Linear, LSTM, and GRU models, respectively. The mean optimal MAEs for 4 window detection schemes are 0.31, 0.35, and 0.33 using the same three models. We observe a distinct advantage of the window detection method for selecting *PIC*s. On average, the window detection methods improve over their baselines ($N_{PIC} = 0$) by 16.50%, 12.68%, and 10.07% for Linear, LSTM, and GRU models, respectively. Variable window allocation based on outbreak window detection outperforms fixed window allocation methods. In other words, windowed cross-correlation on a few important outbreak regions performs better than comparing over the entire time series. As the target data is aggregated

from all PICs in the region, the optimally trained Linear models sometimes perform better than recurrent models. One key takeaway is that cases for the entire region can be predicted with improved accuracy while using a small subset (PICs) of data as features. For the results shown in Figure 3.12, the optimal MAE values can be reached for N_{PIC} values between 2 and 5. This aggregation provides a fast way to predict cases, with a small fraction of regional outbreak data. While the accuracy achieved at the ecoregion level is not on par with the accuracy achieved for single ICs, this prediction is useful to generate risk maps.

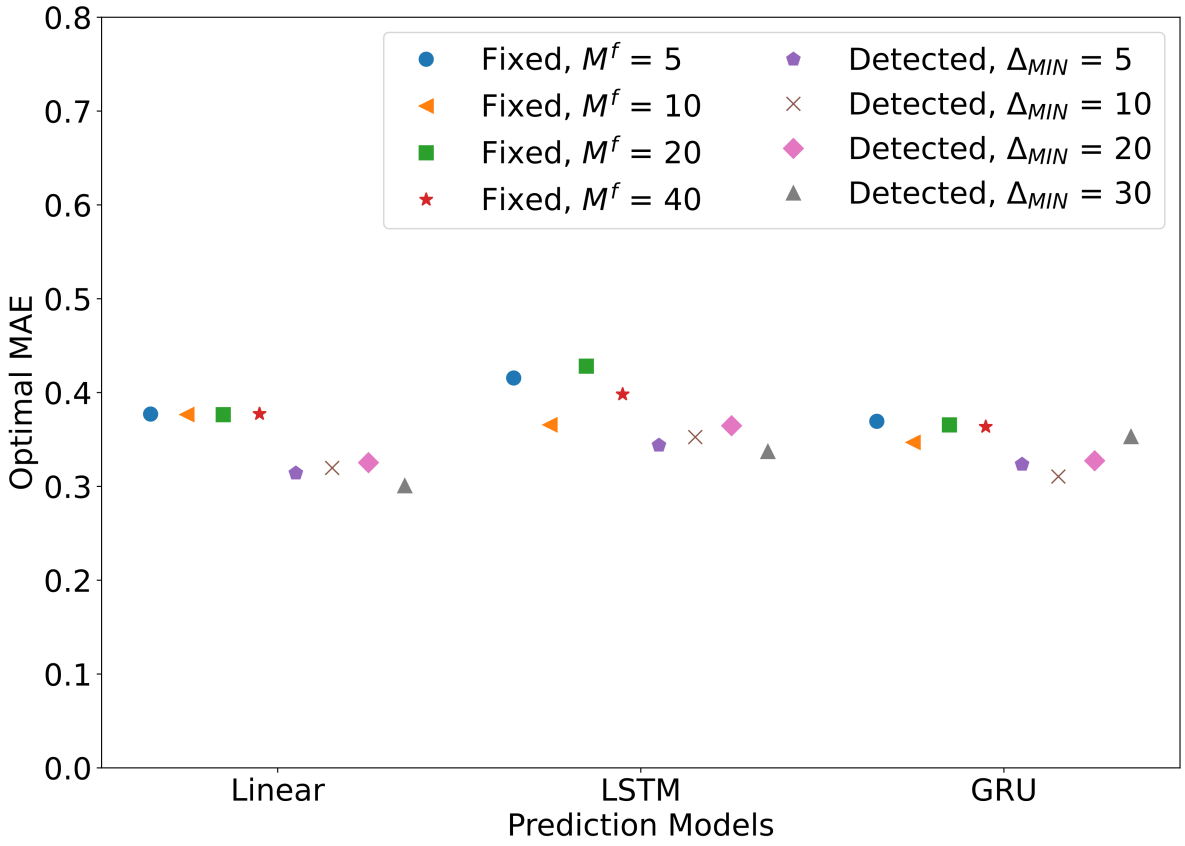


Figure 3.12: *A comparison chart of lowest prediction error obtainable using different fixed and detected windowing schemes for case data aggregated across an ecoregion. Four fixed windowing schemes with the number of windows, $M^f = 5$, 10, 20, and 40, along with four window detection schemes having minimum window lengths, $\Delta_{MIN} = 5$, 10, 20, and 30 weeks are compared for three models. The vertical axis denotes the optimum mean absolute error (MAE). The results were obtained for the ecoregion named Bahia Coastal Forest*[145].

We construct a risk map for the Bahia Coastal Forest ecoregion using the predicted cases and weights of the top-ranked PIC in Figure 3.13. The top 40 PICs are included in the

map, and the PICs that contribute more towards the weight (Γ) are shown as high-risk regions (e.g., red). The aggregated prediction was done using the optimal Linear model with $\Delta_{MIN} = 30$ and $N_{PIC} = 4$.

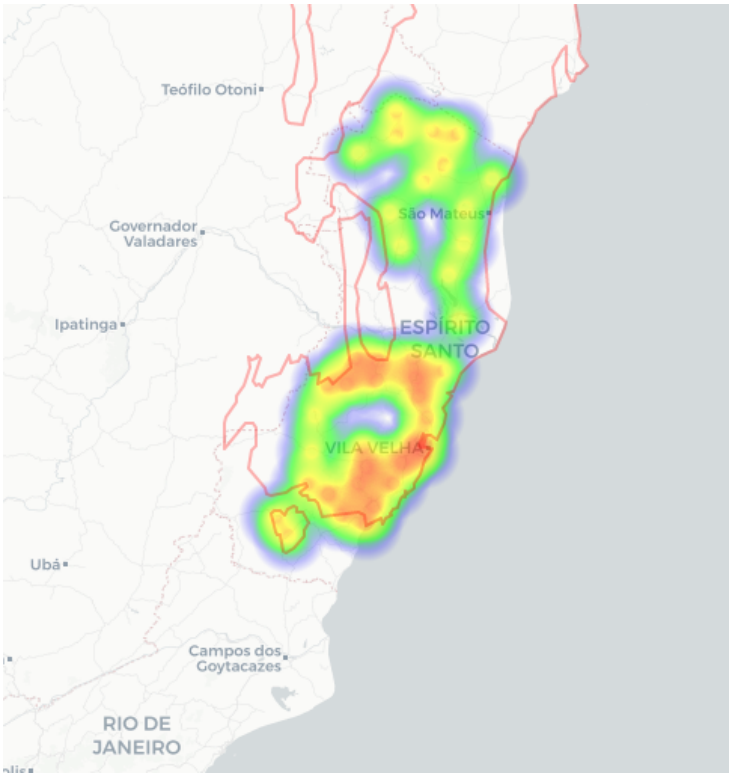


Figure 3.13: *A geospatial map showing the predicted risk of infection in an ecoregion of Brazil. A part of the ecoregion (Bahia Coastal Forests) shown here is marked with red borders. The risk of infection is shown as a heatmap with color shades ranging from blue (low risk) to red (high risk). The heatmap is constructed by combining the ranked PICs in the ecoregion, with higher-ranked PICs contributing more to the risk. The aggregated data was predicted using the Linear model with an optimum number of features ($N_{PIC} = 4$) selected with window detection method ($\Delta_{MIN} = 30$). This heatmap depicts the risk on the date of 09-June-2019, with 40 top-ranked PIC. Map generated using Folium[141] with OpenStreetMap[142]. Basemap tiles provided by CartoDB[143].*

3.5 Discussion

In this work, we develop a method to select relevant incidence data from peripheral locations as features to improve the prediction of dengue fever outbreaks. In order to rank features, we use windowed cross-correlation analysis on dengue case data. We propose two methods for

allocating correlation windows (position and size) over the time series to compute correlation weights. For a target location (IC), peripheral locations (PIC) are ranked based on a combination of correlation, distance, prevalence metrics. The predictive models benefit from the ranked feature sets, as these reach model and location-specific optimal performances with a relatively small subset of features. We tested three predictive models using dengue case data from Brazil, showing different levels of accuracy gains.

On average, the proposed feature enhancement methods improve prediction performance by 10.79% to 33.6% over the baseline feature set for the locations we tested, depending on the prediction model and the window allocation scheme. For the location with the highest total cases (2010-2019) in the Espírito Santo region of Brazil, we could get MAE values as low as 0.13 (normalized case data) using the GRU model with data from just 6 locations added to the feature set. In a test across multiple locations, both RNN models (LSTM and GRU) performed with comparable accuracy (average MAE ranging from 0.3435 to 0.3569) when using an optimal number of additional features. The Linear model also benefited (18.97% to 27.13% improvement over the baseline) from windowed correlation-based feature enhancements, although its performance never got close to recurrent models. When compared with the respective optimal number of features, the best performing recurrent models outperformed the Linear model by at least 41% in terms of prediction accuracy. The window detection methods showed performance comparable to fixed window allocation. This can be advantageous when working with extensive sets of data, as the detected windows only compare a subset of important time steps instead of the entire series. For municipality-level dengue case prediction, GRU was the best performing model, closely followed by LSTM. The performance gaps between these two models diminished after feature set optimization. When predicting aggregated data for the entire region using a subset of constituent locations, our methods reach optimal performance with the addition of only 2-5 locations (out of 77), depending on the model and window selection scheme. This is especially useful for situations where the lack of trainable data hinders forecasting. For example, dengue risk for a region can be predicted if a small subset of data from epidemically important region locations is available. This addresses the issue of incomplete data and eases the ‘curse of dimensionality‘

while improving training efficiency.

Future efforts in this area can focus on gaining further insights based on epidemiological, environmental, and economic characteristics of locations and combining them to improve feature ranks. While case data are indicators of the severity of an outbreak, these are not always adequate to explain future possibilities. Machine learning models cannot generalize beyond training data for every location with the same degree of accuracy. Complex interactions of dengue viral strains and changes in host immunity patterns may evolve outbreak characteristics over the years. Several random factors, including host travel patterns, natural disasters, and lifestyle changes because of other infectious outbreaks (e.g., COVID-19 pandemic), may affect dengue outbreaks. While the metrics used in our method capture such factors' long-term characteristics, these do not account for randomness. While keeping the feature sets reasonably small, our method can improve outbreak prediction. The proposed method can be generalized and used for projecting any infectious outbreak where temporal data at a reasonable spatial resolution are available.

Chapter 4

Generation of swine movement network and analysis of efficient mitigation strategies for African swine fever virus[1]

4.1 Background

Animal movement networks are important to model disease outbreaks and identify the pathways of disease spread. In the US, pig farm data, including herd sizes, geolocations, and movements between farms, are challenging to obtain due to the sensitive nature of data and the potential economic risk of making such information public. Epidemiologists and other researchers who need such data have to rely on models that can disaggregate available county or state-level data. One such example is the work of Burdett et al., who developed a simulation model to quantify pig population and generate geolocation of individual farms[146]. However, this model does not produce movement data. In another work by Valdes-Donoso et al., machine learning techniques were used to predict movement networks in the

[1]This chapter is a slightly modified version of a published article[2], Copyright 2019, PLOS ONE.

State of Minnesota[147]. Recent work uses a maximum information entropy approach to estimate movement probabilities among swine farms[148] and suggests that the 'small-world phenomenon' could make the US swine industry vulnerable to infectious disease outbreaks. Despite several efforts, pig level networks in the US swine industry are not readily available for simulating disease outbreaks. One way to overcome this issue is to design a network generator that can produce synthetic swine networks given some of the available movement network characteristics and census data.

There has been substantial work in the area of graph generation. The most basic random graph model is the Erdös - Rényi model[14] that can produce graphs with a certain edge probability between any pair of vertices. The vertex degrees of such random graphs follow the Poisson distribution[149]. There are several mechanisms to generate graphs with prescribed degree sequences. Milo et al. describes[150] two mechanisms: switching algorithm[151;152] and matching algorithm[149;153]. In the switching algorithm, graphs are generated based on a degree sequence, and the edges are shuffled without changing the degrees to introduce randomness. The matching algorithm is also called the configuration model[154] where stubs (open-ended handles) are assigned to vertices and later joined pairwise completely at random. Our limited movement data situation with a swine movement network presents us with a unique challenge where we have several different vertex types with their given average in/out degrees and their range of values[147]. We also have the probability of having a directed edge from one vertex type to another. Using these two sets of data, we design a network generator that uses a modified version of the configuration model, and the generalized random graph model[154]. Generated random graphs have been used for various purposes, including running outbreak simulations[1] and predicting the impacts of disease control[8]. Pig movement networks have been analyzed and found to be useful in predicting the risk of infectious disease outbreaks[155]. The effects of immunizations based on network centrality metrics have been explored before[156;157] for human diseases, and such studies can suggest efficient strategies for disease control. In this chapter, we use several proven network metrics to understand disease spreading phenomena in pig networks.

African swine fever (ASF) is a highly contagious infection that poses a threat for the pork

industry due to its high mortality and no effective vaccine or cure[158]. Several recent outbreaks in Romania, Bulgaria, and Belgium have already threatened European pork producers[159;160]. China, the largest pork-producing country, has an ongoing ASF outbreak and has reportedly culled 1,170,000 hogs as of 3rd October 2019[161]. They reported their first outbreak in early August 2018, and since then, there have been about 158 outbreaks in 32 provinces[161]. Several major Chinese pork producers have cut their profit forecasts; some expect as much as 80% reduction compared to 2017[162]. The Chinese officials have undertaken several methods to control the outbreaks that include culling of all pigs within 3km of the infected area, pig movement restrictions, surveillance around containment/protection zones, and destruction of pig products[163]. The analysis of Herrera-Ibata et al. finds that although the US has a low risk of ASF introduction overall, multiple states such as Iowa, Minnesota, and Wisconsin are the ones to be more vigilant about for an ASF introduction by the legal import of live pigs[164]. There have been several attempts to model ASF outbreaks. Barongo et al. used a stochastic compartmental model to investigate the effects of control measures on ASFV and found that early intervention can help manage the ASF epidemics[165]. The effects of residue from deceased animals were included in the work of Halasa et al. to simulate the spread of ASFV[166]. Using transmission experiments on the Georgia 2007/1 ASFV strain, Guinat et al. estimated pig-to-pig transmission parameters for both within pen and between pen infections, and they found the reproductive ratios to be 5.0 and 2.7 respectively[167]. On the other hand, Gulenkin et al. estimated the basic reproductive ratio for the outbreaks in the Russian Federation to be 8-11 within the infected farms, and 2-3 between farms[168]. Barongo et al. also estimated this ratio for Uganda outbreaks to be in the range of 1.58-3.24 depending on various estimation methods they used[169]. In another work, Guinat et al. inferred transmission parameters using pig mortality data[170]. Recent work by Hu et al. used Bayesian inference on previous transmission experiments[167] to account for unobserved infection times and latent periods[171]. Most of the ASFV research is focused on parameter estimates, while several others investigate virus importation risk in the US mainland. Despite the numerous studies, there is a lack of knowledge on how the swine industry in the US would be affected in case an ASFV outbreak starts in the US.

The contributions of this chapter are several: i) we propose a swine movement network generator, ii) we run ASFV epidemic simulations and compare how different farm operation types affect the outbreak dynamics, and iii) we analyze and compare the effectiveness of multiple centrality-based targeted control measures. In the *Results* section, we describe our generated farm-level network along with the outcomes of preliminary network analyses. We also explain the ASFV outbreak simulation results and compare different operation types as sources of infection. Finally, we investigate the impact of different disease control strategies. The *Materials and Methods* section contains detailed information on swine movement data, network generation, analysis methods, ASFV epidemic model, and its parameters. The pseudocodes for the algorithms are detailed in Appendix A.

4.2 Results

4.2.1 Movement network

The generated farm level movement network is shown in Figure 4.1. This directed network contains 84 farms from two Minnesota counties (Stevens and Rice). There are five different swine operations marked as Boar Stud (B), Farrow (F), Nursery (N), Grower (G), and Market (M) with 3, 22, 12, 39, and 8 sites, respectively. A visual inspection of Figure 4.1 suggests that movements of pigs start from farrow and nursery operations and end at the markets while a large number of grower farms lie in those paths. We also analyze the node centrality measures of the generated network, shown in Figure 4.2. As the network is generated based on degree centrality data (Table 4.2), we expect that results, shown in Figure 4.2(K_{in} and K_{out}) would resemble it. The market operations have significantly high in-degree centralities (median value of 9). In contrast, the nursery operations have high out-degree centralities (median value of 3) followed by farrow and grower operations (both with median values of 2). The farrow operations have high betweenness values (median of 8.9167) followed by grower operations (median of 4).

To understand how the connectivity in the farm network can be disrupted, we perform a

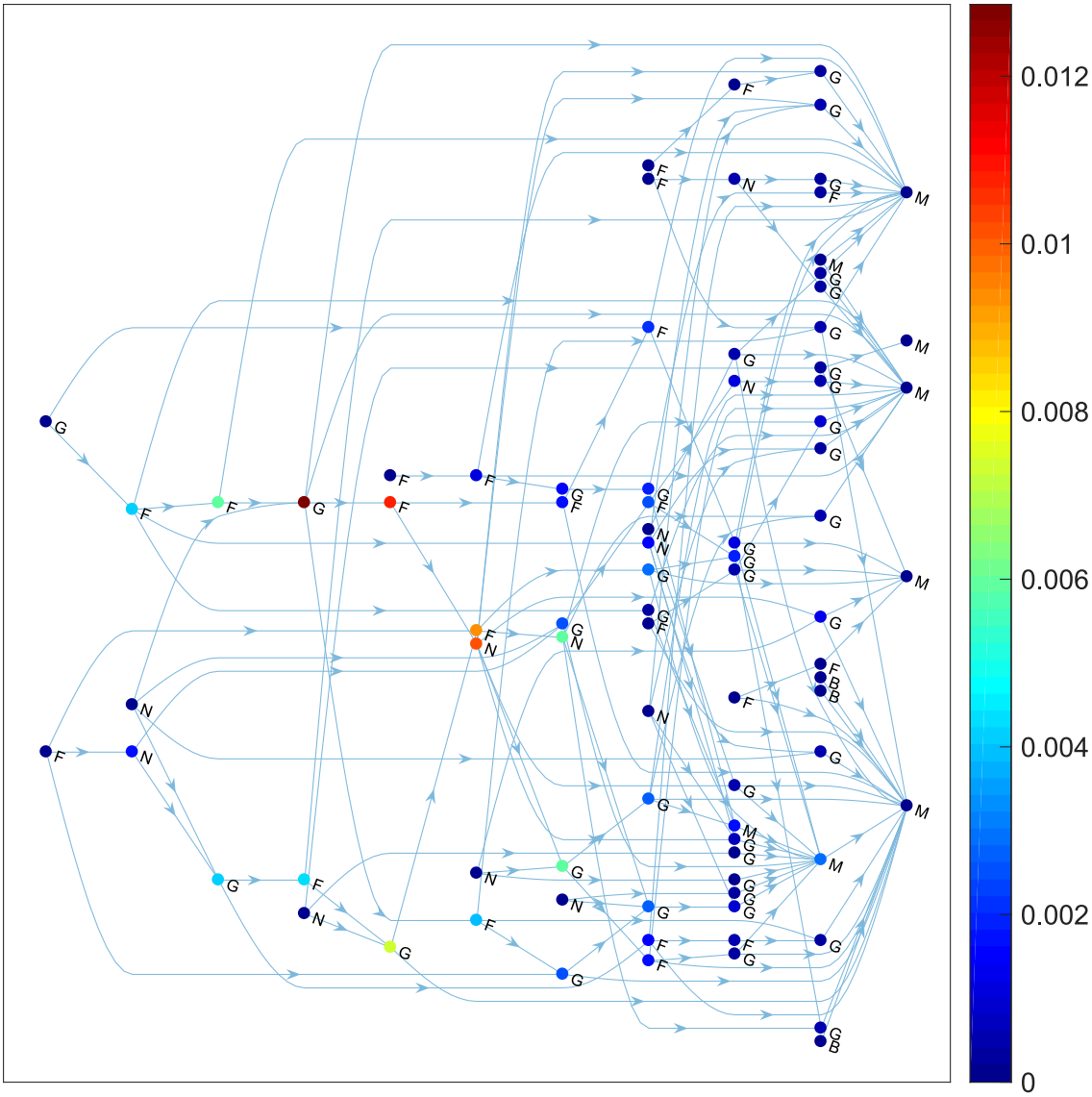


Figure 4.1: *A Generated farm level swine movement network. The solid colored circles (nodes) indicate swine operations, and the light blue arrows connecting them indicate pig shipments with directions. The colors of the circles indicate node betweenness centrality (values depicted by the color bar on the right). The swine operations (nodes) are labeled according to their types: Boar Stud (B), Farrow (F), Nursery (N), Grower/Finisher (G), and Market/Slaughterhouse (M).*

robustness analysis. Based on the node centrality measures of the network, we rank the nodes in decreasing order and create three lists (K_{in}, K_{out}, and BC). Going through those lists, we remove (isolate) nodes one by one from the network and compute the largest connected

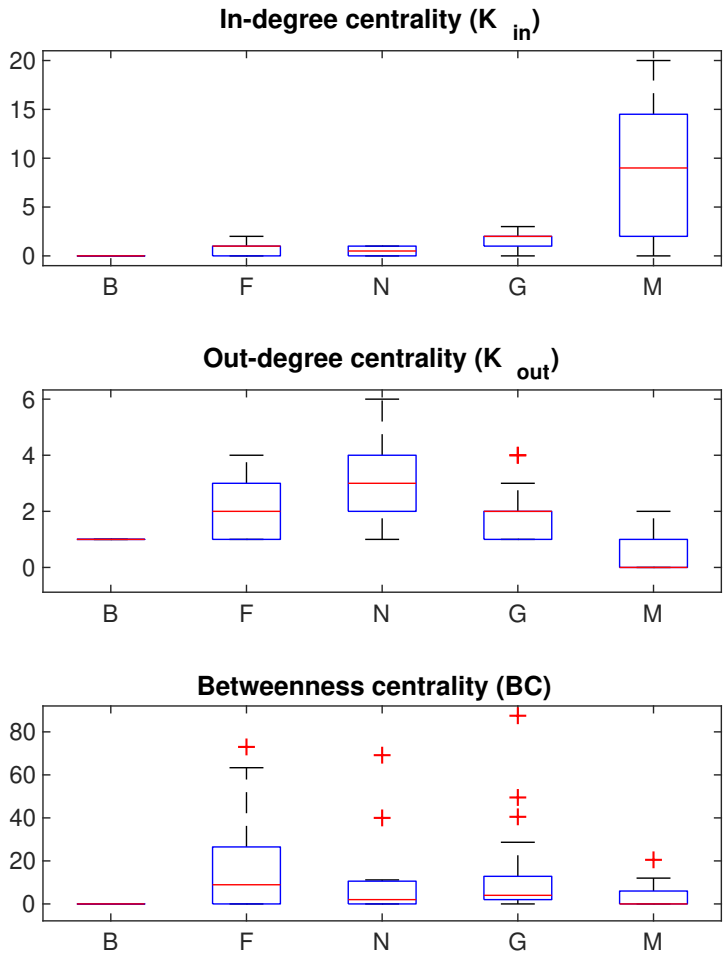


Figure 4.2: *Centrality measures of the generated network. The three sets of boxplots show three different centrality measures as marked (In-degree (K_{in}), Out-degree (K_{out}), and Betweenness (BC)). The five different pig operations are marked in the horizontal axes as Boar Stud (B), Farrow (F), Nursery (N), Grower/Finisher (G), and Market/Slaughterhouse (M). Each boxplot shows the range between 25^{th} and the 75^{th} percentiles (blue box) and the median (red line). The values outside 1.5 times the inter-quartile range are marked as outliers (+ signs).*

component in every step. The results are depicted in Figure 4.3, where the relative sizes of the largest components are plotted against three centrality-based node removal/isolation schemes. While all three schemes decrease the component sizes, the removal of high K_{in} nodes demonstrates a relatively better outcome in breaking the network. Approximately 94.1% of the farms in total can be isolated from the original network by isolating only

33.3% of the high in-degree farm nodes. For the other two schemes, isolation of 33.3% high centrality (BC and K_{out}) farms will isolate about 38.1% of the farms in total. The in-degree centrality-based isolation strategy shows a significant (about $\sim$ 2.5 times) improvement over other options.

