Studies on Approximate Bayesian Computation and Speedup of Spatial Data Access Using Distributed Quadtrees

Mayumbo Nyirenda

August, 2017

Division of Computer Science

Graduate School of Information Science and Technology

Hokkaido University

Abstract

Rapid advances in information technology are driving epidemiological research towards a focus on the predication of diseases dynamics. One of these technological advances is the extremely large amounts of observational data that can be analyzed computationally to reveal interesting patterns of information. Another is the increase in computational power which allows for simulations using agent based models. Coupled with these advances is improvements in mathematical models for epidemics and the need for efficient resource planning to prevent major disease outbreaks. To this effect data assimilation is increasingly becoming important in offering a way to forecast and estimate epidemic dynamics by using initial conditions and model parameters to constrain a mathematical model used for simulation to yield results that approximate the real world fairly well.

Many Mathematical models describing the epidemiological dynamics of diseases have been proposed. However the dynamics for multi-strain diseases such as influenza B are complex. Computation of likelihoods for such models is often intractable and thus rendering most studied and used approaches inapplicable. This problem is further amplified by the computational cost of simulating such dynamics. Furthermore data assimilation involving large amounts of spatial data faces a data access bottleneck. In this thesis we therefore propose a mathematical model and a procedure that uses a likelihood-free approach for epidemic estimation. Further, We propose an architecture based on dis-

tributed quadtrees for accelerating access to large amounts of spatial data to alleviate the data access bottleneck.

In Chapter 3 we propose and discuss a likelihood free procedure for epidemic estimation. We also propose a mathematical model for complex multi-strain epidemic dynamics. We then use our proposed procedure and model to estimate the dynamics of influenza B. Using the estimated parameters, we predicted the dominant lineage in 2015-2016 season in Japan. The accuracy of this prediction is 68.8% if the emergence timings of the two lineages are known and 61.4% if the emergence timings are unknown. This demonstrates the practical applicability of our proposed procedure and model.

In Chapter 4 we consider the use of distributed quadtrees in a shared-nothing memory approach to reduce the data access bottleneck in data assimilation systems. We distribute data across nodes and construct a directory for the distributed nodes by using a quadtree built from sampled points. We discuss approaches for partitioning and allocating data and queries across the distributed nodes. Results from the experiments we conducted using a scale-down parallel data load and search distributed processor system show that a collection of small indices of distributed shared-nothing memory is more efficient than the conventional approach with a single processor with a large external index.

In Chapter 5 we take into account the process of query redirection during the construction of the distributed quadtree as well as query redirection during a data retrieval process. We propose taking advantage of the static nature of the sample points of the data and use of hashmaps and dilated integers to speed up traversal of the directory. Results from the experiments conducted show a threefold improvement in performance and also show less sensitive to data skewness.

Finally, in Chapter 6 we conclude this thesis and discuss future researches. The main

point and focus of this thesis is the application of already existing techniques to improve data assimilation processes with application to real-world problems. We propose a mathematical model and use Approximate Bayesian Computation as a likelihood free data assimilation tool to estimate the dynamics of influenza B. Results from the experiments conducted show that our proposed approach is capable of learning the essential parameters of influenza B required to predict the dominant lineage of the following year. Furthermore our proposed architecture for acceleration of spatial data access in data assimilation systems results in significant gains in performance.

Acknowledgments

First and foremost I would like to thank Professor Hiroki Arimura for the guidance and support that he offered. For the knowledge he imparted in me and for his supervision during my research work. I would also like to thank Associate Professor Takuya Kida for the support he rendered materially.

I would also like to thank my advisers Professor Makoto Haraguchi and Professor Shin-ichi Minato for their valuable contributions and advise.

A special thank you to Professor Kimihito Ito and Assistant Professor Ryosuke Omori for their supervision and advise. Japan was a home away from home all because of the wonderful fellowship they gave me.

I would also like to thank the members of the Information Knowledge Laboratory of the Graduate School of Information Science and Technology and also the members of the Research Center For Zoonosis studies research laboratory. Friends in need and indeed they are to me.

Last but not the least I would like to thank my family for being there for me through all the hard times during my studies. The wonderful support I received from my wife is indescribable. Without you I would not have reached this far. I would also like to thank my children for understanding me even when I was so far away from them. I would also like to thank my siblings, my mother and the rest of my friends and family.