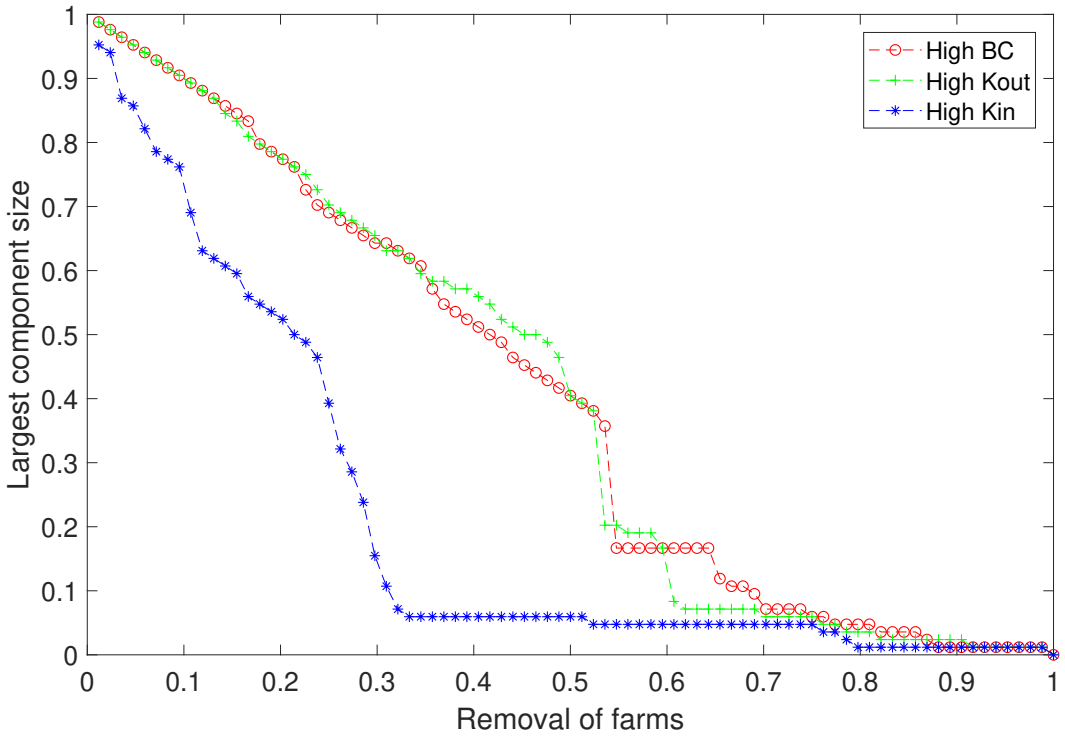


Figure 4.3: *Network robustness analysis by the gradual removal/isolation of farm nodes. The farm nodes are removed in decreasing order of different centrality measures, and the size of the largest weakly connected component (at the farm level) is plotted. Both of the axes are plotted as fractions of total farms in the network. For the removal of nodes, they are separately ranked with three independent centrality measures: high betweenness centrality (BC), high out-degree centrality (K_{out}), and high in-degree centrality (K_{in}).*

4.2.2 Outbreak dynamics

In a generated swine pig level network of the two Minnesota counties, we introduce an ASFV outbreak by choosing a pig farm uniformly at random as the seed farm. Within this selected farm, we infect at most 10 (if there are more than 10) pigs to introduce the pathogen and observe the progression of the disease spread. The averaged out results of 1000 independent

simulations are shown in Figure 4.4. We use the parameter values given in Table 4.5. For the infection rate, β, we use the median value given in Table 4.5 along with the values 25% above and below the median as indicated in the legends of the plots in Figure 4.4. We observe outbreaks lasting about 378 days for the median value of β, which infects about 1.84% [95% CI 1.65 2.03] of the pork population. For a network of 249,150 pigs, this roughly translates to about 4,584 [95% CI 4,111 5,047] pigs dying from the outbreak. A 25% increase in β would lengthen the outbreak duration by about 33% and affect twice as many pigs. A 25% reduction in β shortens the outbreaks by 32% and reduces the outbreak size by 59.8%. For β values around the median and above, outbreaks reach their peaks within 95-100 days. For β values below the median, outbreaks do not surpass the initial fraction of infected pigs.

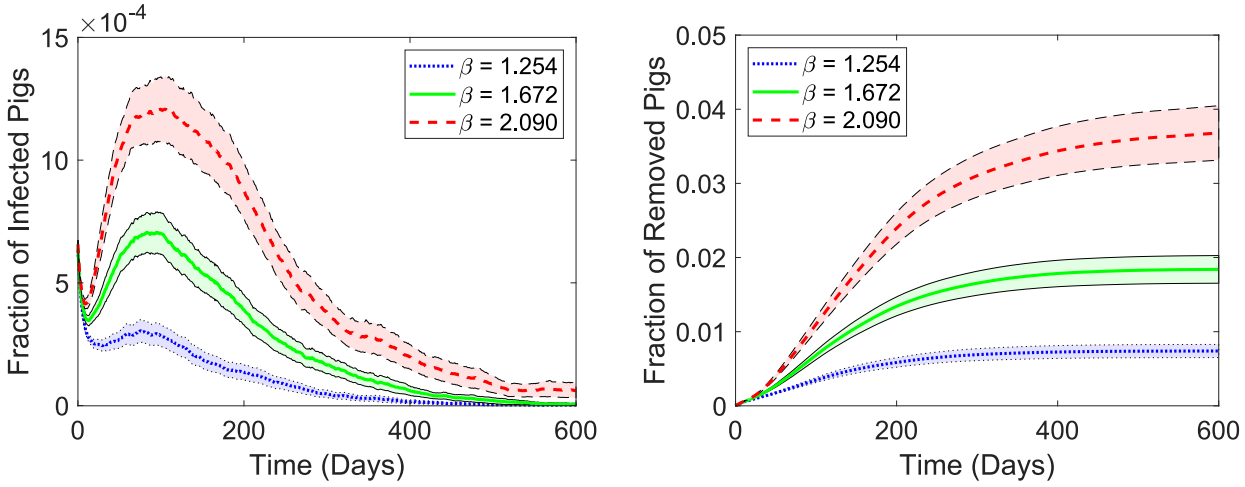


Figure 4.4: *Simulated time series outbreak dynamics in the generated swine network. The results shown above are the averages of 1000 independent simulations. To start each outbreak, a herd/farm is selected uniformly at random, where we infect up to 10 randomly chosen pigs. The simulations were run for three different β values (1.672, 1.254, and 2.090), which are shown using different line styles and colors as indicated by the legends. The shaded regions in the plots show 95% confidence intervals. The left plot shows the fraction of infected pigs, and the right plot shows the fraction of removed (dead) pigs over time for the generated pig network.*

For the results of Figure 4.4, we infected about 10 pigs in a farm that was chosen uniformly at random from all the farms. As there are five different pig operation types in our network, we would like to evaluate how each type affects the outbreaks. We run independent sets of simulations where we target a specific operation type (boar stud, farrow, nursery, grower,

and market) in each set. We select an operation of that particular type and use it to seed the infection. It is important to note, the number of pig operations in each type/category is different. The pig population also varies among operations. In our generated network, we have approximately 3.82%, 28.72%, 11.73%, 44.86%, and 10.87% pigs in Boar Stud, Farrow, Nursery, Grower, and Market operations, respectively. The outcomes are shown in Figure 4.5. Here, we define the term 'Epidemic Attack Rate' as,

$$Epidemic\ Attack\ Rate = \frac{Number\ of\ pigs\ infected\ during\ the\ outbreak}{Total\ number\ of\ pigs} \tag{4.1}$$

We find that markets are most capable among the five types in spreading infections, while grower and farrow farm types are the second and third most important to consider. Although grower farms have 4.13 times the population of the market sites, the market sites cause 1.98 times bigger outbreaks (0.0406 [95% CI 0.0398 0.0414]) compared to grower sites (0.0205 [95% CI 0.0199 0.0212]). Despite that, the duration of the outbreaks caused by the farrow, grower, and market sites are quite comparable (387 [95% CI 374 401], 399 [95% CI 392 409], and 423 [95% CI 416 431] days respectively). The large populations in the grower and farrow farms have contributed to their large outbreaks. On the other hand, market sites are potent infection spreaders due to their high connectivity (high in-degree centrality) with remaining farm types.

4.2.3 Control measures

Due to the lack of a cure for the African swine fever virus, movement restriction remains a key control method for policymakers. For this experiment, we use three different network centrality measures (in-degree centrality, K_{in}, out-degree centrality, K_{out}, and betweenness centrality, BC) for the farm nodes and sort the farms in a descending order based on these measures. Next, we gradually place movement restrictions on an increasing number of farms selected from the sorted lists and run outbreak simulations. The attack rates and the outbreak lengths are compared in Figure 4.6 for three network centrality measures. Placing movement restrictions based on in-degrees (K_{in}) demonstrate the best performance in dis-

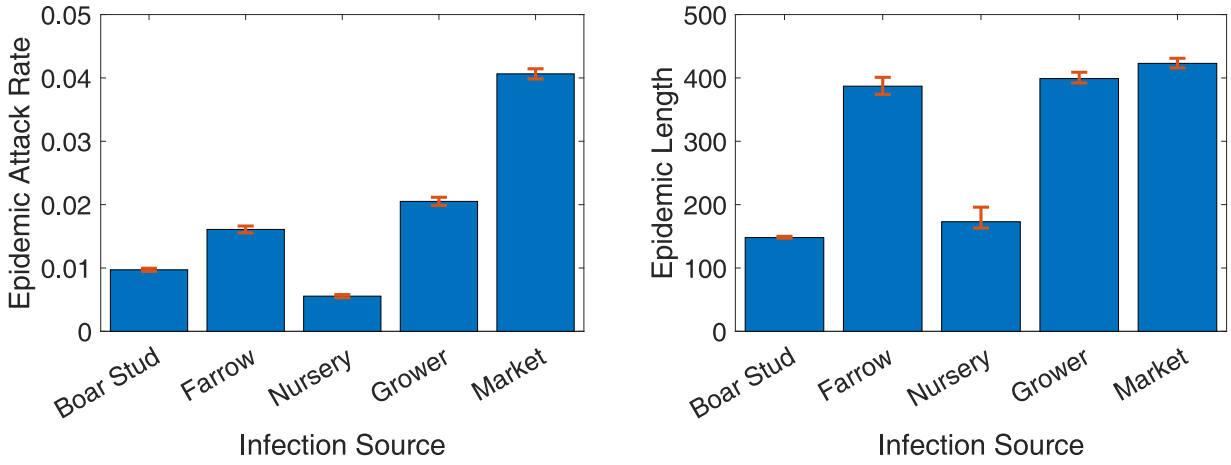


Figure 4.5: *Simulated outbreak analysis based on the source of infection. The results shown above are the averages of 10,000 independent simulations. The 95% confidence intervals are shown in red error bars. To start each outbreak, a pig operation was chosen from a given type (either Boar Stud, Farrow, Nursery, Grower, or Market), and up to 10 pigs from that operation were infected. The left plot shows the epidemic attack rates as defined in Equation 4.1 and the right plot shows the duration of outbreaks.*

ease control while restrictions based on betweenness centralities (BC) perform the worst. Isolation of top 5 farms based on K_{in} shows about 63.04% [95% CI 61.96 64.13] reduction in the outbreak size (attack rate) and 51.59% [95% CI 50.26 52.91] reduction in outbreak duration compared to the situation without any control measure (Figure 4.4). For the K_{out} and BC based isolation schemes, we observe 19.6% [95% CI 16.85 21.74] and 4.9% [95% CI 1.63 7.61] reductions respectively in outbreak sizes with 8.5% [95% CI 6.61 11.64] and 6.4% [95% CI 4.5 8.47] reductions respectively in outbreak durations when we isolate 5 farms.

As there is no effective vaccine for ASF, we model hypothetical vaccines with 80% efficacy. This efficacy value has been mentioned in other cases as a nominal requirement to make a marketable vaccine[172]. For our model, it means that 80 out of 100 vaccinated pigs will be fully immune to the invading pathogen. We use the same set of centrality-based sorting strategies to select farms for vaccinations (in-degree K_{in}, out-degree K_{out}, and betweenness centrality, BC measures). The results are shown in Figure 4.7. Once again, immunizing farms based on high in-degree (K_{in}) is found to be the most effective strategy, while immunization based on high betweenness centrality (BC) is found to be the least effective in disease control.

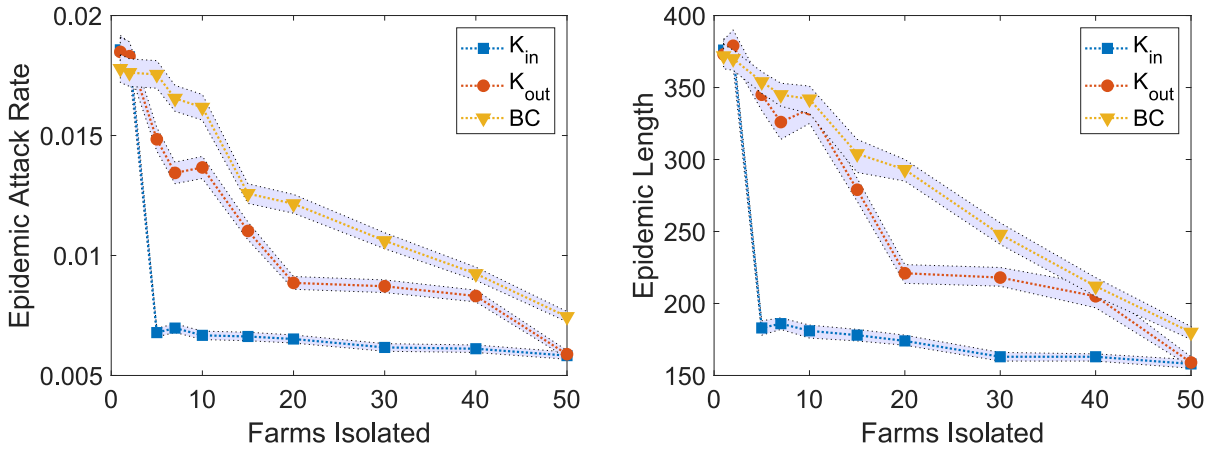


Figure 4.6: *Comparison of different targeted isolation schemes based on farm node centrality measures. Three different movement restriction strategies (high in-degree, K_{in}, high out-degree, K_{out}, and high betweenness, BC) are compared. For each strategy, a different number of farms are isolated from a centrality-based sorted descending list. The left plot shows the epidemic attack rates, and the right plot shows the epidemic lengths. The data points are mean values computed from 10,000 stochastic simulations, and the shaded regions show 95% confidence intervals.*

Vaccination of top 5 farms based on K_{in} shows about 59.78% [95% CI 58.70 60.87] reduction in the outbreak size (attack rate) and 44.18% [95% CI 42.86 45.77] reduction in outbreak duration compared to the situation without any control measure (Figure 4.4). For the K_{out} and BC based immunization schemes, we observe 17.93% [95% CI 15.22 20.65] and 3.8% [95% CI 0.54 7.07] reductions respectively in outbreak size with 5.56% [95% CI 2.91 7.67] and 5.03% [95% CI 2.12 7.94] reductions respectively in outbreak duration when we vaccinate 5 farms. The comparative results of the vaccination strategies resemble the results found in the previous experiment for movement restriction measures.

4.3 Discussion

This study proposed a method to generate movement networks from available data on the US swine industry, utilizing movement network characteristics available for two counties in Minnesota. Using the generated farm-level movement network, we have analyzed multiple centrality properties and performed a robustness analysis to obtain a better insight into the

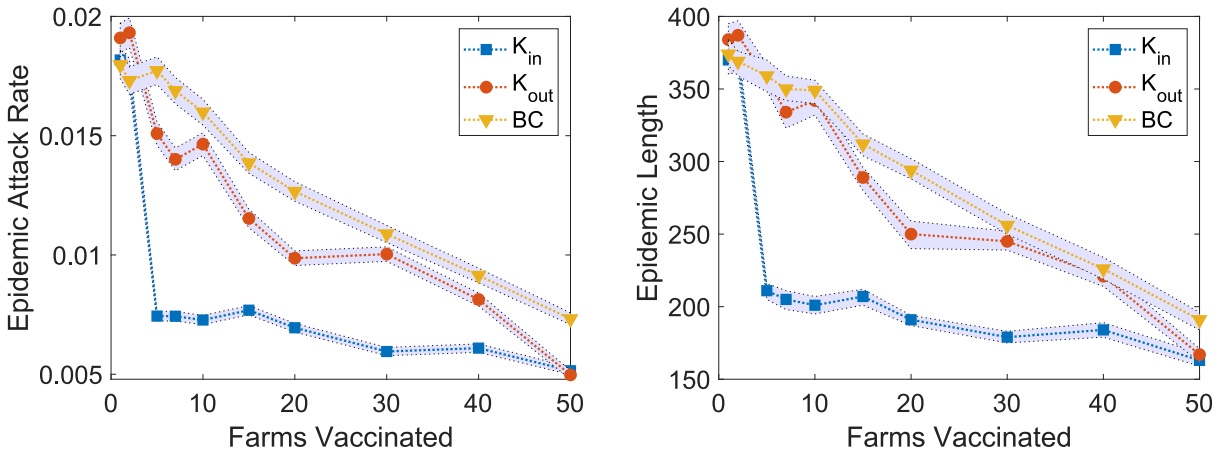


Figure 4.7: *Comparison of different targeted vaccination schemes based on farm node centrality measures. Three different vaccination strategies (high in degree, K_{in}, high out-degree, K_{out}, and high betweenness, BC) are compared. For each strategy, a different number of farms are immunized from a centrality-based sorted descending list. The hypothetical vaccines are 80% effective. The left plot shows the epidemic attack rates, and the right plot shows the epidemic lengths. The data points are mean values computed from 10,000 stochastic simulations, and the shaded regions show 95% confidence intervals.*

network structure. Using the generated pig-level contact network, we formulated a stochastic $SEIR$ model for the transmission of African swine fever. We ran outbreak simulations and examined time-series data with different pig operation types as sources of infection and compared the outcomes. Finally, we analyzed and compared the outcomes of centrality-based targeted isolation and vaccination methods.

The outbreak simulations show that if ASFV is introduced in a random herd and is allowed to spread unchecked, it may affect approximately 1.84% of the total swine population, with a high probability for the two counties in our consideration. Among the five different farm types, infecting the pig population in the market operations causes the most significant outbreaks. The high connectivity of the markets with other farm types and the both-way transmission caused by fomites (e.g., transport vehicles) are reasons behind such a high impact of the markets. The large populations in grower and farrow farm types also make them significant in spreading ASFV infections. Control measures can target these farm types in the event of such outbreaks. Our preliminary farm network analysis found that the nursery operations have high out-degrees while the market operations have high in-degrees. We also

find that grower operations have high betweenness centrality values. A network robustness analysis reveals that isolating high in-degree farms disrupt the connectivity in the network the most compared to using other centrality measures.

When we examine the impact of centrality-based targeted control measures, the outcomes reinforce our results from the preliminary analysis. We have examined two different control measures with outbreak simulations: movement restriction and hypothetical vaccine. In both cases, we find that controlling farms with high in-degree proves to be beneficial in containing the disease spread. Implementing control in high out-degree farms proves to be slightly better than doing so in high betweenness farms, while both are inferior compared to high in-degree based targeted control. In a separate analysis (Figure 4.5), market operations have proven to be highly potent sources of infections, causing more significant outbreaks than other farm types. As the market operations have very high in-degree values, our results consistently suggest that these sites should be prioritized in the case of ASFV outbreaks.

Limited public data availability on swine movement in the US compels us to rely on probabilistic network-generation methods to close analytical gaps. Available data on Stevens and Rice counties of Minnesota aided the construction of the movement network. However, these data may be inadequate for the extrapolation of more extensive swine-movement networks. Despite that, our generated network has degree distributions that agree with the given data and the real-world characteristics of the swine production industry. If additional data for movement networks in other locations become available, our network generation algorithms can be used with little or no modifications, depending on the data. We also made a simplifying assumption of having one operation type at a single site, while in practice, there can be multiple operation types. In addition to that, individual-based simulation models are limited due to computational complexities caused by a large population. Metapopulation models can be a viable solution when considering state-level networks. The network generation techniques can be improved further if more data on swine production operations is made available. Distributed databases could be used to improve traceability and data sharing for the agriculture production supply chain. Further efforts could be made in performing surveys, raising awareness, and motivating the livestock industry to participate in

data exchange to support research solutions that can benefit the industry operations.

4.4 Data and models

4.4.1 US swine data

We generate the swine movement network utilizing some of the network characteristics (mixing matrix, in-degree, and out-degree centralities) reported in the Valdes-Donoso[147] paper. The mixing matrix is given in Table 4.1 and the centralities are shown in Table 4.2. We define several pig operation types that include farms and markets. Using the operation type distribution described in the same work[147], we classify 5 different pig operations (Boar Stud, Farrow, Nursery, Grower, and Market) as shown in Table 4.3. The operation types are defined below,

- **Boar Stud**. These farms are used to keep male boars for breeding.
- **Farrow**. Sows are moved to these farrowing farms to give birth (farrow). Piglets stay here up to 3 weeks.
- **Nursery**. Piglets are moved to nursery after weaning where they could stay up to 8 weeks.
- **Grower**. Pigs are moved from nursery to grower/finisher farms where they will gain market weight at about six months of age.
- **Market**. The market type includes buying stations and/or slaughter plants.

Obtaining data from the United States Department of Agriculture - National Agricultural Statistics Service (USDA-NASS)[173], we find that two counties (Rice & Stevens) of Minnesota have 84 farms and 249,150 pigs in total. We take the 84 farms and, as the operation types are unknown, assign types randomly based on the distribution shown in Table 4.3.

Availability of operation type distribution data is incomplete as well; there are several suppressed data fields. We allot pigs in those unknown fields randomly and make sure that

	B	**F**	**N**	**G**	**M**
B	0.00	0.00	0.00	0.00	0.01
F	0.00	0.03	0.04	0.09	0.10
N	0.00	0.00	0.00	0.13	0.00
G	0.01	0.10	0.00	0.07	0.40
M	0.00	0.00	0.00	0.00	0.02

Table 4.1: *Mixing matrix (probability of movement from row type to column type) for swine movement network[147]. The pig operation types are abbreviated as B (Boar Stud), F (Farrow), N (Nursery), G (Grower), and M (Market).*

		B	F	N	G	M
In-degree	Average	0.67	0.92	0.77	1.05	11.73
	SE	0.67	0.14	0.1	0.07	3.59
	Max	2	5	2	5	57
Out-degree	Average	1.00	2.08	3.07	1.74	0.46
	SE	0	0.26	0.62	0.15	0.18
	Max	1	8	12	12	3

Table 4.2: *Swine movement network degree centrality data[147].*

Boar Stud(B)	Farrow(F)	Nursery(N)	Grower(G)	Market(M)
1.27%	27%	12.66%	51.9%	7.17%

Table 4.3: *Pig operation type distribution.*

the aggregate statistics are maintained. The adjusted combined statistics for Stevens and the Rice counties are provided in Table 4.4.

Farm Size	No. of Farms	No. of Pigs
1 to 24	17	204
25 to 49	0	0
50 to 99	0	0
100 to 199	2	300
200 to 499	3	700
500 to 999	11	7,904
1,000+	51	240,042
Total	84	249,150

Table 4.4: *Distribution of pigs in Stevens and Rice counties of Minnesota.*

While the USDA-NASS data provide the total number of farms and pigs in a size class, it is impossible to infer the number of pigs at individual farms. Hence, we use a random allocation mechanism to assign the number of pigs for each farm while maintaining the

aggregate statistics of Table 4.4. Once we generate the network edges, we assign a weight to them to indicate the amount/rate of movement via that edge. According to the work of Spencer R. Wayne[174], the Rice and the Stevens counties experience mean shipment of 21 and 15 per year and median shipment of 10 and 7 per year, respectively. Our combined network is estimated to have a mean shipment of 17.38 per year and a median shipment of 8.5 per year based on those values. We use lognormal distribution and assign randomly generated shipment rate values to network links.

4.4.2 Network terminology

We use several network structure and analysis related terminologies throughout this chapter. These terminologies are described below,

- **Network/Graph**. A network (also called a graph) is a structure consisting of nodes (also called vertices) and links (also called edges). A link connects two vertices, and it can be either directed or undirected.

- **Stub**. A stub is half a link. It's a link with a node on one end and an empty handle on the other end. Empty handles of two stubs can be joined together to form the link and create a connection between two nodes.

- **Path, Shortest Path**. A path is a sequence of links that joins a sequence of distinct vertices. A shortest path is the minimum length path between two nodes in a network.

- **Connected Component**. A connected component (also referred to as a component) is a subset of nodes with a path between every pair of nodes in that subset. Two distinct components aren't connected by any path. If all nodes in a component are connected via bi-directional paths, then the component is strongly connected. Otherwise, it is called weakly connected (path in one direction). In this chapter, we consider weakly connected components as transmission can happen in the reverse direction of the animal movement via fomites (e.g., transport vehicles).

We use several centrality measures to determine the importance of the nodes. The centrality measure can quantitatively characterize how important a node is in the network.

- **Degree Centrality**. The degree (K) of a node is the number of links associated with that node. In the case of directed networks, we define in-degree (K_{in}) as the number of links going into the node and out-degree (K_{out}) as the number of links coming out of the node.

- **Betweenness Centrality**. There is a shortest path for every pair of nodes in a connected component. The betweenness centrality (BC) of a node is the total number of shortest paths that pass through that node (not counting the paths starting from or ending at that node).

4.4.3 Network generation

The swine network is synthesized using the available swine farm and movement-related data described in the previous section. The network generation process is completed in several stages:

1. Assign each farm node a single operation type randomly based on the farm type distribution given in Table 4.3.

2. Assign directed in and out-degree values or handles (stubs) to each farm node randomly based on the degree distribution given in Table 4.2.

3. Connect out-handle (stub) of a farm node to in-handle (stub) of another farm node randomly, based on the mixing matrix given in Table 4.1.

4. Assign shipment rate values to all the directed links from a lognormal distribution with the obtained mean and the median shipment rate values.

5. Assign each farm a certain number of pigs randomly, based on the distribution given in Table 4.4.

6. Generate the within-farm undirected contact links among the pigs based on the Erdös - Rényi process with 50% probability.

7. Convert the shipment rates of farm links into probabilities and generate between-farm undirected contact links for the pigs based on those rates.

We generate a farm-level movement network at step 4 and a pig-level contact network at step 7. It is necessary to mention that working with a graph that has 249,149 nodes is computationally intractable due to a large number of within-farm links among the pigs. Hence, we scale down the pig population by a constant factor of 20, which makes the network small enough to be computationally feasible while retaining sufficient pig nodes to maintain connectivity properties of the farm-level network. As a consequence, most of our ASF model results are qualitative investigations of outbreak behavior.

4.4.4 ASFV epidemic model

Our network-based epidemic model is shown in Figure 4.8. Using the farm-level movement network, we generate a pig-level movement network. In this network, each node is an individual pig, and the links connecting a node to other nodes indicate interactions with other pigs (nodes). A pig has many more links to other pigs within the same farm than pigs at other farms. The links to other farms are generated based on the movement network. In Figure 4.8, a host node (pig) is marked using a solid circle, and the links to other nodes are marked by the solid lines. A host (pig) can get exposed from any of its infected neighbors at the rate of β, which is defined as the infection rate. For modeling African swine fever infection dynamics, we divide the pig population into four groups: Susceptible (S), Exposed (E), Infected (I), and Removed/Dead (R). The healthy pigs which are free from ASF infection are classified as Susceptibles. If such a healthy pig comes into contact with infected pigs containing the virus, it may get infected at the rate $\beta Y_i(t)$, where $Y_i(t)$ is the number of infected neighbors of node i at the time t. If the pathogen transmission occurs, a healthy pig enters into the Exposed group, where it stays for the incubation period. On average, this period is denoted by $1/\sigma$. Once it shows symptoms, it moves into the Infected group. It stays there

for an average time of $1/\gamma$ before it is removed. As for ASF, the mortality is assumed to be 100%, and no pig recovers. Hence, all infected pigs die at the end of the infected period. However, in multiple cases for our simulations, we will hypothetically vaccinate pigs. Based on the vaccine efficacy, alive pigs may move to the removed class too.

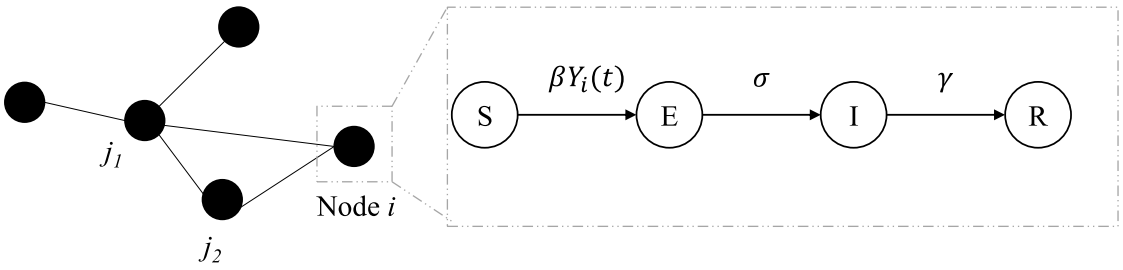


Figure 4.8: *The network-based SEIR epidemic model for African swine fever virus. The solid black circles indicate host nodes (individual pigs), and the solid lines connecting them indicate contacts (direct or fomites) that can act as infection pathways of ASFV. Each node can be in any of the four states, Susceptible (S), Exposed (E), Infected (I), or Recovered (R). The parameters indicate the rates at which a host can move from one state to another (See Table 4.5) adjacent to corresponding arrows. Here, $Y_i(t)$ is the number of infected contacts of node i at time t.*

The model parameters are shown in Table 4.5. The last column in this table mentions the different sources from where we obtained the parameter values. For β, we used estimated data from[170] where median transmission rate values were computed for 9 herds. These values are listed in Table 4.6. We take the weighted median from this set of data and use that β value in our simulations. We use the well-developed GEMFsim[75] tool to run our simulations.

Symbol	Definition	Range	Value	Reference
β	Transmission Rate	0.7 - 2.2	1.6719	[170] [171]
$1/\sigma$	Latent Period	-	7.78	[170]
$1/\gamma$	Infectious Period	-	8.3	[170]

Table 4.5: *ASFV epidemic model parameters.*

Herd Size	1614	1949	1753	1833	1320	600	600	600	2145
β	2	1	2.2	0.7	1.6	2.1	1.6	2.2	0.8

Table 4.6: *Transmission rate estimated for 9 pig herds by Guinat et al.*[170]

Chapter 5

A permissioned distributed ledger for the US beef cattle supply chain[1]

5.1 Background

The cattle industry in the US does not provide the appropriate traceability for the rapid identification of likely infected cattle during infectious outbreaks. The United States Department of Agriculture (USDA) mandates veterinary inspections for inter-state movements[175]. However, intra-state movement data are kept private by the farm owners, making it challenging to trace animal movements. There are several projects which are addressing traceability. CattleTrace was established in 2018 to create an infrastructure for an animal disease traceability system in Kansas[176]. Such projects rely on the mutual trust of the participants for keeping data secure, which can impede widespread adoption. In this chapter, we propose a technological solution to security, privacy, and control of private and shared data using a permissioned blockchain. Blockchain technology is designed to be immutable, transparent, and decentralized, enabling secure communications among parties without the need for intermediate or central authorities. Cryptocurrencies[37] were among the first applications of blockchain, which expanded into multiple fields due to some valuable features of the

[1]This chapter is a slightly modified version of a published article[3], Copyright 2020, IEEE Access.

blockchain architecture. The possibilities are endless with smart contracts that can execute Turing-complete instruction sets[41;42].

5.2 Preliminaries

5.2.1 The US cattle farm system

The beef supply chain consists of several components. One prior work[177] divides the beef cattle operations into four stages: ranch, stocker, feedlot, and packer. We add two more components, distributor and retailer, to complete the chain. The diagram is shown in Figure 5.1. The cattle stay at different farm types (Ranch, Stocker, and Feedlot) based on their ages and weights. After starting its life in a ranch, a steer or heifer moves into a Stocker when it is 6 to 9 months old and weighs about 400 to 700 pounds. Cattle may move into Feeders for further weight gain. Feeder cattle are aged between 12 to 24 months and weigh about 800 to 1,000 pounds. The time that cattle spend in a feeder is often called the finishing phase. The animals are slaughtered at the Packers, where meats are prepared and packed for distribution. We also add a Distributor stage before a Retailer, as this can be the case in many beef cattle operations. The consumer of the products lies at the end of the supply chain.

5.2.2 Prior work and motivation

There have been multiple blockchain-based applications for supply chains. Approximately 1.1 billion USD have been invested in blockchain technology[178]. IBM and the Danish shipping company Maersk created a platform named TradeLens and experimented with blockchain to track shipping containers around the world[179]. As an application in the fishing industry, Provenance launched a project in Indonesia to track tuna using blockchain[180]. Lu et al. explored the adaptability of blockchains for product traceability in supply chains and discussed the technology's strengths and limitations[181]. Dudder et al. described a model to track timber by using a tamper-proof system based on blockchains[182]. Saberi et al. discusses

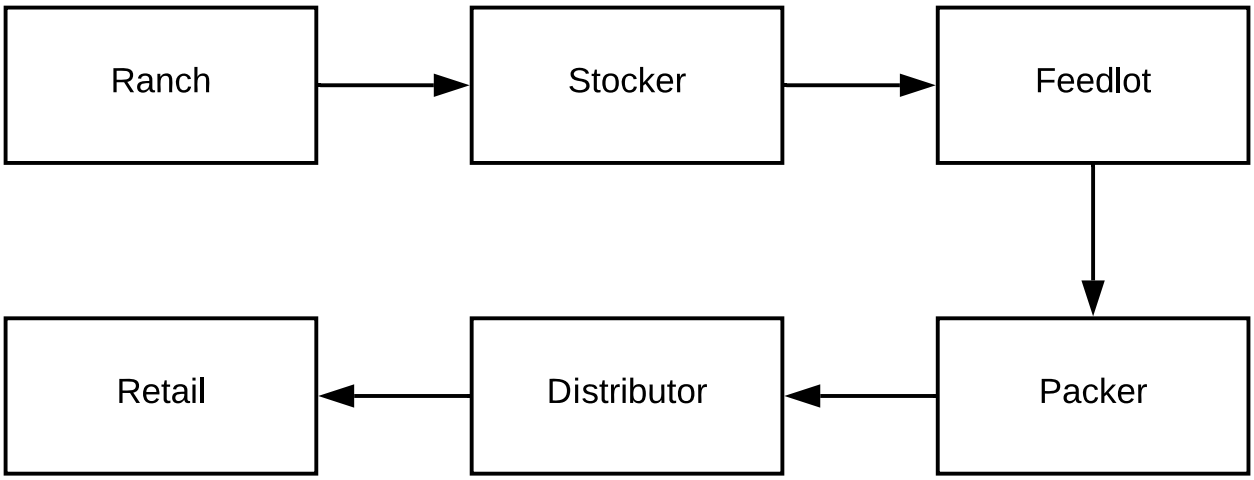


Figure 5.1: *A block diagram of the US beef supply chain. The first three (ranch, stocker, and feedlot) segments deal with live animal production. The packer is the manufacturing plant. Once the beef is processed, it reaches the consumers via distributors and retailers.*

different barriers of blockchain technology adoption and identifies the lack of collaboration as one of the inter-organization barriers[183]. A work of Casado-Vara discusses how blockchains can improve traditional linear supply chains into circular economies[184]. Another work developed a provenance knowledge framework and addressed how it can enhance assurances of product quality in the supply chain[185]. Leng et al. propose a dual-chain architecture for use in agricultural supply chains[186]. For traditional supply chains, adoption of this technology would enable traceability and provenance[187–190], prevent counterfeits and defects[191], reduce regulatory costs and complexities[192], and even take advantage of smart transportation systems[193]. After a recent *E. coli* outbreak in Romaine lettuce, the difficulty in tracing back to the source of infection prompted Walmart Inc. to work with IBM to use blockchains storing data of all leafy green vegetables[194]. The technology was also applied in cattle industries. One such example is the Beefchain system in Wyoming that promises the ranch owners with opportunities to acquire the differentiating profits based on the quality of beef (e.g., premium grass-fed)[195].

Different applications have gained various advantages from this technology. Wang et al.[43] proposed a decentralized system where smart contracts enable fine-grained access control. Blockchains also made possible an anonymous reputation system for vehicular ad hoc net-

works (VANET)[196]. It has been suggested to keep raw data off-chain to deal with scalability issues[181]. The security and immutability of blockchain have found popularity in electronic health record (EHR) systems. Recent work uses an interplanetary file system (IPFS) to store data off-chain[44]. In supply chains, the benefits can be numerous, including proof of product delivery with automatic incentives[197], provenance tracking[198], and traceability[199]. While different applications focus on various benefits of the blockchain, none of them properly fits the US animal farming industry requirements. We need a system that can be quickly adapted to existing infrastructure while maintaining user anonymity, animal/product traceability, and farm data ownership at the same time.

5.2.3 Smart contracts and blockchain

Blockchains use a linked-list type of structure where a block is linked to its previous block via cryptographic hashes. The hashes are computed from the contents of the blocks. Hence, any modification in the contents of the earlier blocks would require all the subsequent blocks to be updated. This makes it difficult for a perpetrator to change any past data. A block is confirmed in the chain via a process called consensus, where the participants in the chain (also called miners) agree on the data contained by the block. There are several configurations for blockchains. Bitcoin is a permissionless and public network where anyone can join and participate. There are frameworks such as Hyperledger or Ethereum that can be deployed in permissioned networks. These can be run in corporate or private setups where only the permitted users can join and participate. There are multiple consensus protocols, the most popular being Proof of Work (PoW). The miner needs to successfully solve a cryptographic problem to mine a block in PoW. This prevents parties with malicious intents who want to corrupt the data, as they will run out of computational resources while going against honest miners in a practical world. However, PoW is computationally expensive and wasteful. In permissioned systems, several other consensus protocols can be used, such as Proof of Stake (PoS), Proof of Authority (PoA), etc. These protocols use significantly less computational resources compared to PoW. In PoA based systems, blocks are signed by pre-approved

accounts called validators. A block in the chain contains a list of transactions that are analogous to bank checks in the context of cryptocurrencies. Many blockchain frameworks now support smart contracts, which are pieces of executable code that run when a block is confirmed and modify the state of the system. Smart contracts can contain codes that can enforce business policies, privacy practices, and access control. They can also store data. The blockchain framework ensures that everyone has the same version of code and data, which creates trust.

5.3 The proposed system

We have designed a system that ensures data immutability while preserving data ownership. The system also supports animal traceability, user anonymity, and data aggregation. Our proposed system model is shown in Figure 5.2. More details about the system can be found in Appendices B and C. Our framework is designed on the Ethereum platform. The framework supports smart contract storage and execution, which is a key to our work. The active nodes in the network communicate as peer-to-peer (P2P), and there is no centralized server like in the client-server model. External applications can connect to their respective nearest P2P nodes to retrieve information, manage system operations, or generate transactions. All transactions are stored in each of the active nodes of the network. In addition to the transactions, smart contracts are also stored redundantly in each active node. Cryptographic hashes are computed for both transactions and smart contracts. The hashes are stored inside the chain data structure in multiple blocks with timestamps. The blocks are linked in a list structure using cryptographic hashes of their contents. Hence, a modification in the data or the code would require the system to recompute the hashes. As the blockchain data structure is redundantly stored in all the participating nodes, all nodes must agree on the modification (e.g., come to a consensus). We define five major smart contracts for managing the farm animal tracking system: 1) Profile Manager, 2) Farm Manager, 3) Transaction Manager, 4) Trace Manager, and 5) Data Aggregator as shown in Figure 5.2. The dots in the P2P Network are Ethereum nodes (clients) (Either geth[200], or parity[201] with Proof

of Authority (PoA) configuration) where each of them contains Ethereum transaction data, smart contract bytecodes, and the blockchain itself. The business owners can run such nodes in their local systems. Alongside the Ethereum clients, a business owner would also run a local database service for storing their raw farm data. The raw data contain animal data that include inspections, vaccinations, movements, and other relevant information. Each time a local database is updated, a cryptographic hash is computed by combining all the raw data of that farm, and it is stored in the blockchain (via the Farm Manager contract). This hash links the blockchain with the local database. Additional information on the local database operation is available in Appendix C. Privacy and a sense of ownership are the two key reasons we chose to keep the raw databases local to respective business owners. The system is more resistant to eavesdropping, and a business owner is in control of how his/her data are used or shared. However, to enforce data immutability and create mutual trust, we use the blockchain to enforce that any alteration in the local data must be reported with an updated hash. Hence, any prospective purchaser or authorized auditing entity can validate whether, how, and when data have been modified from the original state.

5.3.1 Algorithmic procedures

The proposed blockchain-based animal farm management framework will be able to perform a variety of tasks. We demonstrate some of the basic tasks in this section.

User profile and farm management

Several smart contracts in the system enforce user and farm policies to keep data secure and identities anonymous. The Ethereum account that deploys a smart contract is automatically assigned as the administrator (admin) for that contract. The admin has specific access permissions to perform tasks related to management. Despite that, the admin and all outside accounts are prevented from accessing and modifying user and farm data. All users of the system (owners, managers, viewers/auditors) must be registered by an admin using the `ProfileManager` contract. All the farms owned by the users must also be registered by the

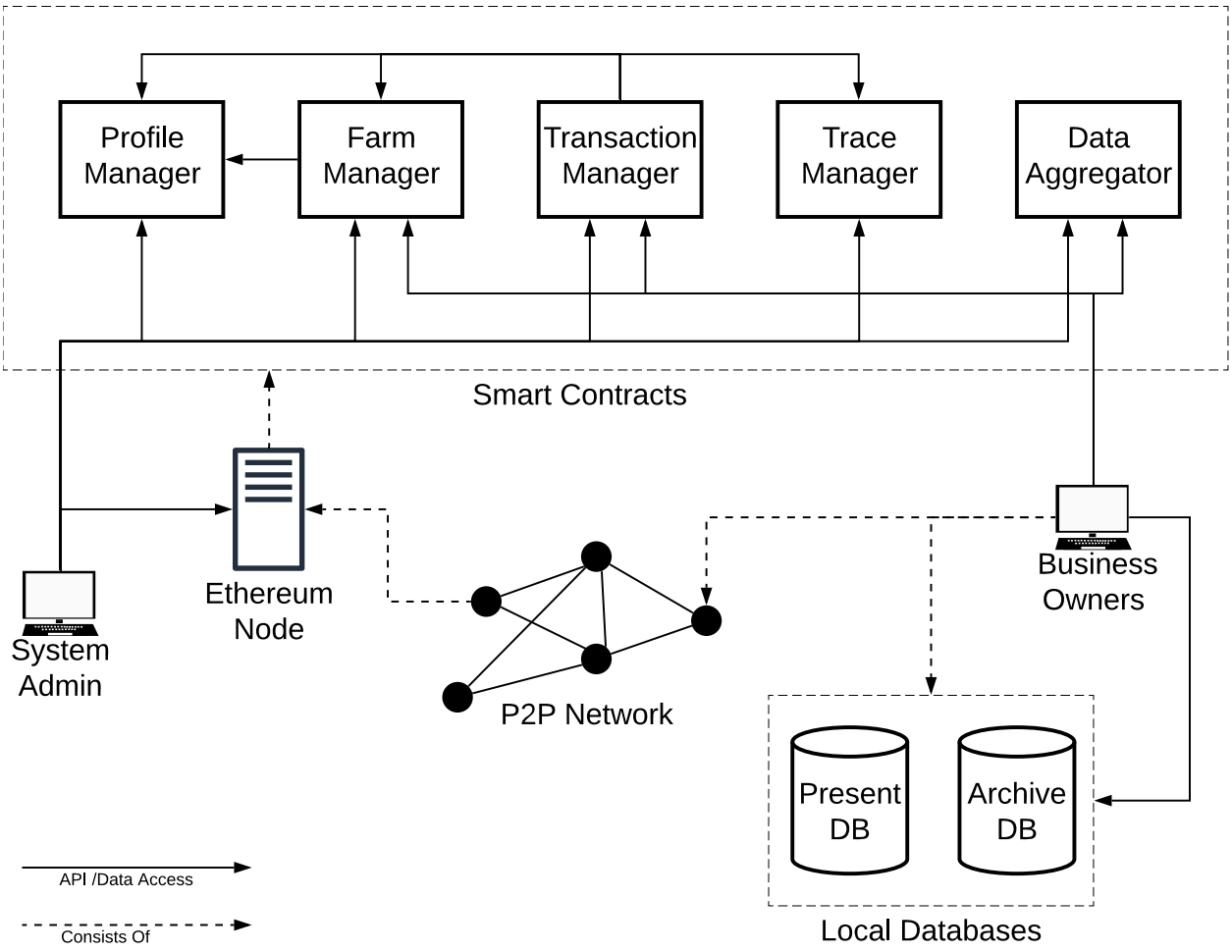


Figure 5.2: *A simplified block diagram of the blockchain based farm animal management system. The blockchain network is shown in the bottom where the solid circles represent blockchain nodes (clients). Each blockchain client locally stores smart contracts, associated data, and the respective farm animal databases.*

admin using the `FarmManager` before they can participate in the system.

Someone willing to join the network first contacts the network admin. The prospective user provides the admin with a request containing the access level desired (`viewer(1)`, `manager(2)`). This is off-chain communication and is completed using traditional methods (emails, messages, letters). Once a request is received and approved, the admin generates a user id which is a 160-bit Ethereum address. The admin registers the user by calling the `registerUser()` method. During this function call, the user ID and the access level are sent as arguments. The `ProfileManager` contract validates whether the function call is coming from an admin, and it checks if the requested user already exists in the system. Upon validation, the Profile Manager registers the user. The admin notifies the user with the confirmation and the user's Ethereum address. This address is not shared with anyone else to keep the user identity secret. Once registered, the user can access other contracts of the system. The admin can change the user access level by calling the `updateUser()` method. If a user needs to be deactivated/restricted/terminated, the admin can set the access level to 0.

A farm owner who is willing to register his/her farm requests the network admin in an off-chain communication. This request contains the owner's Ethereum address that is already registered in the system. The admin generates a farm id which is also a 160-bit Ethereum address. The admin calls the `registerFarm()` method of `FarmManager` contract to register the farm in the system. The `FarmManager` contract validates whether the function call is coming from an admin. It also checks if the farm id already exists in the system or not. Once registered, the `FarmManager` contract contains both the farm id and the owner id in a mapped data structure. After the registration is completed, the farm owner can regularly update farm information by calling the `updateFarmInfo()` method of the `FarmManager` contract. During such calls, the animal count and the farm hash are updated by the farm owner.