Contents

Chapter1

Introduction

1.1 Background

By rapid increase of computational power and the amount data collected worldwide, data analytics systems such as data assimilation systems have attracted much attention. Data assimilation is a statistical method by which actual observations are integrated into computer simulations of mathematical and numerical models of the system dynamics using statistical inference or machine learning methods. For instance, ensemble Kalman filters and particle filters [19, 22] are innovative examples of such inference or learning methods. So far, data assimilation has been applied in prediction of epidemics, meteorology, oceanography, engineering, and life science, achieving significant improvements in the accuracy of predictions. For example, using sensor data from ocean buoys and the ocean floor, oceanographers in the United States predicted the tsunami of the 2011 Tohoku earthquake in Japan in real-time and provided important information immediately after the earthquakes.For data assimilation of epidemics much study has be done for single lineage analysis.

The influenza virus is one of most common respiratory viruses and causes a high disease burden worldwide [38]. The influenza viruses co-circulating among humans can

be classified as influenza A and influenza B viruses. Approximately 75% of confirmed cases of influenza are infections by the influenza A virus [12]. The disease burden of influenza B is also high, and 25% of confirmed cases of influenza virus infection and 22-44% of pediatric influenza related deaths in the US are caused by influenza B [12, 55]. The number of major lineages of influenza B is relatively low compared to type A. There are two major genetically and antigenically distinct lineages; the Yamagata lineage and the Victoria lineage. Trivalent vaccines against influenza include one of those two lineages. The selection of the correct vaccine lineage is essential for high vaccine efficacy against influenza B infections. Despite the limited number of existing influenza B lineages, vaccine strain selection is still difficult because the dominant lineage changes over time and the switching time of the dominant lineage is difficult to predict. Although the quadrivalent vaccine includes both influenza B lineages, Hopping et al. 2016 pointed out the necessity of vaccine strain selection because the use of trivalent vaccines is still common worldwide and the cost-effectiveness of quadrivalent vaccines is under debate [17].

To predict an effective vaccine strain for influenza B, a good model capturing the mechanisms of its complex dynamics is needed. This can be achieved by using individual based models. Consequently such models are often computationally expensive. Furthermore the likelihoods associated with parameters of such models are often intractable or highly computationally expensive to compute. Important factors to consider regarding the complex epidemic dynamics of the Yamagata and Victoria lineages are

1. i) the seasonal variation of transmissibility,

2. ii) epidemiological interference between the two lineages, and

3. iii) time series changes of antigenicity due to the evolution of the pathogens.

The incidence of influenza B shows seasonal fluctuation which can be explained by the seasonality of the transmissibility of influenza B. This transmissibility seasonality has been shown to be determined by the seasonal variation of absolute humidity [48, 49]. Previous theoretical studies have shown that seasonal variation in transmissibility can induce rich epidemic dynamics, such as periodic or chaotic behavior [14, 47, 52], therefore a model capturing seasonal fluctuation of transmissibility is essential to predicting epidemic dynamics.

Epidemiological interference is a known factor in complex epidemic dynamics [6, 15, 23, 44]. The time series of confirmed cases between two lineages are negatively correlated, implying epidemiological interference between the two lineages. Moreover, vaccine efficacy studies also imply the existence of immune cross-reaction between lineages [17]. The epidemic dynamics of a lineage are affected by that of the other lineage assuming the existence of immune cross-reaction. The change of antigenicity of influenza across seasons is one major obstacle for prediction. Especially in the case of influenza A, the large variety of lineages and epidemic interference between these lineages makes it difficult to predict the dynamics. To analyze these complex dynamics, models taking into account these complex evolutionary dynamics have been proposed so far [13, 28, 31, 36, 42, 58].

Although the number of co-circulating influenza B lineages is limited compared to influenza A the large genetic diversity within the lineages and the changing antigenicity over time, especially for the Victoria lineage [61], makes dominant lineage prediction difficult. Modelling the complex evolutionary dynamics is essential to predicting future changes in influenza B. This requires the use of data assimilation procedures that can learn from complex models. So far most works focus on prediction for one lineage. It is

therefore important that a procedure that can be used to learn the parameters and state of a complex model be developed. Other studies involve the analysis of epidemics using Monte Carlo Simulations such as [24, 32, 39, 46, 62, 66].