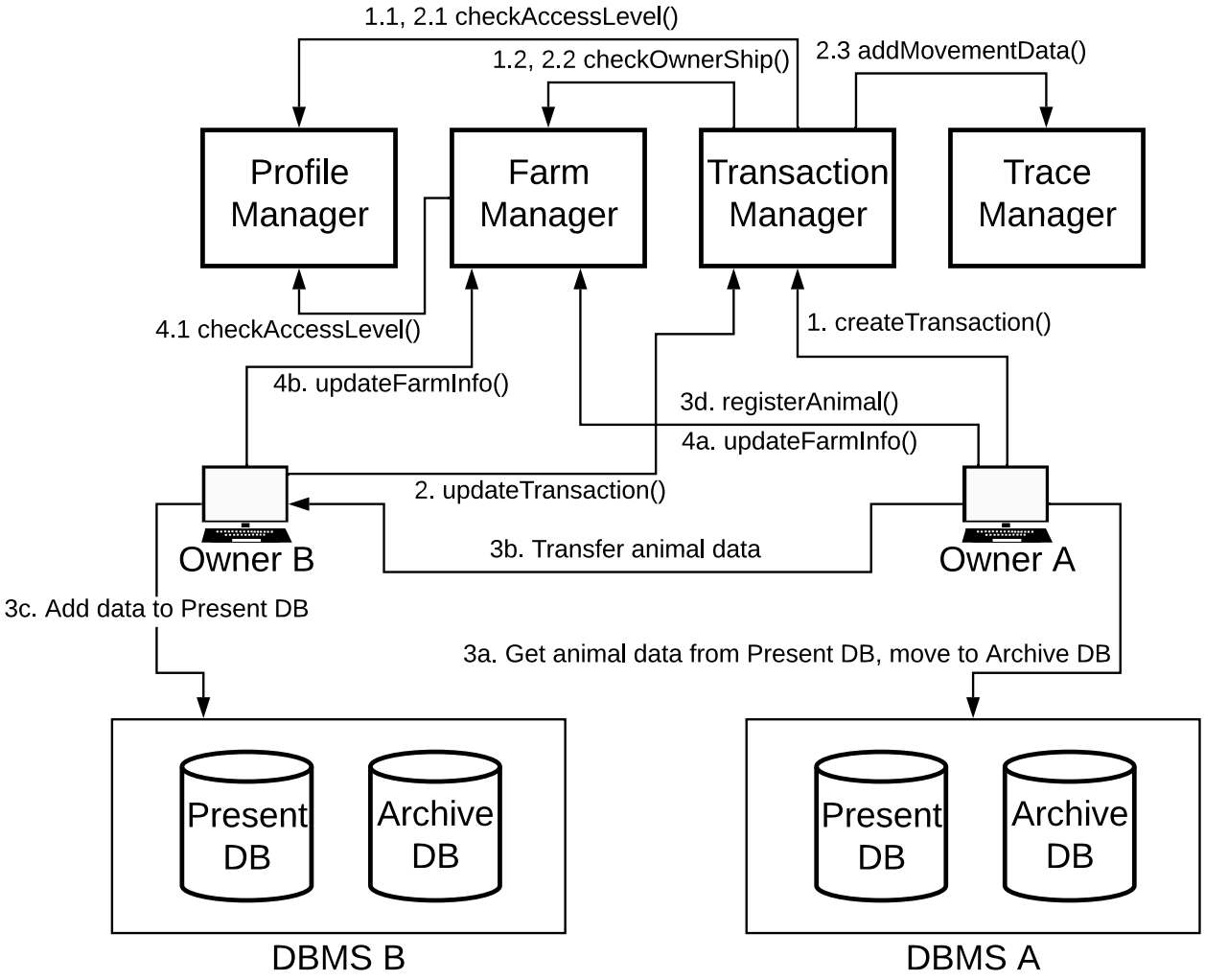


Figure 5.3: *Detailed block diagram illustrating communication steps among the system components during a business transaction.*

Process of business transactions

A typical example of a financial transaction is shown in Figure 5.3. We create a typical scenario where two parties (farm owners) A and B exchange some animals. Creating the transaction is a two-step process where both owners need to communicate with the `TransactionManager` contract (Steps 1 and 2). In the first step, one farm owner proposes a transaction by calling the `createTransaction()` method. This process creates a `Transaction` object in the contract with several properties, including the addresses of both owners, farm addresses, and the full list of animals being transferred. Animals tags are stored in the list. In the second step, the other owner confirms the transaction using the `updateTransaction()` method. When a farm owner communicates with the `TransactionManager` contract, the contract validates permissions and ownership of the farms between which the transaction would take place. These are marked as sub-steps 1.1, 1.2, 2.1, and 2.2 in Figure 5.3. Upon confirmation, the `TransactionManager` stores anonymous movement data in the `TraceManager` contract (step 2.3). After the confirmation, the seller/exporter of animals (Owner A in Figure 5.3) retrieves data of the animals in the transaction from its local database (step 3a) and transmit directly to the buyer/importer (step 3b). The buyer/importer stores that data in his/her local database (step 3c). The seller/exporter registers the sold animals to the new farm (step 3d). The seller also removes the sold animal data from its current database and archives them if necessary. As both the party had their databases modified, they will update the `FarmManager` contract with freshly computed farm hashes (steps 4a and 4b). The `FarmManager` validates their permissions and updates the hash values.

Animal tracing process

Our framework provides the ability to trace animals using a `TraceManager` contract. Trace data are stored in the contract during a business transaction (See Figure 5.3, Step 2.3). There can be several reasons why someone would want to trace an animal back to the origins. One such scenario could be when a farm wants to use its reputation to gain prospective customers

by giving them ways to securely verify whether their animal/food product came from that farm or not. Another scenario could be tracing the source of infection during an outbreak to mitigate the problem rapidly. The `TraceManager` contract stores animal movement data in a nested key-value map structure. Each animal is identified by its id (tag), which is used as the key. A custom-defined data structure `Animal` is considered to be the value. This structure contains another map listing all the movements, each of which is an instance of the structure `Movement`. We use simple indices as keys to the movement map and keep track of the total number of movements (hence, movement entries). The admin can call the method `getMovementCount()` and provide the animal id to know the total number of movement entries. The admin can then call the `getMovementdata()` method with animal id and movement index as arguments to get the actual movement entry. A movement entry contains the source farm address, the destination farm address, and the time when the movement was recorded in the blockchain.

Data aggregation procedure

Sometimes the scientific community or the industry can benefit from summary data on animal production. Examples of such data can be the average weekly growth of certain breeds, the effect of a vaccine, or growth improvements of animals under certain diets. In situations like these, it is needed to provide such information without jeopardizing the privacy of the business. Hence, we implement a `DataAggregator` contract to help with anonymous data collection. To handle the data collection, the admin first creates a `Dataset` object in the `DataAggregator` contract by calling the `createDataset()` method. The method requires a key and a secret as arguments. If the method succeeds, the key would be used to locate the `Dataset` object, and the secret would be used to authenticate the accounts performing read/write operations on the dataset. The admin is responsible for generating, maintaining, and distributing this key-secret pair. Using off-chain communications, the admin can request the farm owners to provide certain summary data. These requests contain a description of the data requested and key-secret pair for the dataset. Upon receiving such a request, the farm

owners may or may not participate in the survey. A survey participant calls the `addData()` method to submit data, and this method requires the actual data, the key, and the secret as its arguments. Once the submission period ends, the admin may obtain the stored data using the `getData()` method, which also requires the key and the secret. Only the farms participating in the data sharing will be able to access the aggregated information.

5.4 System analysis

In this section we evaluate our system and explain how it handles anonymity of users, data security.

5.4.1 User privacy

The private Ethereum blockchain users can operate in the system without providing information that can identify them or their location. Each user is provisioned by the system administrator when an externally owned account (EOA) in Ethereum is generated. These accounts have several components, including private keys and Ethereum addresses. Private keys are always securely stored by users and never revealed outside. The public keys are derived from private keys. The addresses are derived from public keys using the Keccak-256 hash function, where the last 20 bytes (LSB) of the hash are kept. The one-way hash function prevents association back to the public key. Hence, even after knowing someone's address, one cannot derive the identity of that person.

Due to the system configuration as a private blockchain network, outsiders cannot enter the system without proper authorization from the administrator. If someone already in the system wants to impersonate another user, it would be automatically prevented if private keys are not accessible by anyone other than the legitimate owner.

5.4.2 Data security

Each farm business may choose to store animal-related data locally on its premises. A locally running relational database (SQL) management system would contain tables of data as shown in Figure C.1. Each time the tables are updated, a cryptographic hash is generated for each table using a hash function. Eventually, all the table level hashes are combined to create a single hash which is stored in the `FarmManager` contract by the user. This technique ensures that a single owner cannot alter data without updating the hash in the Ethereum blockchain. If one needs to verify the integrity of data, he/she can do so by recalculating the hashes from the data. To enable faster verification of data, the `FarmManager` contract also stores hashes related to individual animal data in its state variables. When animals are transferred during a transaction, animal data are sent to the new owner (steps 3a, 3b, and 3c in Figure 5.3). The new owner can check the animal hashes stored in the `FarmManager` contract and verify if they match the data by recomputing hashes from the received data.

5.4.3 Provenance

The `TraceManager` contract stores trace data for every animal moved from one farm to another. In addition to that, every farm owner keeps records of animals' past movement histories with cryptographic hashes as proofs in the blockchain system. While the cryptographic hashes ensure the integrity of the data, the movement records in the `TraceManager` contract provide faster tracing without requesting data from independent local databases. If someone wants to fabricate the origin of an animal or animal product, he/she will face two major obstacles: i) the hashes stored for the animal movement data won't match and ii) the trace data captured by the `TraceManager` would not match. As a hypothetical situation, let's assume the person with the intent of corrupting animal trace data forms a coalition with all the owners involved in the animal transfer process. As the owners control their respective local databases, they could theoretically store and exchange fabricated movement data and make it look real. However, the `TraceManager` prevents such fabrication by making the tracing process automatic. It directly captures data from the `TransactionManager`. For

that reason, the entities (owners and farms) involved in a transaction cannot fabricate their addresses as they can only authenticate with their IDs.

5.4.4 Secured data aggregation

While surveys and collections of data are primarily beneficial, owners may still be discouraged from a business perspective if they fear a loss of privacy. Our system contains a smart contract to facilitate anonymous data collection. It uses a simple key-secret pair to authenticate a submitter during data collection and does not use any other identification mechanism that others can exploit. Let's assume a hypothetical scenario where the admin itself is corrupted in its ability to protect user privacy. Although the admin has a basic idea of who a user is due to user profile and farm ID management, it does not have access to the user's or farm's data. If the admin conducted the surveys in a direct manner to collect user data, it could identify who sent the data. However, the `DataAggregator` prevents this issue by not keeping records of survey participants. The admin, who has access to aggregated data, can only know how many entries were submitted and what was submitted.

5.4.5 Fairness of the system

The system provides comparable levels of accessibility, security, and privacy to its users. The consensus mechanism does not differentiate among the participants. The lack of central authority eliminates bias and gives power back to the individual entities. The local data ownership mechanisms indicated in Figure 5.2 provide users with more control over their private business data. The origin tracking (provenance) makes the system fair for the consumers who lie at the end of the chain. Consumers can verify the authenticity of claims about beef products produced by business entities.

5.4.6 Reliability

Traditional server-centric systems have a vulnerability commonly referred to as a single point of failure. The redundancy introduced by the proposed framework, which takes advantage of

the Ethereum blockchain, can mitigate this issue. We categorize failures into two types: link failure and node failure. A link failure is defined as the situation when a link connecting two blockchain nodes is unable to sustain communication. On the other hand, a node failure is defined as the scenario when a node cannot communicate with the network via any of its links or any data stored on the node is lost. As we are dealing with a peer-to-peer (P2P) network, link failures do not affect operations unless enough links fail to disconnect or isolate a node. A fully connected network with n nodes has $n(n-1)/2$ links (maximum number of edges in an undirected graph). To keep the network fully operational, we need at least $n-1$ links (minimum number edges needed to maintain connectivity in an undirected graph). Hence, up to $(n-1)(n-2)/2$ links can fail, and the system can still operate. Note, this number depends on which links are failing. In worse cases, when all links connecting to a node fail, that node becomes isolated. A user may still connect to the network via any alternate node as the credentials are valid across the system. Once the failed links recover, the node can come out of isolation and re-sync all data. A node failure has the same effect as the complete link failure described above. If data is lost or damaged due to a node failure, the node can re-sync once a connection is re-established. Data loss is tolerable up to $n-1$ nodes as every node contains all the blockchain-related data and smart contract bytecodes.

5.4.7 Computational costs

In Ethereum, the gas cost is a measure of how much computational resource or storage is needed to complete a transaction or a smart contract operation. Any operation that changes the Ethereum virtual machine (EVM) state creates a transaction and every such transaction has an associated gas cost. Although gas costs imply spending real money in Ether when the main Ethereum network is used, it is not the same in our case. In private chains, Ether has no value. Nevertheless, it is an available measure of system resource usage. We demonstrate the gas costs of contract creation in Figure 5.4 and the costs of calling some commonly used methods in Figure 5.5. In these figures, a higher gas cost indicates that more processor cycles, dynamic memory, or persistent storage are required to complete an operation. It is

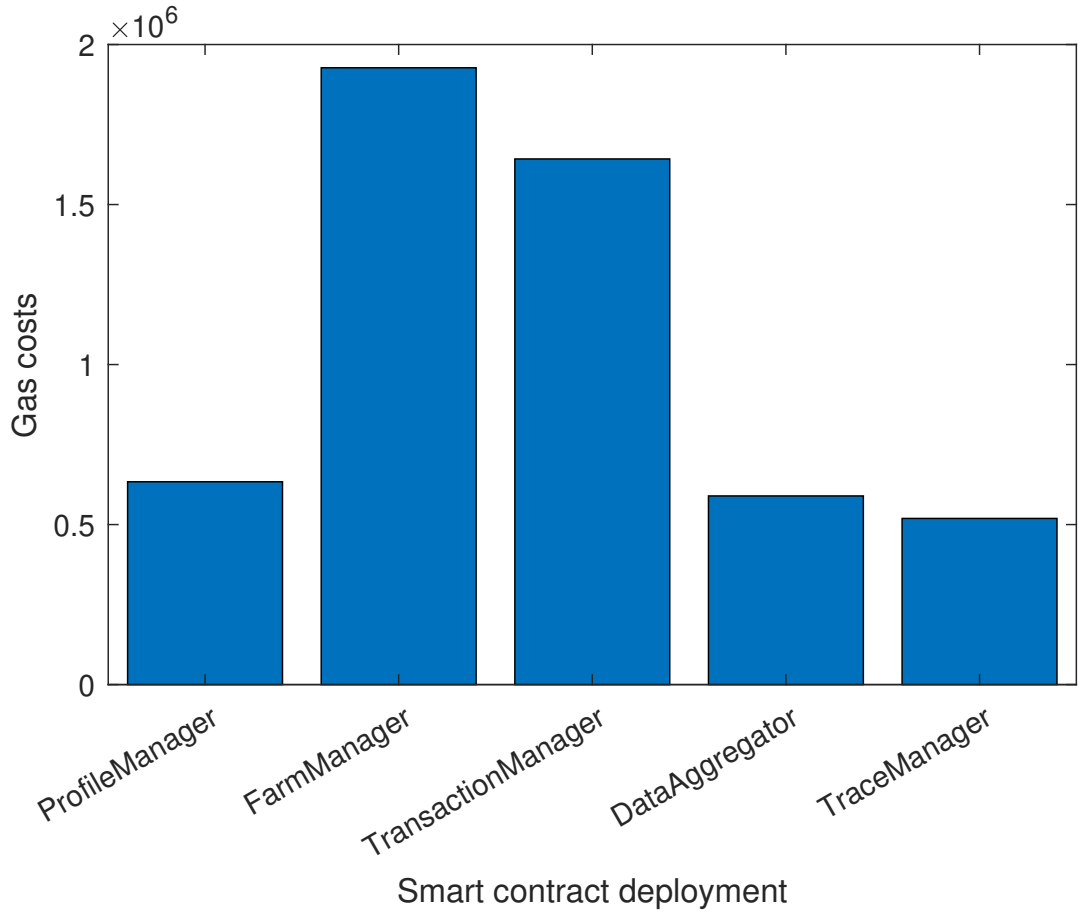


Figure 5.4: *A comparative chart showing contract creation (deployment) costs for the five proposed smart contracts. Gas cost is a measure of computational resources (processing power and storage) needed for the operation.*

important to note that read-only methods (that do not change any data in the memory or the state of the chain) do not create transactions at all. As every node has a snapshot of the system, such methods simply read data from the local running node.

The `FarmManager` and the `TransactionManager` contracts implement a large number of functionalities (See Figures B.2 and B.3), hence, it is no wonder that, they cost a lot more gas (about three times more) compared to others. The contract deployments are one-time operations that are done at the beginning of the system setup. The operations depicted in Figure 5.5 can occur at any arbitrary time once the system is running. These methods change the system's state by adding or modifying data in both memory and persistent storage. The methods related to business transactions and movement data consume more resources,

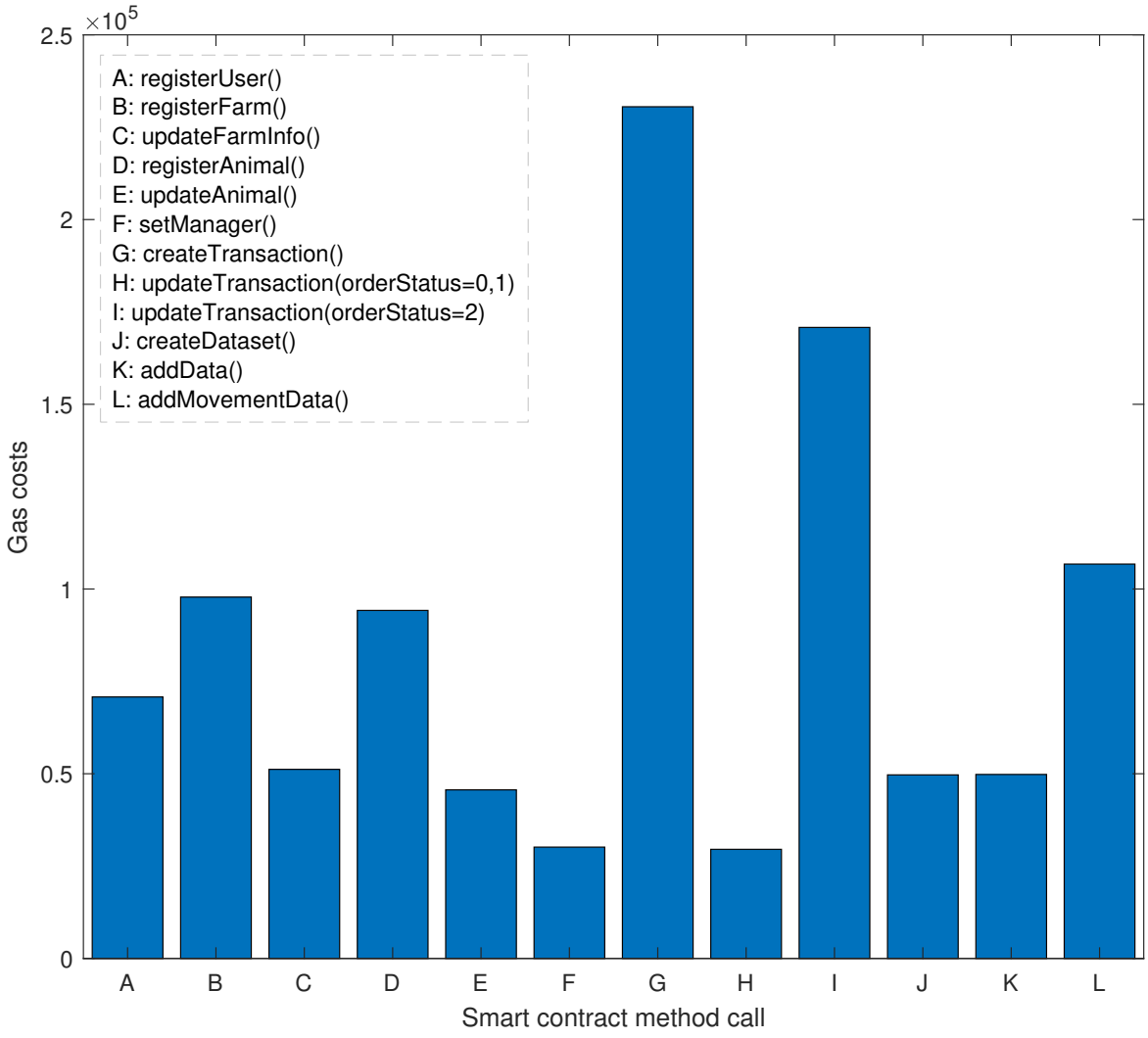


Figure 5.5: *A comparative chart showing costs of calling different smart contract methods (functions) that change the state of the system. Read only (view) methods are not included here. Gas cost is a measure of computational resources (processing power and storage) needed for the operation.*

mostly due to their space complexity. They create new objects and add new information. The `updateTransaction()` method has two distinct scenarios. When a transaction is `confirmed` (`orderStatus = 2`), it stores the movement data (internally calls the `addMovementData()` method). Hence, it requires significantly more gas compared to the other scenarios such as an order being `proposed` (`orderStatus = 1`).

5.4.8 Integration test

We test several operational scenarios with a prototype running system. The configuration of the prototype where these scenarios are simulated and tested is described in Appendix D. Every scenario consists of doing some tasks to simulate a situation. These test scenarios along with the outcomes are described in Tables 5.1, 5.2, and 5.3. Every case is classified as either positive (P) or negative (N). The positive cases are those where all the communications done with the system are expected and valid by design. The negative cases are those where one or more operations done with the system are defined as illegal or a failure in authentication or validation. Each entry in the 'Test Case' column of the three tables (5.1, 5.2, and 5.3) contains test case number # and classification (P or N), followed by the test case name. For example, [1P] stands for test case 1, which is positive. The system is designed to provide a secure environment and ensure any breach of data or operational access. The logs and results generated for few of the test cases are shown in Figures 5.6, 5.7, and 5.8. The complete set of results can be found in the supplementary materials document. The table validates how our proposed system is temper-proof and provides data security and traceability while maintaining user anonymity.

5.5 Discussion

This chapter proposed a blockchain-based supply chain management framework to be used in the US beef cattle industry. We explained in detail how this system would operate and communicate with various entities involved. Finally, we analyzed how the framework will

Test Case	Prerequisites	Tasks	Results
[1P] Authorized farm owner attempts to update farm info	i) Admin registers the farm owner (node 2) with access level 2. ii) Admin registers the farm (farm A) and appoints ownership (farm A - node 2) to the appropriate owner.	The farm owner (node 2) calls `updateFarmInfo()` to update farm (farm A) information.	Status code 202. The `FarmManager` contract updates the information when the next block is mined.
[2N] Unregistered user attempts to update farm info	The user (node 4) is never registered to the system.	Unregistered user (node 4) calls `updateFarmInfo()` to update farm (farm A) information.	Status code 403. The `FarmManager` contract rejects the request as the validation fails.
[3N] Registered user attempts to update info of a farm that is not owned	i) The user (node 3) is registered with access level 2. ii) The farm (farm A) is never assigned as owned by the user (node 3).	The user (node 3) calls `updateFarmInfo()` to update farm info for the farm (farm A) that is not owned by it.	Status code 403. The `FarmManager` contract rejects the request as the validation fails.
[4P] Authorized farm owner attempts to create a business transaction with another authorized owner	i) Admin registers the users (node 2 and node 3) with access level 2. ii) Admin registers the farms (farm A and farm B) and appoints ownership (node 2 - farm A and node 3 - farm B) to the respective users.	One farm owner (node 2) calls `createTransaction()` to create a business transaction of several animals to be transferred from one farm (farm A) to another farm (farm B).	Status code 202. The `TransactionManager` contract accepts the proposal upon validation and registers a transaction when the next block is mined.
[5N] Unregistered user attempts to create a business transaction	The user (node 4) is never registered to the system.	Unregistered user (node 4) calls `createTransaction()` to create a business transaction.	Status code 403. The `TransactionManager` contract rejects the request as the validation fails.

Table 5.1: *Integration test procedures and results (cases 1 - 5)*

Test Case	Prerequisites	Tasks	Results
[6P] Admin attempts to retrieve trace data	At least one transaction must be created and confirmed. Note: Here the transaction 1581115549157 moved the animals [1515334092888, 1515334092898] from farm A (`0x4FB4...FA26`) to farm B (`0x4491...830C`).	i) The admin calls the `getMovementCount()` method to know the number of movements recorded for an animal (1515334092898). ii) The admin calls the `getMovementData()` method to get each row of the movement entries for an animal (1515334092898)	The `TraceManager` contract responds to the method calls and returns the results.
[7N] Unauthorized user attempts to retrieve trace data	At least a single transaction must be created and confirmed. Note: Here the transaction 1581115549157 moved the animals [1515334092888, 1515334092898] from farm A (`0x4FB4...FA26`) to farm B (`0x4491...830C`).	i) Any user who is not an admin (node 4) calls the `getMovementCount()` method to know the number of movements recorded for an animal (1515334092888). ii) Any user who is not an admin (node 4) calls the `getMovementData()` method to get each row of the movement entries for an animal (1515334092888),	The `TraceManager` contract responds with null value (0) as the validations fail.
[8P] Invited user attempts to submit survey data	i) Admin registers a data set with a key and a secret. (We use key: 1001, secret: 2020) ii) The key and the secret are mailed to every survey participant.	An user who is a participant (node 4) calls the `addData()` method to submit some data to the data set.	Status code 202. The `DataAggregator` contract accepts the submission upon validation and stores it when the next block is mined.

Table 5.2: *Integration test procedures and results (cases 6 - 8)*

Test Case	Prerequisites	Tasks	Results
[9N] Unauthorized user attempts to submit survey data	i) Admin registers a data set with a key and a secret. (We use key: 1001, secret: 2020) ii) The test user gets the incorrect key and secret (1003, 2019).	An uninvited user (node 3) calls `addData()` method with wrong key-secret pair to submit some data to the data set.	Status code 403. The `DataAggregator` contract rejects the submission upon validation.
[10P] Admin attempts to retrieve summary data	A data set must be registered and at least a single participant must submit some data. Note: Here the data set with key: 1001 and secret: 2020 is used. A user (node 4) submitted some data (0x44) in the survey.	i) The admin calls the `getDataCount()` method to know the number of data entries recorded in the survey. ii) The admin calls the `getData()` method to get each row of data collected in the survey.	The `DataAggregator` contract responds to the method calls and returns the results.

Table 5.3: *Integration test procedures and results (cases 9,10)*


```
"from": "0x768B3EF0467EdB5E5e904E69A5E4AB2310B43429",
"topic": "0x2a393e5acf4fd5f9824b7de658976e4ab9fb7322287eafe28a4c2346eb16fa80",
"event": "ReturnStatus",
"args": {
        "0": "202",
        "1": true,
        "code": "202",
        "status": true,
        "length": 2
-
```


Figure 5.6: *The logs generated as a response by the* `FarmManager` *(with address* `0x768B...3429`*) contract when an authorized farm owner attempts to update farm info. This result is generated from test case 1 of Table 5.1.*

```
"from": "0x51b69a37142a942cd5FBCCB8811316459C8273C94",
"topic": "0x2a393e5acf4fd5f9824b7de658976e4ab9fb7322287eafe28a4c2346eb16fa80",
"event": "ReturnStatus",
"args": {
        "0": "403",
        "1": false,
        "code": "403",
        "status": false,
        "length": 2
}
```


Figure 5.7: *The logs generated as a response by the* `TransactionManager` *(with address* `0x51B6...3C94`*) contract when an unregistered user attempts to create a business transaction. This result is generated from test case 5 of Table 5.1.*

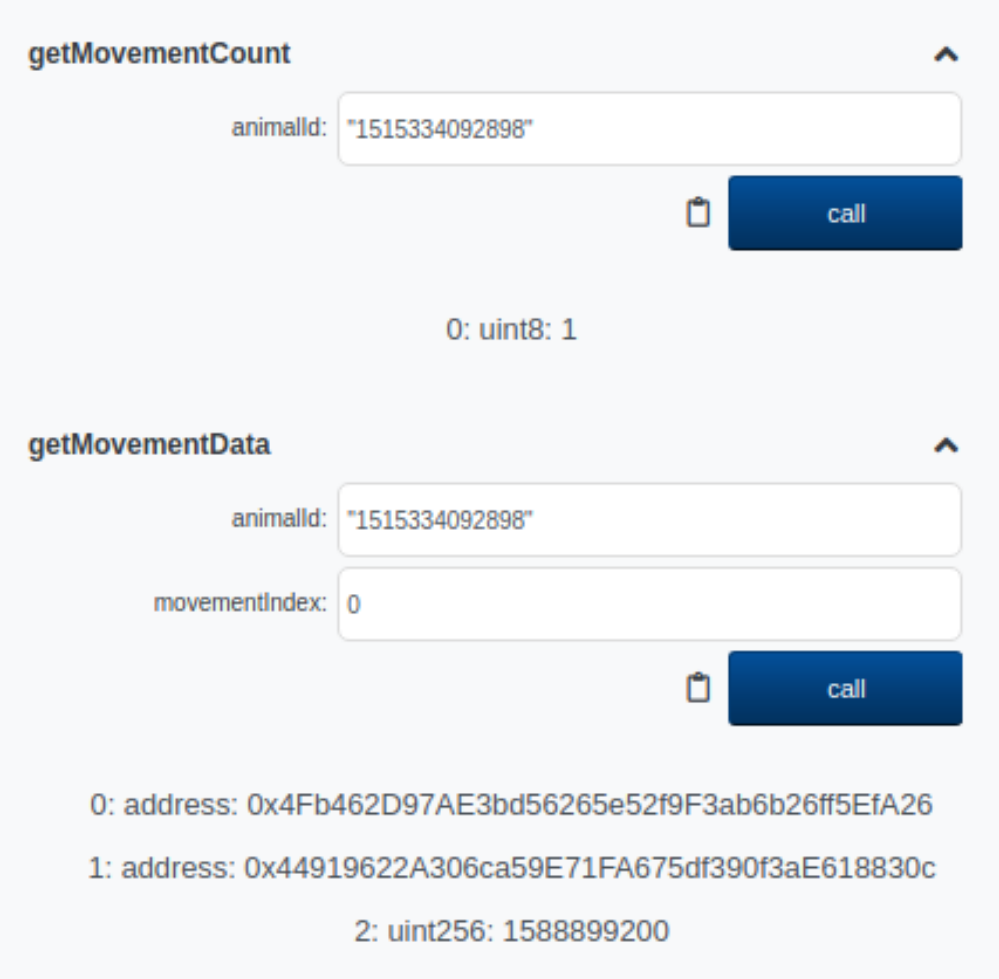


Figure 5.8: *The returned outputs when the authorized admin attempts to read trace data by calling the methods provided by the* `TraceManager` *contract. Here, the results indicate an animal with id* `1515334092898` *was moved from farm A (*`0x4FB4...FA26`*) to farm B (*`0x4491...830C`*). This result is generated from test case 6 of Table 5.2.*

ensure user anonymity, improve data privacy, and ensure trace data integrity. We performed integration tests to evaluate the system operation in various scenarios.

The proposed framework operates as a private / consortium blockchain and uses proof of authority (PoA) for achieving consensus, eliminating the computationally expensive hash computations. This enables us to run complex smart contract functions and store more data on smart contracts to facilitate traceability and improve data security. The database can be hosted on any SQL database management system by appropriately configuring the specified schema. The smart contracts can be run on any Ethereum based clients (geth, parity, etc.). The system requires an admin who will initiate everything and register the users and their farm businesses. This could be a potential weak point as the admin will know the mappings of the user and the farm addresses. Hence, despite being a trustless, decentralized architecture dealing with supply chains, the framework requires some trust. However, smart contracts are carefully designed to isolate private data from admins and any unintended users. The owners do have control over who can see their data. Different designs in the architecture may affect privacy, security, and resource requirements. This design is optimized for the US beef cattle industry, focusing on privacy, data ownership, and security. The system runs optimally with traditional computational resources and does not require any special hardware. The technical knowledge on blockchains is also commonplace, given that it is being used in varying types of industries, including cryptocurrencies. However, blockchains do have scalability issues when it comes to storage. With time, the requirements for persistent storage would increase. Ethereum supports light client nodes to help with such an issue. In the future, it may also be possible for the blockchain systems to discard data old enough to become irrelevant. Blockchains also face adaptability issues as an emerging technology and require their users to go through a learning curve.

Our proposed system provides a technical solution to several specific issues encountered by the US farm industry. It improves the existing knowledge of how blockchains can meet specific industrial needs such as identity protection and data ownership while ensuring immutability and product traceability. The framework can be used in other supply chains with minor modifications. Future work can focus on improving the scalability of blockchains in

general. While different industries have varying requirements, most can benefit from scalable, secured, and efficient blockchain frameworks.

Chapter 6

Conclusion

6.1 Summary

This dissertation aimed to solve research problems associated with infectious disease modeling, prediction of outbreaks, and data related to infectious diseases. It presented a network-based host-vector spreading model, network generation algorithms, data-driven feature enhancement techniques for machine learning, and a blockchain-based farm animal data management system. It also presented a substantial amount of analyses on model parameters, infectious outbreak-related data, and results produced by the tools and techniques presented.

First, the modeling problem associated with Zika virus infection was solved in Chapter 2 using an interconnected model that supports both modes of transmission (insect-vectored and sexual) by coupling a heterogeneously mixed host population with a homogeneously mixed vector population. I investigated effects of seasonal variations of the mosquito population, sexual transmission, and several model parameters on Zika virus outbreak dynamics using this model. Additionally, a sensitivity analysis compared contributions of key model parameters to outbreak size, outbreak duration, and vector-free pathogen survival in the network. Another vector-borne disease, dengue fever, was modeled using a machine learning approach in Chapter 3. A framework was presented for feature ranking and selection using nearby incidence data, which remedied the problem of incidence data inadequacy. This framework

achieved improvements in prediction performance for most of the Brazilian municipalities I tested.

The next set of problems stemmed from a culture of secrecy in U.S. farm animal businesses. One problem was the lack of data to construct farm animal movement networks. In my first approach, this was resolved with a network generation algorithm in Chapter 4. The algorithm can use limited available data from agricultural census and small-scale surveys to generate and extrapolate movement networks that follow given node degree distributions. A generated network was used to simulate spread of the African swine fever virus (ASFV) and analyze effectiveness of two targeted control measures: movement restrictions and hypothetical vaccinations. Control measures were targeted to nodes ranked using network centrality metrics. In comparing control measures' effectiveness, I found that farm nodes with many incoming movements (high node in-degree) should be vaccinated or isolated first for effective disease control. To address the data secrecy issue with farm businesses, I took a different approach in Chapter 5 by presenting a blockchain-based decentralized database management framework. This framework provides data security, user anonymity, and animal traceability using a private network that allows business owners to share data without losing data ownership. Additionally, smart contracts implement business policies, which can be adapted based on application-specific requirements.

Overall, the contributions have provided several solutions for problems critical in infectious disease modeling, analysis, and control.

6.2 Future works

Trade-offs arise between model accuracy and computational efficiency. Networks used in Chapters 2 and 4 are static, where node interactions are averages over a time period. In reality, host contacts change with time. An infected host can only infect its present and future contacts depending on the duration of the infection. Thus, dynamic networks where contacts change with time can add another degree of accuracy in exchange for computational efficiency. Improving efficiency of dynamic networks so that epidemic simulations can

be performed on many nodes remains an open problem. On the other hand, data-driven dengue prediction models in Chapter 3 do not discriminate among viral strains. It can be difficult to obtain detailed surveillance data with viral strain information. If such data are available, predictive models should consider including that information in the set of features (predictors). Machine learning methods can always benefit from feature selection tailored by our knowledge of specific infectious diseases instead of using naive assumptions about data. The feature selection method proposed in Chapter 3 is an example of one such tailoring. As we understand a complex process more and more, the knowledge can be incorporated into the model design, feature engineering, and feature selection. This is a shift from purely black-box modeling, which has benefits similar to mechanistic approaches. When there are many features, redundancies can be remedied by removing highly correlated incidence data from selected feature sets.

The farm animal data management framework proposed in Chapter 5 can be expanded in several ways. Additional incentives for different parties can be added to attract participation and promote ethical practices from a business perspective. The framework assumes that initial data entered into the system by business owners are correct. A verification step could be added in the real world. A rating system can also be implemented to reward honesty and punish unethical practices. Trade-offs occur between traceability and privacy from a design perspective, depending on when trace data are collected. The system presented records trace data when animal movement takes place. This approach may interfere with data secrecy requirements set by some businesses. Another strategy is to request trace data from business owners when an emergency (e.g., infectious outbreak) occurs. This approach will give owners more control over their data. However, the process can be slow or even obstructive, depending on individual responses to data requests. The unique properties of blockchains open numerous possibilities for future developments in managing data related to infectious diseases.

Bibliography

[1] Tanvir Ferdousi, Lee W Cohnstaedt, D Scott McVey, and Caterina M Scoglio. Understanding the survival of Zika virus in a vector interconnected sexual contact network. *Scientific Reports*, 9(1):7253, 2019.

[2] Tanvir Ferdousi, Sifat Afroj Moon, Adrian Self, and Caterina Scoglio. Generation of swine movement network and analysis of efficient mitigation strategies for African swine fever virus. *PLOS ONE*, 14(12):e0225785, 2019.

[3] Tanvir Ferdousi, Don Gruenbacher, and Caterina M Scoglio. A Permissioned Distributed Ledger for the US Beef Cattle Supply Chain. *IEEE Access*, 8:154833–154847, 2020.

[4] Matt J Keeling and Ken TD Eames. Networks and epidemic models. *Journal of the Royal Society Interface*, 2(4):295–307, 2005.

[5] Mark Newman. *Networks*. Oxford University Press, 2018.

[6] Ken TD Eames and Matt J Keeling. Contact tracing and disease control. *Proceedings of the Royal Society of London. Series B: Biological Sciences*, 270(1533):2565–2571, 2003.

[7] Sifat A Moon and Caterina M Scoglio. Contact tracing evaluation for COVID-19 transmission in the different movement levels of a rural college town in the usa. *Scientific Reports*, 11(1):1–12, 2021.

[8] Narges Montazeri Shahtori, Tanvir Ferdousi, Caterina Scoglio, and Faryad Darabi Sahneh. Quantifying the impact of early-stage contact tracing on controlling ebola diffusion. *Mathematical Biosciences & Engineering*, 15(5):1165, 2018.

[9] Stephen P Borgatti and Martin G Everett. A graph-theoretic perspective on centrality. *Social networks*, 28(4):466–484, 2006.

[10] Fredrik Liljeros, Christofer R Edling, Luis A Nunes Amaral, H Eugene Stanley, and Yvonne Åberg. The web of human sexual contacts. *Nature*, 411(6840):907–908, 2001.

[11] Luis A Nunes Amaral, Antonio Scala, Marc Barthelemy, and H Eugene Stanley. Classes of small-world networks. *Proceedings of the National Academy of Sciences*, 97(21): 11149–11152, 2000.

[12] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific American*, 288(5):60–69, 2003.

[13] Stephane Helleringer and Hans-Peter Kohler. Sexual network structure and the spread of HIV in Africa: evidence from Likoma Island, Malawi. *Aids*, 21(17):2323–2332, 2007.

[14] P Erdös and A Rényi. On random graphs I. *Publ. Math. Debrecen*, 6:290–297, 1959.

[15] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[16] Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[17] Daniel T Gillespie. Exact stochastic simulation of coupled chemical reactions. *The Journal of Physical Chemistry*, 81(25):2340–2361, 1977.

[18] Michael I Jordan and Tom M Mitchell. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

[19] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Computing Surveys (CSUR)*, 50(6):1–45, 2017.

[20] Stuart Russell and Peter Norvig. *Artificial intelligence: A Modern Approach*. Pearson, 2002.

[21] Douglas M Hawkins. The problem of overfitting. *Journal of Chemical Information and Computer Sciences*, 44(1):1–12, 2004.

[22] Christopher Olah. Understanding LSTM networks, 2015, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs.

[23] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[24] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

[25] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[26] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[27] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

[28] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to Recommender Systems Handbook. In *Recommender Systems Handbook*, pages 1–35. Springer, 2011.

[29] Jerome T Connor, R Douglas Martin, and Les E Atlas. Recurrent neural networks and robust time series prediction. *IEEE transactions on neural networks*, 5(2):240–254, 1994.

[30] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318. PMLR, 2013.

[31] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*, 2014.

[33] Gopala K Anumanchipalli, Josh Chartier, and Edward F Chang. Speech synthesis from neural decoding of spoken sentences. *Nature*, 568(7753):493–498, 2019.

[34] Alex Graves, Santiago Fernández, Marcus Liwicki, Horst Bunke, and Jürgen Schmidhuber. Unconstrained online handwriting recognition with recurrent neural networks. In *Advances in Neural Information Processing Systems 20, NIPS 2008*, 2008.

[35] Hansika Hewamalage, Christoph Bergmeir, and Kasun Bandara. Recurrent neural networks for time series forecasting: Current status and future directions. *International Journal of Forecasting*, 37(1):388–427, 2021.

[36] Dave Bayer, Stuart Haber, and W Scott Stornetta. Improving the efficiency and reliability of digital time-stamping. In *Sequences II*, pages 329–334. Springer, 1993.

[37] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008.

[38] Zibin Zheng, Shaoan Xie, Hongning Dai, Xiangping Chen, and Huaimin Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. IEEE, 2017.

[39] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[40] Bitcoin Wiki. Block hashing algorithm, 2021. URL https://en.bitcoin.it/wiki/Block_hashing_algorithm.

[41] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151:1–32, 2014.

[42] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, page 30. ACM, 2018.

[43] Shangping Wang, Yinglong Zhang, and Yaling Zhang. A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems. *IEEE Access*, 6:38437–38450, 2018.

[44] Dinh C Nguyen, Pubudu N Pathirana, Ming Ding, and Aruna Seneviratne. Blockchain for secure EHRs sharing of mobile cloud based e-Health systems. *IEEE access*, 7: 66792–66806, 2019.

[45] Faryad Darabi Sahneh, Caterina Scoglio, and Piet Van Mieghem. Generalized epidemic mean-field model for spreading processes over multilayer complex networks. *IEEE/ACM Transactions on Networking*, 21(5):1609–1620, 2013.

[46] Sifat Afroj Moon, Faryad Darabi Sahneh, and Caterina Scoglio. Group-based general epidemic modeling for spreading processes on networks: Groupgem. *IEEE Transactions on Network Science and Engineering*, 2020.

[47] Ilkka Hanski. Metapopulation dynamics. *Nature*, 396(6706):41–49, 1998.

[48] M. J. Keeling and P. Rohani. *Modeling Infectious Diseases in Humans and Animals*. Princeton University Press, 2011.

[49] Qian Zhang, Kaiyuan Sun, Matteo Chinazzi, Ana Pastore y Piontti, Natalie E Dean, Diana Patricia Rojas, Stefano Merler, Dina Mistry, Piero Poletti, Luca Rossi, et al. Spread of Zika virus in the Americas. *Proceedings of the National Academy of Sciences*, 114(22):E4334–E4343, 2017.

[50] Eric D'Ortenzio, Sophie Matheron, Xavier de Lamballerie, Bruno Hubert, Géraldine Piorkowski, Marianne Maquart, Diane Descamps, Florence Damond, Yazdan Yazdanpanah, and Isabelle Leparc-Goffart. Evidence of Sexual Transmission of Zika Virus. *New England Journal of Medicine*, 374(22):2195–2198, 2016.

[51] Didier Musso, Claudine Roche, Emilie Robin, Tuxuan Nhan, Anita Teissier, and Van-Mai Cao-Lormeau. Potential sexual transmission of Zika virus. *Emerging Infectious Diseases*, 21(2):359, 2015.

[52] Daozhou Gao, Yijun Lou, Daihai He, Travis C Porco, Yang Kuang, Gerardo Chowell, and Shigui Ruan. Prevention and Control of Zika as a Mosquito-Borne and Sexually Transmitted Disease: A Mathematical Modeling Analysis. *Scientific Reports*, 6:28070, 2016.

[53] FB Agusto, S Bewick, and WF Fagan. Mathematical model for Zika virus dynamics with sexual transmission route. *Ecological Complexity*, 29:61–81, 2017.

[54] Ondrej Maxian, Anna Neufeld, Emma J Talis, Lauren M Childs, and Julie C Blackwood. Zika virus dynamics: When does sexual transmission matter? *Epidemics*, 21: 48–55, 2017.

[55] CM Saad-Roy, Junling Ma, and P van den Driessche. The effect of sexual transmission on Zika virus dynamics. *Journal of mathematical biology*, pages 1–25, 2018.

[56] YA Terefe, H Gaff, M Kamga, and L van der Mescht. Mathematics of a model for Zika transmission dynamics. *Theory in Biosciences*, pages 1–10, 2018.

[57] Devika Sirohi, Zhenguo Chen, Lei Sun, Thomas Klose, Theodore C Pierson, Michael G Rossmann, and Richard J Kuhn. The 3.8 å resolution cryo-EM structure of Zika virus. *Science*, 352(6284):467–470, 2016.

[58] Veronica Sikka, Vijay Kumar Chattu, Raaj K Popli, Sagar C Galwankar, Dhanashree Kelkar, Stanley G Sawicki, Stanislaw P Stawicki, and Thomas J Papadimos. The emergence of Zika virus as a global health security threat: A review and a consensus statement of the INDUSEM Joint working Group (JWG). *Journal of Global Infectious Diseases*, 8(1):3, 2016.

[59] Jernej Mlakar, Misa Korva, Nataša Tul, Mara Popović, Mateja Poljšak-Prijatelj, Jerica Mraz, Marko Kolenc, Katarina Resman Rus, Tina Vesnaver Vipotnik, Vesna Fabjan Vodušek, et al. Zika Virus Associated with Microcephaly. *New England Journal of Medicine*, 2016(374):951–958, 2016.

[60] Robert W Malone, Jane Homan, Michael V Callahan, Jill Glasspool-Malone, Lambodhar Damodaran, Adriano De Bernardi Schneider, Rebecca Zimler, James Talton, Ronald R Cobb, Ivan Ruzic, et al. Zika Virus: Medical Countermeasure Development Challenges. *PLOS Neglected Tropical Diseases*, 10(3):e0004530, 2016.

[61] Zika Cases in the United States, February 2018. URL https://www.cdc.gov/zika/reporting/case-counts.html. [Accessed: 03-May-2018].

[62] Lauren A Castro, Spencer J Fox, Xi Chen, Kai Liu, Steven E Bellan, Nedialko B Dimitrov, Alison P Galvani, and Lauren Ancel Meyers. Assessing real-time Zika risk in the United States. *BMC Infectious Diseases*, 17(1):284, 2017.

[63] Adam J Kucharski, Sebastian Funk, Rosalind M Eggo, Henri-Pierre Mallet, W John Edmunds, and Eric J Nilles. Transmission Dynamics of Zika Virus in Island Populations: A Modelling Analysis of the 2013–14 French Polynesia Outbreak. *PLOS Neglected Tropical Diseases*, 10(5):e0004726, 2016.

[64] Chris J Kuhlman, Yihui Ren, Bryan Lewis, and James Schlitt. Hybrid Agent-based modeling of Zika in the United States. In *2017 Winter Simulation Conference (WSC)*, pages 1085–1096. IEEE, 2017.

[65] FB Agusto, S Bewick, and WF Fagan. Mathematical model of Zika virus with vertical transmission. *Infectious Disease Modelling*, 2017.

[66] Omomayowa Olawoyin and Christopher Kribs. Effects of multiple transmission pathways on Zika dynamics. *Infectious Disease Modelling*, 2018.

[67] Seyed M Moghadas, Affan Shoukat, Aquino L Espindola, Rafael S Pereira, Fatima Abdirizak, Marek Laskowski, Cecile Viboud, and Gerardo Chowell. Asymptomatic Transmission and the Dynamics of Zika Infection. *Scientific Reports*, 7(1):5829, 2017.

[68] Andrea Pugliese, Abba B Gumel, Fabio A Milner, and Jorge X Velasco-Hernandez. Sex-biased prevalence in infections with heterosexual, direct, and vector-mediated transmission: a theoretical analysis. *Mathematical Biosciences & Engineering*, 15(1):125–140, 2018.

[69] Sourav Kumar Sasmal, Indrajit Ghosh, Amit Huppert, and Joydev Chattopadhyay. Modeling the Spread of Zika Virus in a Stage-Structured Population: Effect of Sexual Transmission. *Bulletin of mathematical biology*, pages 1–30, 2018.

[70] Antoine Allard, Benjamin M Althouse, Laurent Hébert-Dufresne, and Samuel V Scarpino. The risk of sustained sexual transmission of Zika is underestimated. *PLOS pathogens*, 13(9):e1006633, 2017.

[71] Jean Marie Turmel, Pierre Abgueguen, Bruno Hubert, Yves Marie Vandamme, Marianne Maquart, Hélène Le Guillou-Guillemette, and Isabelle Leparc-Goffart. Late sexual transmission of Zika virus related to persistence in the semen. *The Lancet*, 387(10037): 2501, 2016.

[72] Tom William Smith. *American Sexual Behavior: Trends, Socio-Demographic Differences, and Risk Behavior*. National Opinion Research Center Chicago, 1998.

[73] Faryad Darabi Sahneh, Caterina Scoglio, and Fahmida N Chowdhury. Effect of coupling on the epidemic threshold in interconnected complex networks: A spectral analysis. In *American Control Conference (ACC), 2013*, pages 2307–2312. IEEE, 2013.

[74] Marian Boguná, Luis F Lafuerza, Raúl Toral, and M Ángeles Serrano. Simulating non-Markovian stochastic processes. *Physical Review E*, 90(4):042108, 2014.

[75] Faryad Darabi Sahneh, Aram Vajdi, Heman Shakeri, Futing Fan, and Caterina Scoglio. GEMFsim: a stochastic simulator for the generalized epidemic modeling framework. *Journal of Computational Science*, 22:36–44, 2017.

[76] Ronald H Gray, Maria J Wawer, Ron Brookmeyer, Nelson K Sewankambo, David Serwadda, Fred Wabwire-Mangen, Tom Lutalo, Xianbin Li, Thomas C Quinn, et al. Probability of HIV-1 transmission per coital act in monogamous, heterosexual, HIV-1-discordant couples in Rakai, Uganda. *The Lancet*, 357(9263):1149–1153, 2001.

[77] Patrick K Mitchell, Luis Mier-y Teran-Romero, Brad J Biggerstaff, Mark J Delorey, Maite Aubry, Van-Mai Cao-Lormeau, Matthew J Lozier, Simon Cauchemez, and Michael A Johansson. Reassessing Serosurvey-Based Estimates of the Zika Symptomatic Proportion. *American journal of epidemiology*, 2018.