Another area of application of data assimilation is that of spatial data bcentric applications. However, real-time storage and retrieval of massive remote sensing data is an emerging bottleneck in data-centric application systems including data assimilation systems in meteorology. For instance in the data assimilation example we mentioned above, advancements in remote sensing technologies have increased the amounts, sources, and rates of capture of observed data. Taking advantage of these massive data pauses new challenges among which is input/output (I/O) related challenges which are an emerging bottleneck in data assimilation systems in meteorology [19, 37]. To provide more accurate predictions modern data assimilation assimilates a large number of observations in the order of millions often in less than an hour [63]. This makes Big Data assimilation I/O intensive [37]. Vastavia reports that in 2013 NASA was generating 5TB of data per day [60]. It is therefore important to investigate the properties of this bottleneck and give possible solutions to relax it by extending or adapting existing algorithms for spatial indexing. In this thesis we investigate the properties of this bottleneck and give a possible solution to relax it by extending or adapting existing algorithms for spatial indexing.

Observational data are also usually multidimensional and thus require specialized data structures in order to simplify their traversal and exploration [10]. In an experiment comparing the runtime for a data assimilation setup accessing files from RAM and a setup accessing files from a shared drive in a cluster of 40 machines, Miyoshi [5, 37] shows that performance improves threefold when data is accessed from RAM (taking 40

minutes to do the data assimilation) as opposed to a shared drive (taking 130 minutes). This experiment shows that improving I/O results in improved performance of data assimilation tasks. There are several data structures and algorithms for spatial indexing, and examples include quadtrees, k-d trees and R-trees [45].

In this thesis, we propose the use of distributed spatial indices on a distributed computing system in the shared-nothing in-memory approach. Our research especially focuses on quadtrees among many spatial and multi-dimensional indices because the quadtree shows relatively good performance for distributed data in the relatively low dimension, say, collections of locations in dimension $d = 2$ or 3 [45]. Furthermore, it is easy to implement than other multi-dimensional indices, and thus widely used by spatial databases and often found in scientific computations [8]. A conventional approach for storing large geographic data is to construct a single, large quadtree for the whole data stored in a single large external memory, say an array of hard disks or SSDs, of a single server processor. However, we can see from recent hardware trends that it is too expensive to have such a single computer with large storage for realizing massive data-centric application systems.

1.2 Scope of this thesis

Because of the challenges highlighted, we study approaches for estimating complex epidemics as well as how to improve the performance of data assimilation systems that use spatial big-data in this thesis .

In Chapter 3 of this thesis, we propose a procedure that can be used to estimate the dynamics of complex epidemic models using individual based models and Approximate Bayesian Computation. Furthermore, we construct a mathematical model for influenza B. We then estimate the parameters of our model from the time series of influenza B confirmed cases per lineage and time series of specific humidity. Using these estimated parameters we assess the predictive potential of the dominant lineage in the next season.

In Chapter 4, we propose an architecture for distributed quadtrees. We discuss static and dynamic partitioning and allocation strategies for data and queries across distributed nodes. We do not proceed to do any data assimilation using our proposed architecture, since the main objective is to find efficient means of transferring big data and we anticipate that efficient transfer of data can alleviate the bottleneck as demonstrated by Miyoshi et al [37]. We use a real geographic data set from NOAA, WMO, which consists of tens of thousands of atmospheric and whether observation stations in the globe. For queries, we used a randomly generated collection of spherical range queries with constant radius whose centers are uniformly distributed in the area. Using scale-down parallel data load and search experiments with a small distributed processor system as proof-of-concept, we show that the proposed approach with a collection of small indices of distributed shared-nothing memory is more efficient than the approach of keeping a large quadtree in the external memory. We also observed that the tree-based partitioning strategy using sampling reduces query time than other conventional partitioning strate-

gies used in databases. We also discuss how to allocate a collection of small tree indices among distributed processors. These results suggest that the use of parallelized access to databases with spatial indexing functions can enhance the throughput of large-scale data-centric applications.

In Chapter 5, we further improve our proposed architecture by taking advantage of the static nature of the base quadtree proposed in Chapter 4. We make use of dilated integers [50] and a hashing function to locate data servers that a data point should belong to. Since the tree is static it becomes possible to create a hashmap for the data servers. Using this hash function we can search the tree by height and thus reduce the redirection and forwarding time. The dilated integers are used to compute the locational code of a point at a given level in O(1) time. We also construct hashmaps for leaf nodes at each level of the quadtree. We then search the quadtree by height by searching for a matching server at each level as opposed to a sorted list of all the leaves. This is possible owing to the static nature of the base quadtree. This reduces the computation expense for locating the data servers. After locating the data server, the data point is then forwarded to the data server which then inserts it into its local quadtree. We proceed to perform experiments that show that our approach is less sensitive to the number of data points as well as the height of the base quadtree. This makes it more scalable and also less sensitive to the skewed nature of some spatial data.