[78] Zika has been sexually transmitted in Texas, CDC confirms, February 2016. URL https://www.cnn.com/2016/02/02/health/zika-virus-sexual-contact-texas/index.html. [Accessed: 01-November-2018].

[79] Cyril Caminade, Joanne Turner, Soeren Metelmann, Jenny C Hesson, Marcus SC Blagrove, Tom Solomon, Andrew P Morse, and Matthew Baylis. Global risk model for vector-borne transmission of Zika virus reveals the role of El Niño 2015. *Proceedings of the National Academy of Sciences*, 114(1):119–124, 2017.

[80] MH Reiskind and LP Lounibos. Spatial and temporal patterns of abundance of *Aedes aegypti* L.(*Stegomyia aegypti*) and *Aedes albopictus* (Skuse)[*Stegomyia albopic-*

tus (Skuse)] in southern Florida. *Medical and Veterinary Entomology*, 27(4):421–429, 2013.

[81] Giovanni Marini, Giorgio Guzzetta, Roberto Rosà, and Stefano Merler. First outbreak of Zika virus in the continental United States: a modelling analysis. *Eurosurveillance*, 22(37), 2017.

[82] Andrew J Monaghan, Cory W Morin, Daniel F Steinhoff, Olga Wilhelmi, Mary Hayden, Dale A Quattrochi, Michael Reiskind, Alun L Lloyd, Kirk Smith, Chris A Schmidt, et al. On the Seasonal Occurrence and Abundance of the Zika Virus Vector Mosquito *Aedes aegypti* in the Contiguous United States. *PLOS Currents*, 8, 2016.

[83] Sifat A Moon, Lee W Cohnstaedt, D Scott McVey, and Caterina M Scoglio. A spatio-temporal individual-based network framework for West Nile virus in the USA: spreading pattern of West Nile virus. *PLOS Computational Biology*, 15(3):e1006875, 2019.

[84] World Development Indicators: Population dynamics, 2017. URL http://wdi.worldbank.org/table/2.1. [Accessed: 16-October-2018].

[85] Katy Robinson, Ted Cohen, and Caroline Colijn. The dynamics of sexual contact networks: Effects on disease spread and control. *Theoretical Population Biology*, 81 (2):89–96, 2012.

[86] Naoki Masuda and Luis EC Rocha. A Gillespie Algorithm for Non-Markovian Stochastic Processes. *SIAM Review*, 60(1):95–115, 2018.

[87] GEMFsim vector-borne, March 2019. URL https://www.ece.k-state.edu/netse/software/.

[88] Ruth E Baker, Jose-Maria Peña, Jayaratnam Jayamohan, and Antoine Jérusalem. Mechanistic models versus machine learning, a fight worth fighting for the biological community? *Biology letters*, 14(5):20170660, 2018.

[89] Samir Bhatt, Peter W Gething, Oliver J Brady, Jane P Messina, Andrew W Farlow, Catherine L Moyes, John M Drake, John S Brownstein, Anne G Hoen, Osman Sankoh, et al. The global distribution and burden of dengue. *Nature*, 496(7446):504–507, 2013.

[90] Gregory A Roth, Degu Abate, Kalkidan Hassen Abate, Solomon M Abay, Cristiana Abbafati, Nooshin Abbasi, Hedayat Abbastabar, Foad Abd-Allah, Jemal Abdela, Ahmed Abdelalim, et al. Global, regional, and national age-sex-specific mortality for 282 causes of death in 195 countries and territories, 1980–2017: a systematic analysis for the global burden of disease study 2017. *The Lancet*, 392(10159):1736–1788, 2018.

[91] Ruiyun Li, Lei Xu, Ottar N Bjørnstad, Keke Liu, Tie Song, Aifang Chen, Bing Xu, Qiyong Liu, and Nils C Stenseth. Climate-driven variation in mosquito density predicts the spatiotemporal dynamics of dengue. *Proceedings of the National Academy of Sciences*, 116(9):3624–3629, 2019.

[92] Douglas O Fuller, A Troyo, and John C Beier. El Nino Southern Oscillation and vegetation dynamics as predictors of dengue fever cases in Costa Rica. *Environmental Research Letters*, 4(1):014011, 2009.

[93] Pei-Chih Wu, Jinn-Guey Lay, How-Ran Guo, Chuan-Yao Lin, Shih-Chun Lung, and Huey-Jen Su. Higher temperature and urbanization affect the spatial patterns of dengue fever transmission in subtropical Taiwan. *Science of the total Environment*, 407(7):2224–2233, 2009.

[94] S Wongkoon, M Jaroensutasinee, and K Jaroensutasinee. Distribution, seasonal variation & dengue transmission prediction in Sisaket, Thailand. *The Indian journal of medical research*, 138(3):347, 2013.

[95] Mammen P Mammen Jr, Chusak Pimgate, Constantianus JM Koenraadt, Alan L Rothman, Jared Aldstadt, Ananda Nisalak, Richard G Jarman, James W Jones, Anon Srikiatkhachorn, Charity Ann Ypil-Butac, et al. Spatial and temporal clustering of dengue virus transmission in Thai villages. *PLOS Med*, 5(11):e205, 2008.

[96] Aurora González-Vidal, Victoria Moreno-Cano, Fernando Terroso-Sáenz, and Antonio F Skarmeta. Towards energy efficiency smart buildings models based on intelligent data analytics. *Procedia Computer Science*, 83:994–999, 2016.

[97] M Victoria Moreno, Fernando Terroso-Sáenz, Aurora González-Vidal, Mercedes Valdés-Vela, Antonio F Skarmeta, Miguel A Zamora, and Victor Chang. Applicability of big data techniques to smart cities deployments. *IEEE Transactions on Industrial Informatics*, 13(2):800–809, 2016.

[98] Youqiang Sun, Jiuyong Li, Jixue Liu, Christopher Chow, Bingyu Sun, and Rujing Wang. Using causal discovery for feature selection in multivariate numerical time series. *Machine Learning*, 101(1-3):377–395, 2015.

[99] Youssef Hmamouche, Alain Casali, and Lotfi Lakhal. A causality based feature selection approach for multivariate time series forecasting. In *DBKDA 2017, The Ninth International Conference on Advances in Databases, Knowledge, and Data Applications*, 2017.

[100] Deon Garrett, David A Peterson, Charles W Anderson, and Michael H Thaut. Comparison of linear, nonlinear, and feature selection methods for EEG signal classification. *IEEE Transactions on neural systems and rehabilitation engineering*, 11(2):141–144, 2003.

[101] Sven F Crone and Nikolaos Kourentzes. Feature selection for time series prediction–a combined filter and wrapper approach for neural networks. *Neurocomputing*, 73(10-12): 1923–1936, 2010.

[102] Lkhagvadorj Munkhdalai, Tsendsuren Munkhdalai, Kwang Ho Park, Tsatsral Amarbayasgalan, Erdenebileg Erdenebaatar, Hyun Woo Park, and Keun Ho Ryu. An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series. *IEEE Access*, 7:99099–99114, 2019.

[103] Lei Yu and Huan Liu. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)*, pages 856–863, 2003.

[104] Mark Andrew Hall. Correlation-based feature selection for machine learning. *PhD Dissertation*, 1999.

[105] Dianbo Liu, Leonardo Clemente, Canelle Poirier, Xiyu Ding, Matteo Chinazzi, Jessica Davis, Alessandro Vespignani, and Mauricio Santillana. Real-time forecasting of the COVID-19 outbreak in Chinese provinces: Machine learning approach using novel digital data and estimates from mechanistic models. *J Med Internet Res*, 22(8):e20285, 2020.

[106] Elisa Mussumeci and Flávio Codeço Coelho. Large-scale multivariate forecasting models for dengue-LSTM versus random forest regression. *Spatial and Spatio-temporal Epidemiology*, 35:100372, 2020.

[107] Désirée Schoenherr, Jane Paulick, Bernhard M Strauss, Anne-Katharina Deisenhofer, Brian Schwartz, Julian A Rubel, Wolfgang Lutz, Ulrich Stangier, and Uwe Altmann. Identification of movement synchrony: Validation of windowed cross-lagged correlation and-regression with peak-picking algorithm. *PLOS ONE*, 14(2):e0211494, 2019.

[108] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media, 2009.

[109] Yvan Saeys, Inaki Inza, and Pedro Larranaga. A review of feature selection techniques in bioinformatics. *bioinformatics*, 23(19):2507–2517, 2007.

[110] Fernando Jimenez, Jose Palma, Gracia Sanchez, David Marin, MD Francisco Palacios, and MD Lucía López. Feature selection based multivariate time series forecasting: An application to antibiotic resistance outbreaks prediction. *Artificial Intelligence in Medicine*, 104:101818, 2020.

[111] George EP Box, Gwilym M Jenkins, and Gregory C Reinsel. *Time series analysis: forecasting and control*, volume 734. John Wiley & Sons, 2011.

[112] Adhistya Erna Permanasari, Indriana Hidayah, and Isna Alfi Bustoni. Sarima (seasonal arima) implementation on time series to forecast the number of malaria incidence. In *2013 International Conference on Information Technology and Electrical Engineering (ICITEE)*, pages 203–207. IEEE, 2013.

[113] Wei-Chiang Hong. Application of seasonal SVR with chaotic immune algorithm in traffic flow forecasting. *Neural Computing and Applications*, 21(3):583–593, 2012.

[114] Thomas W Scott, Esther Chow, Daniel Strickman, Pattamaporn Kittayapong, Robert A Wirtz, Leslie H Lorenz, and John D Edman. Blood-feeding patterns of *Aedes aegypti* (Diptera: *Culicidae*) collected in a rural Thai village. *Journal of medical entomology*, 30(5):922–927, 1993.

[115] Shubham Shrivastava, Divya Tiraki, Arundhati Diwan, Sanjay K Lalwani, Meera Modak, Akhilesh Chandra Mishra, and Vidya A Arankalle. Co-circulation of all the four dengue virus serotypes and detection of a novel clade of DENV-4 (genotype I) virus in Pune, India during 2016 season. *PLOS ONE*, 13(2):e0192672, 2018.

[116] John H Huber, Marissa L Childs, Jamie M Caldwell, and Erin A Mordecai. Seasonal temperature variation influences climate suitability for dengue, chikungunya, and Zika transmission. *PLOS neglected tropical diseases*, 12(5):e0006451, 2018.

[117] Amy Wesolowski, Taimur Qureshi, Maciej F Boni, Pål Roe Sundsøy, Michael A Johansson, Syed Basit Rasheed, Kenth Engø-Monsen, and Caroline O Buckee. Impact of human mobility on the emergence of dengue epidemics in Pakistan. *Proceedings of the National Academy of Sciences*, 112(38):11887–11892, 2015.

[118] Tanujit Chakraborty, Swarup Chattopadhyay, and Indrajit Ghosh. Forecasting dengue epidemics using a hybrid methodology. *Physica A: Statistical Mechanics and its Applications*, 527:121266, 2019.

[119] Nikolay Laptev, Jason Yosinski, Li Erran Li, and Slawek Smyl. Time-series extreme event forecasting with neural networks at Uber. In *International Conference on Machine Learning*, volume 34, pages 1–5, 2017.

[120] Yuxiu Hua, Zhifeng Zhao, Rongpeng Li, Xianfu Chen, Zhiming Liu, and Honggang Zhang. Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, 57(6):114–119, 2019.

[121] Lingxue Zhu and Nikolay Laptev. Deep and confident prediction for time series at Uber. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 103–110. IEEE, 2017.

[122] Jiangyan Gu, Lizhong Liang, Hongquan Song, Yunfeng Kong, Rui Ma, Yane Hou, Jinyu Zhao, Junjie Liu, Nan He, and Yang Zhang. A method for hand-foot-mouth disease prediction using GeoDetector and LSTM model in Guangxi, China. *Scientific Reports*, 9(1):1–10, 2019.

[123] Farah Shahid, Aneela Zameer, and Muhammad Muneeb. Predictions for COVID-19 with deep learning models of LSTM, GRU and Bi-LSTM. *Chaos, Solitons & Fractals*, page 110212, 2020.

[124] Jiucheng Xu, Keqiang Xu, Zhichao Li, Fengxia Meng, Taotian Tu, Lei Xu, and Qiyong Liu. Forecast of dengue cases in 20 chinese cities based on the deep learning method. *International journal of environmental research and public health*, 17(2):453, 2020.

[125] Samuel Rickard Christophers. *Aedes aegypti: the yellow fever mosquito*. CUP Archive, 1960.

[126] Jannelle Couret, Ellen Dotson, and Mark Q Benedict. Temperature, larval diet, and density effects on development rate and survival of *Aedes aegypti* (Diptera: Culicidae). *PLOS ONE*, 9(2):e87468, 2014.

[127] Hélène Delatte, Geoffrey Gimonneau, Aurélie Triboire, and Didier Fontenille. Influence of temperature on immature development, survival, longevity, fecundity, and

gonotrophic cycles of *Aedes albopictus*, vector of chikungunya and dengue in the Indian Ocean. *Journal of medical entomology*, 46(1):33–41, 2009.

[128] Wayne A Rowley and Charles L Graham. The effect of temperature and relative humidity on the flight performance of female *Aedes aegypti*. *Journal of Insect Physiology*, 14(9):1251–1257, 1968.

[129] Lauren B Carrington, Stephanie N Seifert, Neil H Willits, Louis Lambrechts, and Thomas W Scott. Large diurnal temperature fluctuations negatively influence *Aedes aegypti* (Diptera: Culicidae) life-history traits. *Journal of medical entomology*, 50(1): 43–51, 2013.

[130] Douglas M Watts, Donald S Burke, Bruce A Harrison, Richard E Whitmire, and Ananda Nisalak. Effect of temperature on the vector efficiency of *Aedes aegypti* for dengue 2 virus. *The American journal of tropical medicine and hygiene*, 36(1):143–152, 1987.

[131] Michael A Johansson, Francesca Dominici, and Gregory E Glass. Local and global effects of climate on dengue transmission in Puerto Rico. *PLOS Negl Trop Dis*, 3(2): e382, 2009.

[132] INFO Dengue. Info dengue surveillance project, 2021. URL https://info.dengue.mat.br/.

[133] Lutz Prechelt. Early stopping-but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

[134] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI '16)*, pages 265–283, 2016.

[135] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[136] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[137] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[138] Cross correlation, 2021. URL https://en.wikipedia.org/wiki/Cross-correlation.

[139] Charles FF Karney. Algorithms for geodesics. *Journal of Geodesy*, 87(1):43–55, 2013.

[140] Instituto Brasileiro de Geografia e Estatística, 2021. URL https://www.ibge.gov.br/.

[141] Python-Visualization. Folium, 2021. URL https://python-visualization.github.io/folium/.

[142] OpenStreetMap Foundation. OpenStreetMap, 2021. URL https://www.openstreetmap.org.

[143] CartoDB Inc. CartoDB Positron, 2021. URL https://carto.com/.

[144] James M Omernik. Perspectives on the nature and definition of ecological regions. *Environmental Management*, 34(1):S27–S38, 2004.

[145] David M Olson and Eric Dinerstein. The global 200: Priority ecoregions for global conservation. *Annals of the Missouri Botanical garden*, pages 199–224, 2002.

[146] Christopher L Burdett, Brian R Kraus, Sarah J Garza, Ryan S Miller, and Kathe E Bjork. Simulating the distribution of individual livestock farms and their populations in the United States: An example using domestic swine (*Sus scrofa domesticus*) farms. *PLOS ONE*, 10(11):e0140338, 2015.

[147] Pablo Valdes-Donoso, Kimberly VanderWaal, Lovell S Jarvis, Spencer R Wayne, and Andres M Perez. Using machine learning to predict swine movements within a regional

program to improve control of infectious diseases in the US. *Frontiers in Veterinary Science*, 4:2, 2017.

[148] Sifat A Moon, Tanvir Ferdousi, Adrian Self, and Caterina M Scoglio. Estimation of swine movement network at farm level in the US from the census of agriculture data. *Scientific Reports*, 9(1):6237, 2019.

[149] Mark EJ Newman, Steven H Strogatz, and Duncan J Watts. Random graphs with arbitrary degree distributions and their applications. *Physical Review E*, 64(2):026118, 2001.

[150] Ron Milo, Nadav Kashtan, Shalev Itzkovitz, Mark EJ Newman, and Uri Alon. On the uniform generation of random graphs with prescribed degree sequences. *arXiv preprint cond-mat/0312028*, 2003.

[151] A Ramachandra Rao, Rabindranath Jana, and Suraj Bandyopadhyay. A Markov chain Monte Carlo method for generating random (0, 1)-matrices with given marginals. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 225–242, 1996.

[152] John M Roberts Jr. Simple methods for simulating sociomatrices with given marginal totals. *Social Networks*, 22(3):273–283, 2000.

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

[154] Tom Britton, Maria Deijfen, and Anders Martin-Löf. Generating simple random graphs with prescribed degree distribution. *Journal of Statistical Physics*, 124(6):1377–1397, 2006.

[155] Hartmut HK Lentz, Andreas Koher, Philipp Hövel, Jörn Gethmann, Carola Sauter-Louis, Thomas Selhorst, and Franz J Conraths. Disease spread through animal movements: a static and temporal network analysis of pig trade in Germany. *PLOS ONE*, 11(5):e0155196, 2016.

[156] Romualdo Pastor-Satorras and Alessandro Vespignani. Immunization of complex networks. *Physical Review E*, 65(3):036104, 2002.

[157] Caterina Scoglio, Walter Schumm, Phillip Schumm, Todd Easton, Sohini Roy Chowdhury, Ali Sydney, and Mina Youssef. Efficient mitigation strategies for epidemics in rural regions. *PLOS ONE*, 5(7):e11569, 2010.

[158] S Costard, L Mur, J Lubroth, JM Sanchez-Vizcaino, and DU Pfeiffer. Epidemiology of African swine fever virus. *Virus Research*, 173(1):191–197, 2013.

[159] Tsvetelia Tsolova. Bulgaria reports its first outbreak of African swine fever, August 2018. URL https://reut.rs/2NzcCM9. [Accessed: 25-Sep-2018].

[160] Alistair Driver. 4,000 pigs to be culled to control African swine fever in Belgium, September 2018. URL http://www.pig-world.co.uk/news/4000-pigs-to-be-culled-to-control-african-swine-fever-in-belgium.html. [Accessed: 25-Sep-2018].

[161] ASF situation in Asia update, January 2019. URL http://www.fao.org/ag/againfo/programmes/en/empres/ASF/situation_update.html. [Accessed: 29-Jan-2019].

[162] Hallie Gu and Dominique Patton. China's top pig farmers see sharp fall in profits amid disease epidemic, January 2019. URL https://reut.rs/2Vz8TCD. [Accessed: 8-Jan-2019].

[163] Tao Wang, Yuan Sun, and Hua-Ji Qiu. African swine fever: an unprecedented disaster and challenge to China. *Infectious Diseases of Poverty*, 7(1):111, 2018.

[164] Diana María Herrera-Ibatá, Beatriz Martínez-López, Darla Quijada, Kenneth Burton, and Lina Mur. Quantitative approach for the risk assessment of African swine fever and Classical swine fever introduction into the United States through legal imports of pigs and swine products. *PLOS ONE*, 12(8):e0182850, 2017.

[165] Mike B Barongo, Richard P Bishop, Eric M Fèvre, Darryn L Knobel, and Amos Ssematimba. A mathematical model that simulates control options for African swine fever virus (ASFV). *PLOS ONE*, 11(7):e0158658, 2016.

[166] Tariq Halasa, Anette Boklund, Anette Bøtner, Nils Toft, and Hans-Hermann Thulke. Simulation of spread of African swine fever, including the effects of residues from dead animals. *Frontiers in veterinary science*, 3:6, 2016.

[167] C Guinat, S Gubbins, T Vergne, JL Gonzales, L Dixon, and DU Pfeiffer. Experimental pig-to-pig transmission dynamics for African swine fever virus, Georgia 2007/1 strain–CORRIGENDUM. *Epidemiology & Infection*, 144(16):3564–3566, 2016.

[168] VM Gulenkin, FI Korennoy, AK Karaulov, and SA Dudnikov. Cartographical analysis of African swine fever outbreaks in the territory of the Russian Federation and computer modeling of the basic reproduction ratio. *Preventive Veterinary Medicine*, 102 (3):167–174, 2011.

[169] Mike B Barongo, Karl Ståhl, Bernard Bett, Richard P Bishop, Eric M Fèvre, Tony Aliro, Edward Okoth, Charles Masembe, Darryn Knobel, and Amos Ssematimba. Estimating the basic reproductive number (R0) for African swine fever virus (ASFV) transmission between pig herds in Uganda. *PLOS ONE*, 10(5):e0125842, 2015.

[170] C Guinat, T Porphyre, A Gogin, L Dixon, DU Pfeiffer, and S Gubbins. Inferring within-herd transmission parameters for African swine fever virus using mortality data from outbreaks in the Russian Federation. *Transboundary and Emerging Diseases*, 65 (2):e264–e271, 2018.

[171] Ben Hu, Jose L Gonzales, and Simon Gubbins. Bayesian inference of epidemiological parameters from transmission experiments. *Scientific Reports*, 7(1):16774, 2017.

[172] Roxann Motroni, John Neilan, Max Rasmussen, Chungwon Chung, Michael Puckette, David Brake, and Barbara Kamicker. Development of next-generation vaccines and diagnostics for transboundary animal disease preparedness, 2017. URL

http://www.oie.int/eng/BIOTHREAT2017/posters/22_MOTRONI-poster.pdf. [Accessed: 30-Jan-2019].

[173] 2012 Census of Agriculture, May 2014. URL https://www.nass.usda.gov/Publications/AgCensus/2012/Full_Report/Volume_1,_Chapter_2_US_State_Level/st99_2_012_012.pdf. [Accessed: 15-Jun-2018].

[174] Spencer R Wayne. Assessment of the demographics and network structure of swine populations in relation to regional disease transmission and control. 2011. URL http://hdl.handle.net/11299/112785.

[175] Animal Disease Traceability, 2019. URL https://www.aphis.usda.gov.

[176] What is CattleTrace, 2018. URL https://www.cattletrace.org.

[177] Qihui Yang, Don Gruenbacher, Jessica L Heier Stamm, Gary L Brase, Scott A DeLoach, David E Amrine, and Caterina Scoglio. Developing an agent-based model to simulate the beef cattle production and transportation in southwest Kansas. *Physica A: Statistical Mechanics and its Applications*, 526:120856, 2019.

[178] A Banerjee. Integrating Blockchain with ERP for a transparent supply chain, 2018.

[179] TradeLens, 2020. URL https://www.tradelens.com/.

[180] Provenance tuna tracking, 2019. URL https://www.provenance.org/tracking-tuna-on-the-blockchain.

[181] Qinghua Lu and Xiwei Xu. Adaptable blockchain-based systems: A case study for product traceability. *IEEE Software*, 34(6):21–27, 2017.

[182] Boris Düdder and Omri Ross. Timber tracking: Reducing complexity of due diligence by using blockchain technology. *Available at SSRN 3015219*, 2017.

[183] Sara Saberi, Mahtab Kouhizadeh, Joseph Sarkis, and Lejia Shen. Blockchain technology and its relationships to sustainable supply chain management. *International Journal of Production Research*, 57(7):2117–2135, 2019.

[184] Roberto Casado-Vara, Javier Prieto, Fernando De la Prieta, and Juan M Corchado. How blockchain improves the supply chain: Case study alimentary supply chain. *Procedia computer science*, 134:393–398, 2018.

[185] Matteo Montecchi, Kirk Plangger, and Michael Etter. It's real, trust me! establishing supply chain provenance using blockchain. *Business Horizons*, 62(3):283–293, 2019.

[186] Kaijun Leng, Ya Bi, Linbo Jing, Han-Chi Fu, and Inneke Van Nieuwenhuyse. Research on agricultural supply chain system with double chain architecture based on blockchain technology. *Future Generation Computer Systems*, 86:641–649, 2018.

[187] Henry M Kim and Marek Laskowski. Toward an ontology-driven blockchain design for supply-chain provenance. *Intelligent Systems in Accounting, Finance and Management*, 25(1):18–27, 2018.

[188] Feng Tian. A supply chain traceability system for food safety based on HACCP, blockchain & internet of things. In *2017 International Conference on Service Systems and Service Management*, pages 1–6. IEEE, 2017.

[189] Martin Westerkamp, Friedhelm Victor, and Axel Küpper. Blockchain-based supply chain traceability: Token recipes model manufacturing processes. *arXiv preprint arXiv:1810.09843*, 2018.

[190] Xueping Liang, Sachin Shetty, Deepak Tosh, Charles Kamhoua, Kevin Kwiat, and Laurent Njilla. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *Proceedings of the 17th IEEE/ACM international symposium on cluster, cloud and grid computing*, pages 468–477. IEEE Press, 2017.

[191] Kentaroh Toyoda, P Takis Mathiopoulos, Iwao Sasase, and Tomoaki Ohtsuki. A novel blockchain-based product ownership management system (POMS) for anti-counterfeits in the post supply chain. *IEEE Access*, 5:17465–17477, 2017.

[192] Hokey Min. Blockchain technology for enhancing supply chain resilience. *Business Horizons*, 62(1):35–45, 2019.

[193] Yong Yuan and Fei-Yue Wang. Towards blockchain-based intelligent transportation systems. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 2663–2668. IEEE, 2016.

[194] Ron Miller. Walmart is betting on the blockchain to improve food safety, 2018. URL https://techcrunch.com/2018/09/24/walmart-is-betting-on-the-blockchain-to-improve-food-safety/.

[195] Wyoming Beefchain, 2019. URL https://beefchain.com/.

[196] Zhaojun Lu, Wenchao Liu, Qian Wang, Gang Qu, and Zhenglin Liu. A privacy-preserving trust model based on blockchain for VANETs. *IEEE Access*, 6:45655–45664, 2018.

[197] Haya R Hasan and Khaled Salah. Blockchain-based proof of delivery of physical assets with single and multiple transporters. *IEEE Access*, 6:46781–46793, 2018.

[198] Khaled Salah, Nishara Nizamuddin, Raja Jayaraman, and Mohammad Omar. Blockchain-based soybean traceability in agricultural supply chain. *IEEE Access*, 7: 73295–73305, 2019.

[199] Qijun Lin, Huaizhen Wang, Xiaofu Pei, and Junyu Wang. Food safety traceability system based on blockchain and EPCIS. *IEEE Access*, 7:20698–20707, 2019.

[200] Go Ethereum, 2020. URL https://geth.ethereum.org/.

[201] Parity Ethereum, 2020. URL https://www.parity.io/ethereum/.

[202] Remix, 2020. URL http://remix.ethereum.org/.

[203] Solidity, 2020. URL https://solidity.readthedocs.io.

[204] Web3, 2020. URL https://web3js.readthedocs.io.

Appendix A

Network Generation Algorithms

The basic outline of network generation is described in Algorithm 1.

input : $N_F, PIG\_DAT, SF, P, F\_DIST, M\_MIX, K\_DAT, SHP\_DAT$
output: G_F, G_P
// Assigns different operation types to individual farms uniformly at random.
1 $F\_TYPE \leftarrow F\_TYPE\_GEN(F\_DIST, N_F)$;
// Generates a directed farm graph with weighted (shipment rate) links.
2 $G_F \leftarrow F\_GRAPH\_GEN(M\_MIX, F\_TYPE, N_F, K\_DAT, SHP\_DAT)$;
// Assigns pigs to different operations based on data obtained from $NASS$. Scales the population by a factor SF.
3 $PIG\_LIST \leftarrow PIG\_ALLOT(PIG\_DAT, N_F, SF)$;
// Generates a pig level graph.
4 $G_P \leftarrow P\_GRAPH\_GEN(G_F, N_F, PIG\_LIST, P)$;

Algorithm 1: Basic outline of graph generation

In the above box, N_F is the number of farms, PIG_DAT is the pig distribution data given in Table 4.4, SF is the scaling factor used to scale the pig level graph in order to make it computationally feasible in our model, P is the probability of within farm contacts between pigs, F_DIST is the farm type distribution shown in Table 4.3, M_MIX is the mixing matrix shown in Table 4.1, K_DAT contains the degree centrality data from Table 4.2, and SHP_DAT contains the mean and median shipment data. The farm level and pig level graphs are represented by G_F and G_P, respectively. We describe the F_GRAPH_GEN function in more detail in Algorithm 2.

input : $M\_MIX, F\_TYPE, N_F, K\_DAT, SHP\_DAT$
output: G_F
// Allocates in and out-degrees based on production types.
$DEG\_ALLOC \leftarrow DEG\_GEN(F\_TYPE, K\_DAT)$;
for $SRC\_NODE \leftarrow 1$ **to** N_F **do**
  // Get the allocated (max) out-degree of source node.
  $MAX\_KOUT = DEG\_ALLOC[SRC\_NODE, OUTDEG]$;
  for $k \leftarrow 1$ **to** MAX_KOUT **do**
    // Randomly sample the destination type from the mixing matrix distribution.
    $DST\_TYPE \sim M\_MIX[F\_TYPE[SRC\_NODE]]$;
    // Find all the nodes of the given type.
    $DST\_NODES \leftarrow \{n : n \in [1, N_F]$ and $F\_TYPE(n) = DST\_TYPE\}$;
    // Pick the destination node which has the largest in-degree gap (max in-degree - current in-degree) to cover up.
    $DST\_NODE \leftarrow n \in DST\_NODES$ that maximizes $(KIN(n, MAX) - KIN(n, CUR))$;
    // generate shipment data from log-normal distribution with mean and median shipment values given in SHP_DAT.
    $SHP\_RATE \sim Lognormal(SHP\_DAT)$;
    // Create the link with generated shipment rate as weight.
    $G_F \leftarrow G_F + EDGE(SRC\_NODE, DST\_NODE, SHP\_RATE)$;
  end
end

Algorithm 2: F_GRAPH_GEN

We also describe another key component of the graph generator, P_GRAPH_GEN in Algorithm 3, which generates the pig level network.

input : G_F, N_F, PIG_LIST, P
output: G_P

for $f \leftarrow 1$ **to** N_F **do**
    // List of pigs in the farm f.
    $PIG\_NODES \leftarrow PIG\_LIST(f)$;
    // Generate Erdos-Renyi graph for list of pig nodes with edge probability P.
    $EDGES = ERDOS\_RENYI(PIG\_NODES, P)$;
    // Add the generated edges to the pig level graph.
    $G_P \leftarrow G_P + EDGES$;
end
// Normalize the edge weights to be used as probabilities.
$G_F^{NORM} \leftarrow G_F / MAX(G_F)$;
for *each* $(F1, F2)$ *pair where* $F1, F2 \in G_F$ *and* $F1! = F2$ **do**
    // Probability of movement from farm F1 to any other farms.
    $P_{FROM} \leftarrow G_F^{NORM}(F1, ALL)$;
    // Probability of movement from any farm to farm F2.
    $P_{TO} \leftarrow G_F^{NORM}(ALL, F2)$;
    for *each* $P1 \in PIG\_LIST(F1)$ **do**
        for *each* $P2 \in PIG\_LIST(F2)$ **do**
            $r \sim Uniform(0, 1)$;
            if $r \leq P_{FROM} \times P_{TO}$ **then**
                $G_P \leftarrow G_P + EDGE(P1, P2)$;
        end
    end
end

Algorithm 3: P_GRAPH_GEN

Appendix B

Smart Contracts

Smart contracts are immutable pieces of code that run in the blockchain system. Our system has five smart contracts, each with different objectives. In each of those, we define variables, objects, and methods to enable different management capabilities. The methods are designed to enforce policies regarding how to handle different aspects of the system, including permissions, data access, etc. These contracts are loaded into the system and configured by an administrator during the system initialization process. In response to different method calls, we use Ethereum event logs to understand different outcomes. We use the following HTTP style response codes in these logs: 200 (OK/success), 201 (created), 202 (accepted), 400 (bad request), 403 (forbidden), and 404 (not found).

B.1 Profile Manager

Every entity that accesses the system is managed and controlled by the Profile Manager. As shown in Figure B.1, we define a data structure `User`, which has the following components:

- `timeAdded` is the unix timestamp when the user profile is created.
- `accessLevel` is the access classification for the user. There are 3 distinct access levels: `restricted (0)`, `viewer (1)`, and `manager (2)`.

Each entry of the `User` data structure is mapped through a 160 bit Ethereum `address` which is the user id. The `ProfileManager` contract also contains several state variables:

- `userCount` is the number of registered users
- `admin` is the address of the system administrator.
- `userMap` is the mapping data structure that maps user id to `User` object.

Initially, the creator of the smart contract is automatically added as the admin by the constructor function. Only the admin can register new user profiles, change permissions, and change admins. Most of the methods (functions) contain code snippets that validate the entity that invoked the call. The methods in this contract are listed below,

- `registerUser()` adds a new profile to the system, creates a `User` object and stores it using the `userMap` mapping. Only an admin can call this function.
- `getUserInfo()` returns information about a user based on the address given in the argument. The requesting entity must be an admin or the user itself.
- `updateUser()` updates the user profile. An admin can call this function to change access permissions.
- `checkAccessLevel()` returns the access level of a user given its address in the argument.
- `checkAdminAccess()` checks whether the current contract method calling entity has the admin level access (i.e., is the admin) or not.
- `changeAdmin()` assigns a new admin given the `address` in the argument. Only the current admin can call this function. Once called, if the contract transaction is confirmed, the current admin will lose its status as admin.

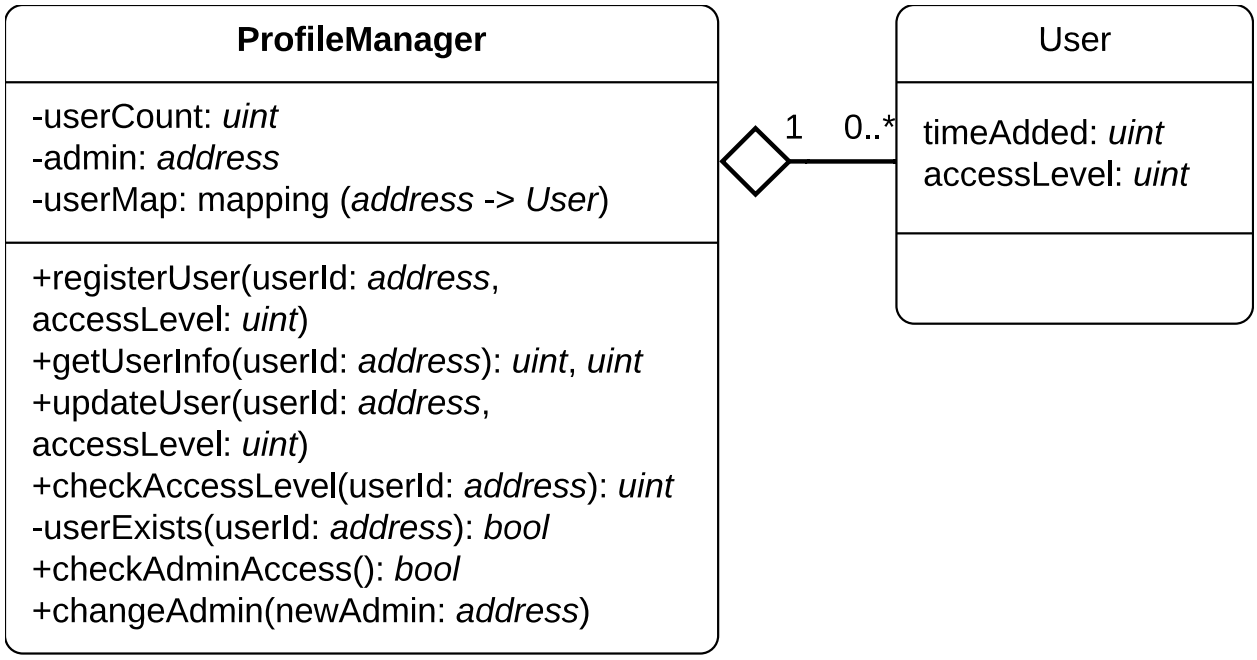


Figure B.1: *The* `ProfileManager` *contract and its associated* `User` *data structure. There can be multiple user profiles in the system, all of which must be registered via the methods of this contract*

B.2 Farm Manager

The Farm Manager contract regulates the contents of the farm databases (marked as 'Local Databases' in Figure 5.2) and provides useful farm-related operational functionalities. The contract and its associated data structures are shown in Figure B.2. To store cryptographic proof of farm database contents, we define the `Farm` data structure, which has the following components:

- `ownerId` is the 160 bit Ethereum `address` of the farm owners profile.
- `animalCount` is the number of animals currently registered in the farm.
- `farmHash` is the most recent cryptographic hash generated from the local database of the farm.
- `timeUpdated` is the unix timestamp when the most recent `farmHash` was stored in the chain.

In addition to that, we define the `Animal` data structure with the following components to store cryptographic proofs of individual animal related data:

- `currentFarm` is the 160 bit Ethereum `address` of the farm that owns the animal.
- `animalHash` is the most recent cryptographic hash generated from the animal from its information stored in the database.
- `timeUpdated` is the unix timestamp when the most recent `animalHash` was stored in the chain.

Each entry of the `Farm` data structure is mapped through a 160 bit Ethereum `address` which is the farm id. Each entry of the `Animal` data structure is mapped through a 48 bit animal id number. The `FarmManager` contract contains several state variables:

- `farmCount` is the number of registered farms.
- `globalAnimalCount` is the total number of animals combining all the registered farms.
- `admin` is the 160 bit Ethereum `address` of the system administrator.
- `PM` is an object containing the address of the `ProfileManager` contract. The methods defined in `ProfileManager` can be used to validate user privileges (For example, checking user access level).
- `farmMap` is the mapping data structure that maps farm id to `Farm` object.
- `animalMap` is the mapping data structure that maps animal id to `Animal` object.

Initially, the creator of the smart contract is automatically added as the admin by the constructor function. The admin can register new farms, link `ProfileManager` contract deployed in the chain, and change admins. However, the admin cannot access the contents of `Farm` data objects; only the registered farm owners (i.e., `Farm.ownerId`) can do so. The methods (functions) described below contain code snippets to enforce such access control. The methods in this contract are listed below:

- `registerFarm()` adds a farm to the system, creates a `Farm` object and stores it in the contract. Only an admin can call this function.
- `getFarmInfo()` returns information about a farm based on the farm id (`address`) provided in the argument. Only a farm owner with proper access level can call this function.
- `updateFarmInfo()` updates information about the farm. Only a farm owner with proper access level can call this function.
- `checkOwnership()` validates if the owner referenced by the `ownerId` in the argument or the method calling entity owns the farm referenced by the `farmId` in the argument. It's an overloaded method.
- `farmExists()` checks whether a given farm referenced by the `farmId` in the argument exists or not.
- `registerAnimal()` adds an animal to the system, creates an `Animal` object and stores it in the contract. If the animal is already registered, the method updates the id of the farm that the animal is in.
- `updateAnimal()` updates the cryptographic hash of an animal. Only the owner of the farm that contains the animal can call this method.
- `getAnimalHash()` returns the cryptographic proof of the animal data that is stored in the contract.
- `animalExists()` checks whether a given animal referenced by the `animalId` in the argument exists or not.
- `setProfileManager()` instantiates the `PM` (i.e., `ProfileManager`) object with the Ethereum `address` of the Profile Manager contract deployed in the chain. Only an admin can call this function.

- `getProfileManager()` returns the address of the Profile Manager contract which is linked to this contract. Only an admin can call this function.
- `checkAdminAccess()` checks whether the current contract method calling entity has the admin level access (i.e., is the admin) or not.
- `changeAdmin()` assigns a new admin given the `address` in the argument. Only the current admin can call this function. Once called, if the contract transaction is confirmed, the current admin will lose its status as admin.

B.3 Transaction Manager

This contract handles business transactions that result in transfers of animals among farms. It provides methods for both the parties (sender and recipient) to initiate and confirm transactions. It also automatically calls appropriate methods of `TraceManager` to store movement data. The contract and its associated `Transaction` data structure are shown in Figure B.3. The `Transaction` data structure has the following components:

- `srcFarm` is the 160 bit Ethereum `address` of the source farm (id).
- `dstFarm` is the 160 bit Ethereum `address` of the destination farm (id).
- `srcOwner` is the 160 bit Ethereum `address` of the owner (id) of the source farm.
- `dstOwner` is the 160 bit Ethereum `address` of the owner (id) of the destination farm.
- `orderStatus` is the current state of the order. It can be either of the following values: `proposed (1)`, `confirmed (2)`, or `canceled (3)`.
- `animalCount` is the number of animals listed in this transaction.
- `animalMap` is the mapping data structure that stores the tags of the animals listed in the transaction. An auto incrementing index is used as the key which goes from `0` to `animalCount - 1`.

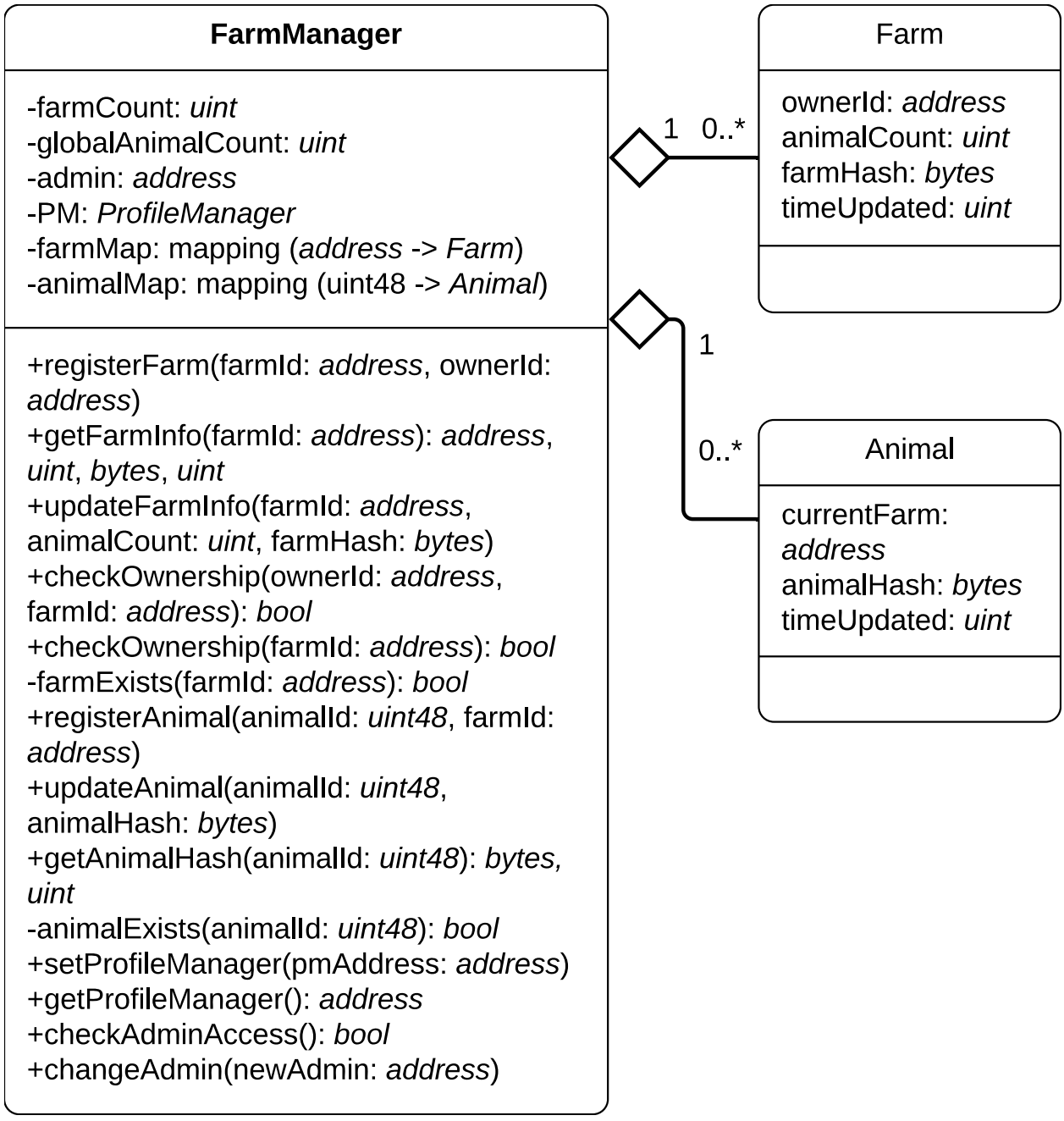


Figure B.2: *The* `FarmManager` *contract and its associated* `Farm` *and* `Animal` *data structures.*

- `timeUpdated` is the most recent unix timestamp when the transaction was created/modified.

Each entry of the `Transaction` object is mapped through a 48 bit transaction id which is the unix timestamp of when the transaction was generated by a client. The `TransactionManager` contract also contains several state variables:

- `transactionCount` is the number of transactions handled by the manager so far.
- `admin` is the address of the system administrator.
- `PM` is an object containing the address of the `ProfileManager` contract. The methods defined in `ProfileManager` can be used to validate user privileges (For example, checking user access level).
- `FM` is an object containing the address of the `FarmManager` contract. The methods defined in `FarmManager` can be used to validate the ownership of the farms (For example, checking if user A owns farm F).
- `TM` is an object containing the address of the `TraceManager` contract. The methods defined in `TraceManager` can be used to store animal movement data.
- `transactionMap` is the mapping data structure that maps transaction id to `Transaction` object.

Initially, the creator of the smart contract is automatically added as the admin by the constructor function. The admin can link other contracts such as, `ProfileManager`, `FarmManager`, and `TraceManager` deployed in the chain and change admins. However, admins cannot create, update, or access business transaction data; only the parties involved in the transaction can do so. The methods in this contract are listed below:

- `createTransaction()` creates a new transaction and submits it into the system to be processed by all the participants listed in the transaction. The method calling entity must be an owner of one of the farms involved in the transaction with necessary privileges.

- `updateTransaction()` updates the state of an existing transaction. The method calling entity must be an owner of one of the farms involved in the transaction with necessary privileges.

- `getTransactionInfo()` returns the contents of an existing transaction stored in the contract. The method calling entity must be either the `srcOwner` or the `dstOwner` listed in the transaction.

- `getAnimalList()` returns an array containing the tags of the animals listed in the transaction. The method calling entity must be either the `srcOwner` or the `dstOwner` listed in the transaction.

- `getTransactionCount()` returns the total number of transactions handled by the `TransactionManager` so far.

- `setManager()` instantiates one of the three manager contract (`PM, FM, TM`) objects with the Ethereum `address` of that respective contract deployed in the chain. The argument `managerType` determines which manager to instantiate (`ProfileManager (1)`, `FarmManager (2)`, or `TraceManager (3)`). Only an admin can call this function.

- `getManager()` returns the address of one of the manager contracts which is linked to this contract. The argument `managerType` determines which manager the query is about. Only an admin can call this function.

- `checkEligibility()` is a private helper method that validates the ownership of farms (by calling a `FarmManager` method) and access privileges (by calling a `ProfileManager` method) of users. This is used by other methods in this contract.

- `checkAdminAccess()` checks whether the current contract method calling entity has the admin level access (i.e., is the admin) or not.

- `changeAdmin()` assigns a new admin given the `address` in the argument. Only the current admin can call this function. Once called, if the contract transaction is confirmed, the current admin will lose its status as admin.

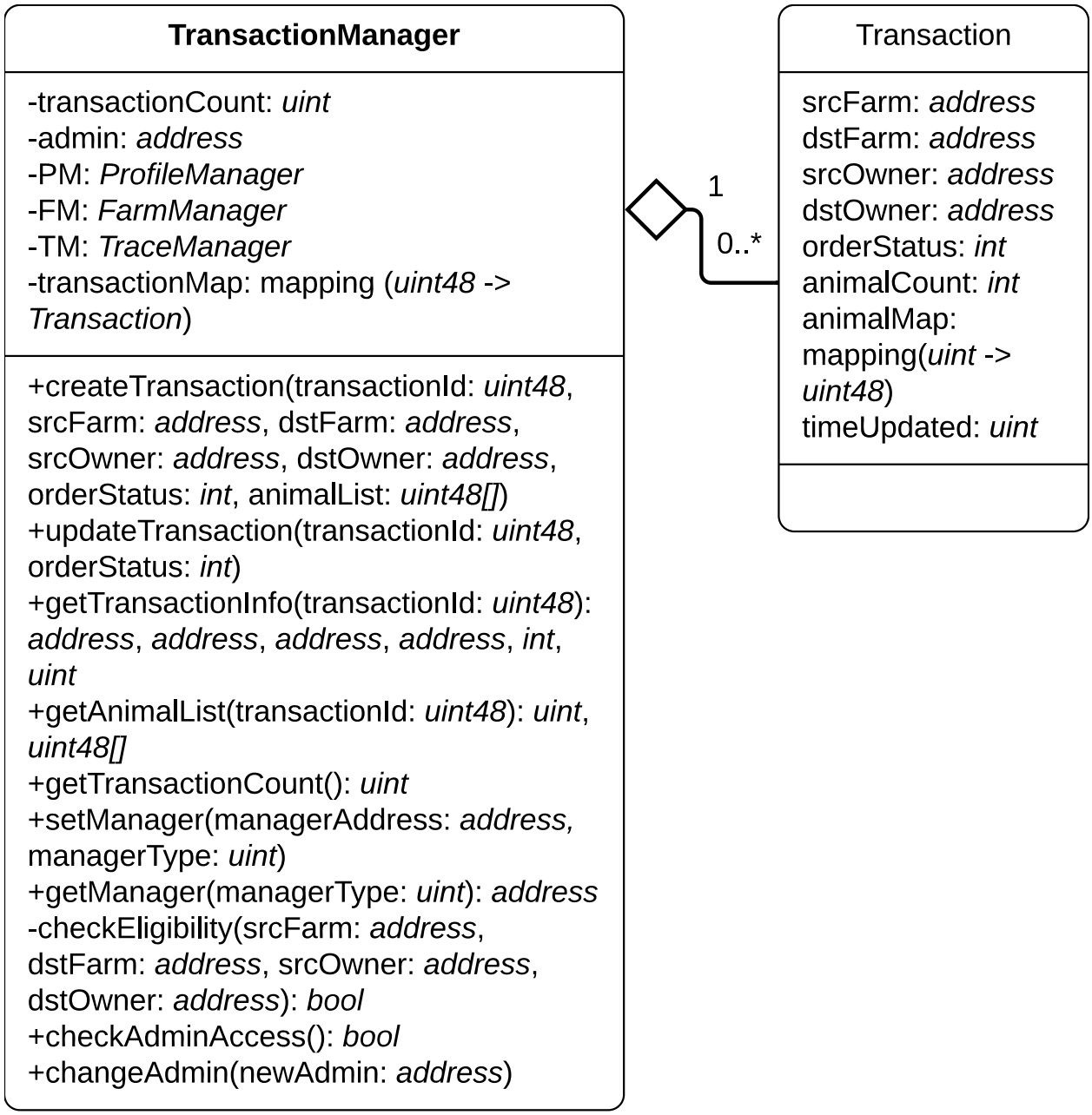


Figure B.3: *The* `TransactionManager` *contract and its associated* `Transaction` *data structure.*

B.4 Trace Manager

This contract enables traceability in the supply chain. It provides methods that the admin can use in urgent situations to trace the movements of targeted animals. The contract and its associated data structure are shown in Figure B.4. The `Animal` structure contains all the movement data encapsulated using the `Movement` structure, which has the following components:

- `srcFarm` is the id of farm that has sold/ delivered the animal.
- `dstFarm` is the id of farm that has purchased/ received the animal.
- `timeMoved` is the unix timestamp when the transfer (transaction) was confirmed.

Each entry of the `Movement` object is mapped through an 8 bit auto-generated index in the `Animal` object. The `Animal` structure contains the following components:

- `movementMap` is the mapping data structure that maps an unsigned integer index to `Movement` object. The index is handled automatically and it is local to each `Animal` object.
- `movementCount` is the total number of movement entries for an animal.

Each entry of the `Animal` object is mapped through a 48 bit animal id (tag) in the `TraceManager` contract. The `TraceManager` contract contains the following state variables:

- `animalCount` is the number of animals handled by the Trace Manager so far.
- `admin` is the address of the system administrator.
- `animalMap` is the mapping data structure that maps animal id (tag) to `Animal` object.

Initially, the creator of the smart contract is automatically added as the admin by the constructor function. The `TransactionManager` contract automatically uses methods from this contract in order to add/update trace data. However, only the admin can inquire this contract about movement data on a particular animal. The methods in this contract are listed below:

- `addMovementData()` adds a single entry of movement data for a particular animal.
- `getMovementCount()` returns the total number of movements that were recorded for a particular animal. Only an admin can call this method.
- `getMovementData()` returns a single entry of movement data for a particular animal given the animal id and the index of the movement. Only an admin can call this method.
- `checkAdminAccess()` checks whether the current contract method calling entity has the admin level access (i.e., is the admin) or not.
- `changeAdmin()` assigns a new admin given the `address` in the argument. Only the current admin can call this method. Once called, if the contract transaction is confirmed, the current admin will lose its status as admin.

B.5 Data Aggregator

This contract provides data structures and methods using which the network administrator can collect and manage anonymous survey data on the farming industry. The contract and its associated `Dataset` structure are shown in Figure B.5. The `Dataset` structure has the following components:

- `dataMap` is the mapping data structure that maps an integer index to `byte` data. The index is handled automatically and it is local to each `Dataset` object.
- `dataCount` is the total number of data entries for a `Dataset`.

Each entry of the `Dataset` structure is mapped through an unsigned integer which is regarded as the key of the data set. In addition to the `Dataset` structure, the data aggregator has the following state variables:

- `datasetCount` is the number of data sets handled by the aggregator so far.

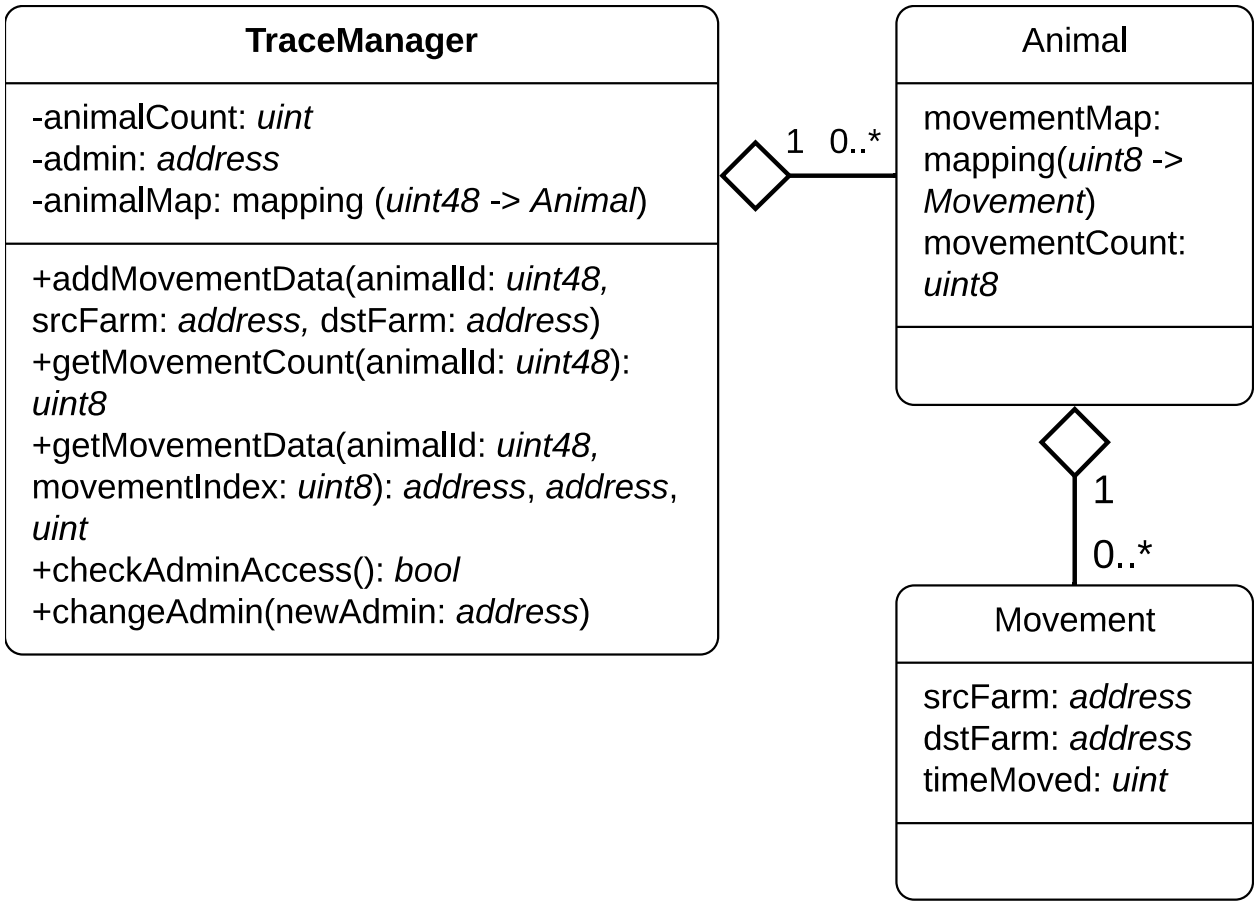


Figure B.4: *The* `TraceManager` *contract and its associated* `Animal` *and* `Movement` *data structures.*

- `admin` is the address of the system administrator.
- `datasetMap` is the mapping data structure that maps an integer key to a `Dataset` object.
- `secretMap` is the mapping data structure that maps an integer key to a confidential access code / password (regarded as the secret).

Initially, the creator of the smart contract is automatically added as the admin by the constructor method. The admin can add and configure new datasets, associate `DataSet` keys with secrets, and retrieve stored data. The users can add entries in a dataset if they can validate with the correct key-secret pair. This contract has the following methods:

- `addData()` adds a single entry of `byte` data for a particular dataset. The key and the secret given in the argument determines the set and authenticates the entry.
- `createDataset()` creates and configures a new `Dataset` object. Can only be called by the admin. The key and the secret provided in the arguments must match in the future for user entries.
- `getDataCount()` returns the total number of entries that were recorded for a particular data set. Only an admin can call this function.
- `getData()` returns a single entry of data for a particular dataset given the key, secret, and the data index. Only an admin can call this function.
- `checkAdminAccess()` checks whether the current contract method calling entity has the admin level access (i.e., is the admin) or not.
- `changeAdmin()` assigns a new admin given the `address` in the argument. Only the current admin can call this method. Once called, if the contract transaction is confirmed, the current admin will lose its status as admin.

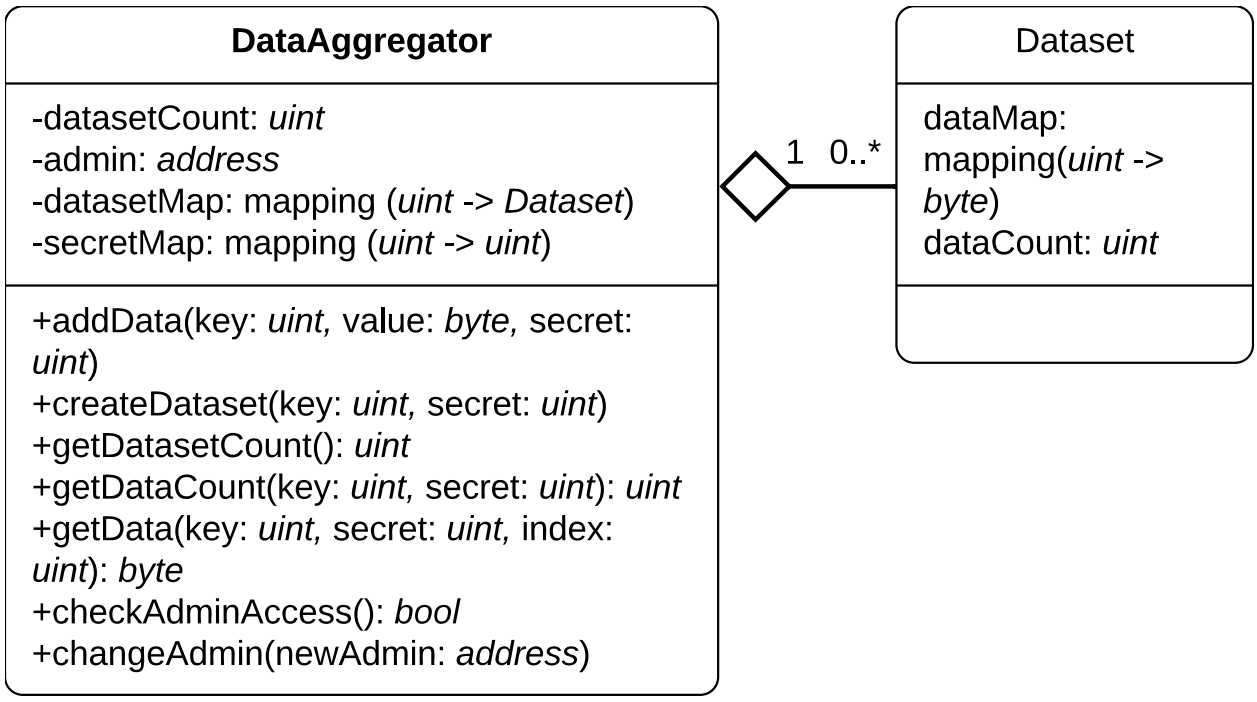


Figure B.5: *The* `DataAggregator` *contract and its associated* `Dataset` *data structures.*

Appendix C

Farm Animal Database

The blockchain nodes store the smart contracts and the Ethereum transaction (should not be confused with business transactions) database. Each contract has storage options in the form of state variables (including objects and arrays of objects). Despite that, we do not store raw farm data in the contracts as it would be computationally expensive to maintain, and the owners will feel a loss of control with their data. Instead, we use a separate Relational Database Management System (RDBMS) to store raw farm data. We used MySQL in our test bench; however, any SQL-based DBMS can be used. In each premise running a local farm RDBMS, there are two databases (db) with identical schema: present and archive. The present db contains information about the animals currently owned/maintained by the farm. The archive database is for storing historical data of the animals which existed in the farm once but were sold. Once an animal is sent out to another farm, all its relevant data from the present database is moved to the archive database. The new owners of the animal may request data during the purchase or sometime in the future. However, the decision of how long the data should be kept in the archive and what data could be shared is at the farm owner's (hence, data owner's) discretion.

As already mentioned, both the present and the archive dbs contain the same structure of tables. The db schema is shown in Figure C.1. The `animals` table is the root table. Each animal is uniquely identified by the tag. The id fields in the table can vary and are only used

for indexing and linking data. The true animal id that remains unchanged throughout the animal's lifetime is the `tag` number which is also used by the contracts to identify animals (`animalId`). The entries of the `inspections`, `vaccines`, and `movements` table are linked to the `animals` table via the use of foreign key, `animal_id`. In these 4 animal data tables, we concatenate the key fields (excluding `id`, `animal_id` fields) in each row entry and compute a SHA3-224 hash of the concatenated string. This hash is stored in the last field of each row entry. For a specific row entry, the hash is computed once and does not change even if the animal undergoes a change. For example, a new movement of the animal results in a new row entry for that animal in the `movements` table. The old entries (if any) remain untouched. To store the cryptographic proofs of the data, we combine hashes and compute a hash of the concatenated hashes. For each table, the hashes that are concatenated are ordered by the primary key, and a SHA3-224 hash is computed for the string of hashes. The resultant hash is called a table hash. We store the table hashes in the respective fields of the `data_proofs` table. Once again, the four table hashes in a row entry of the `data_proofs` table are concatenated, and a single hash is computed, which is called `joint_hash`. This is the top-level hash for a single farm database and is stored in the `FarmManager` contract of the blockchain.

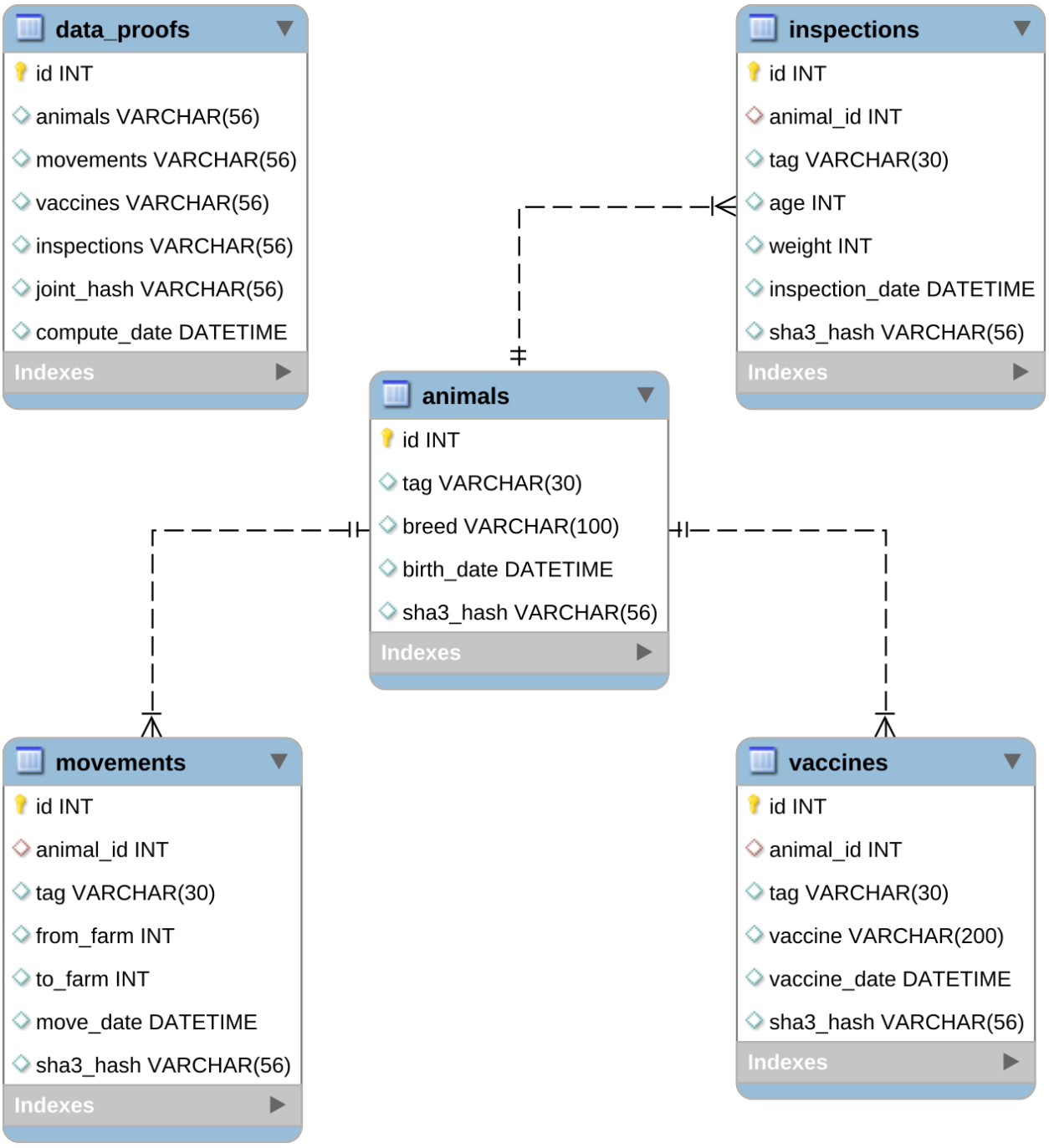


Figure C.1: *An entity relationship diagram depicting the database tables and their relationships.*

Appendix D

Test System Configuration

The test simulation was run on a system whose configuration is described in Table D.1. We use both parity[201], and geth[200] as the Ethereum clients as they support proof of authority (PoA) (also called 'clique' in geth) as a consensus protocol. Both parity and geth can produce similar results. The computational costs (gas cost) were computed by connecting to parity nodes, while the integration tests were performed by creating a prototype running system using geth. We ran 6 geth nodes in the same Linux system with separate data directory, configuration file, keystore, and port numbers for the prototype system. We created a single user account for each node: which would either take part in PoA consensus to validate and sign blocks or deploy smart contracts and invoke contract methods. We configured 5 out of the 6 nodes to participate in the PoA consensus (mining nodes), and node 6 was a basic full node. We used the web-based remix[202] IDE (integrated development environment) to write, deploy, and test the smart contracts which are written in the Solidity[203] language. Contract deployment and method invocations were done from remix via web3[204] by connecting to the running geth nodes of the network via RPC (remote procedure call) ports. The prototype system is illustrated in Figure D.1.

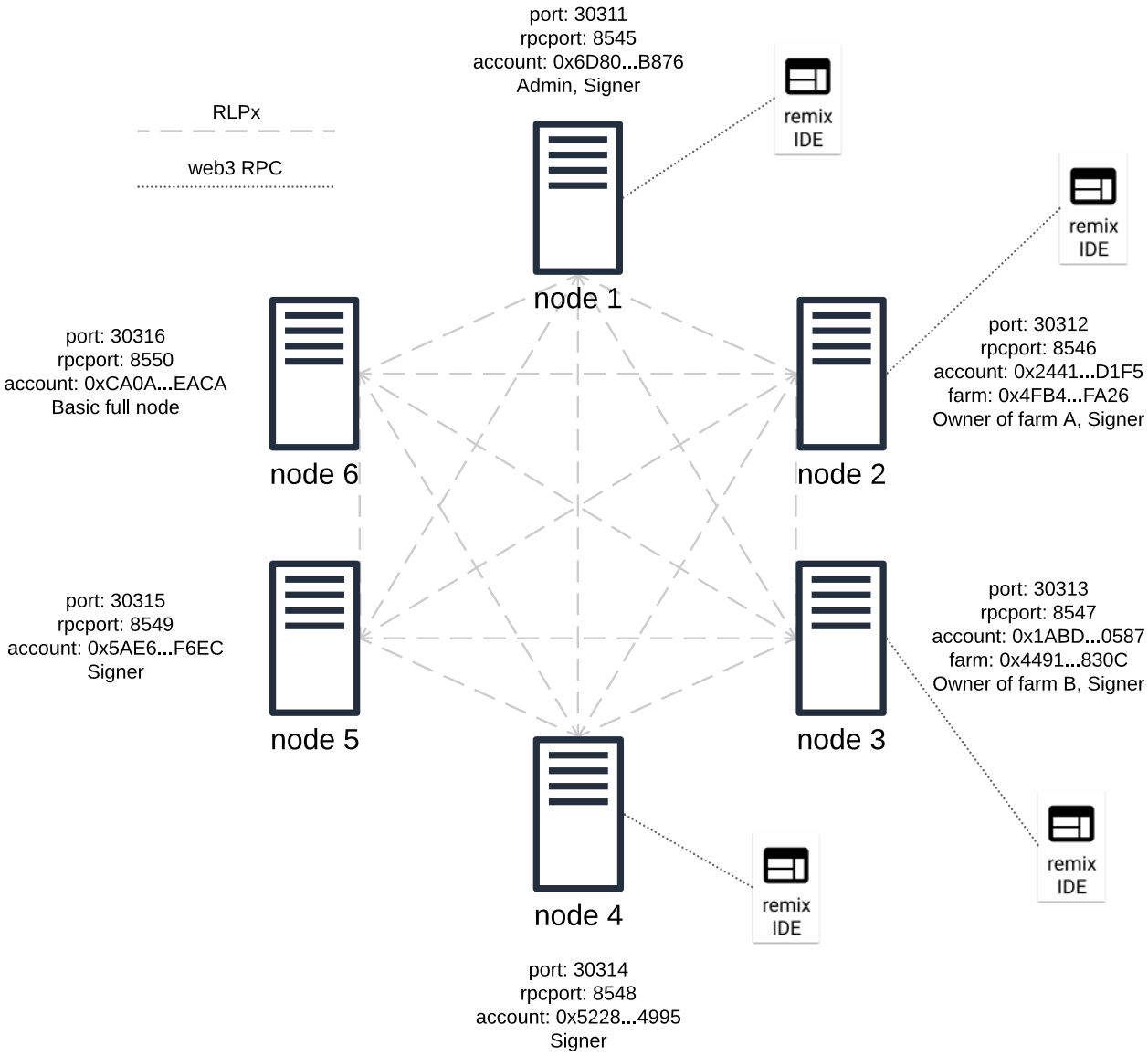


Figure D.1: *The prototype running system which was used for integration tests. The 6 geth nodes are shown connected in a P2P network. Each node is configured with unique ports for RPC and RLPx communications. Each node has an associated user account with an Ethereum address, as shown above. Five out of six nodes were configured as signers; the first node was used as Admin. Remix IDE was used to communicate with the nodes.*

Processor	Core i5-3470 3.20 GHz x 4
Memory	8138 MB
Storage	R/W 550/400 MB/s
Kernel	Linux 4.15.0-96-generic
OS	Ubuntu 18.04.4 LTS
Architecture	x86-64
Parity version	2.7.2
Geth version	1.9.13
Remix version	0.10.1
Solidity compiler version	0.5.3
Consensus protocol	Proof of Authority (PoA)
Step duration	5 seconds
No. of authorities (signers)	5

Table D.1: *Test bench configuration*