Finally, we conclude this thesis with a discussion and propose future areas of research in Chapter 6.

1.3 Contributions of this thesis

In this thesis we focus on finding solutions to real-world problems with regards to data assimilation. We propose a procedure for estimation of the dynamics of muliti-lineage diseases and proceed to test our procedure with a real data. Furthermore we propose a architecture for speedying up access to spatial big-data. We also test this architecture with real-world data.

Our contributions are the following:

1.3.1 Data assimilation

Multi-lineage diseases require complex models to capture their dynamics. This makes them computationally expensive to simulate. Furthermore, their parameters have computationally expensive/intractable likelihood functions. This renders commonly used data assimilation techniques based on likelihood functions impractical. To solve this problem we propose a mathematical model and a procedure that uses a parallelised version of Approximate Bayesian Computation. We proceed to test the model and procedure on real-world data and successfully estimate the dynamics of influenza B.

1.3.2 Spatial Big-Data analytics

The use of big-data is gaining popularity. However, there are challenges related to the use of spatial big-data. To solve the related IO problems we propose an architecture that uses distributed quadtrees in a shared nothing environment. Experiments show that data access improves tremendously when using our proposed architecture as compared to the traditional on disc access approach.

1.4 Related work

1.4.1 Kalman filter based procedures for epidemic estimation

Forecasting disease dynamics is fundamental to managing epidemics. One way of doing this is by using filtering techniques. Amongst the most used techniques is the Kalman filter. When some aspects of the dynamics of a disease can not be observed, changes over time in the parameters can be incorporated by a recursive estimation technique like the Kalman filter. It offers a way to assess any parameter modifications included in new observations. The Kalman filter accounts for stochastic fluctuations in both the model and the data and observations. The Kalman filter approach has been applied to many simple differential models to describe observed epidemics. It is ideal for models where likelihood of parameters is tractable. Using a Kalman Filter, quantitative information on the time-evolution of some parameters of major epidemiological significance (average transmission rate, mean incubation rate, and basic reproduction rate) can be estimated. For influenza much of the observed wintertime increase of mortality in temperate regions is attributed to seasonal influenza. Amongst works that have been done include [7, 18, 26, 39]. Most notable of these is Shaman et al's study. In their study, Shaman et al use a Kalman filter to learn the essentials parameters of influenza which are required to make predictions. They however apply this study to a model that does not account for multiple lineages and cross immunity amongst the lineages. Such models are often easier to compute likelihoods for and hence suitable for Kalman filtering. In the absence of such tractable likelihoods the Kalman filter becomes an inadequate tool.

1.4.2 Particle filter algorithm for epidemic estimation

An alternative to Kalman filter is the use of sequential Monte Carlo methods for estimating dynamics of epidemics. Dukic et al used particle filters with data from Google Flu Trends together with a sequential surveillance model based on the state-space methodology, to track the evolution of an epidemic process over time [9, 56]. They use a compartmental mathematical model within a state-space procedure hence extending the dynamics of the model to allow changes through time. The particle filtering algorithm was used to learn about the epidemic process sequentially through time. It was also used to provide updated estimated possibility of a pandemic with each new surveillance data point. By combining the model with sequential Bayes factors, it provides a tool for online analysis of an influenza pandemic. Also noteable of their procedure is the use of Google Flu Trends. Similar to the Kalman Filter, this methods requires a non-complex model with tractable likelihood for its parameters. This makes it hard to use with models that have computationally expensive or intractable likelihood for their parameters like multi-lineage disease epidemics.

1.4.3 Spatial indexing data structures

A **Sample based quadtree with Map Reduce (SQMR)** was proposed by Min and Noh [45]. The basic idea behind their proposal is the reduction in size of the data points used to build the base quadtree. To achieve this they sample the data and use the sample to build the directory structure and hence save on both time and space. They make use of Map-reduce to implement their spatial structure. In order to construct the quadtree they begin by sampling the entire data set using the mapping process. Then, they use the mapper to construct a quadtree which they call the base quadtree and use it

as an approximation of the tree structure had the construction be done using the entire data set. Each mapper then assigns each leaf of the base quadtree to a reducer. Then, each reducer builds a local quadtree. The final stage of their algorithm involves the aggregation of the local quadtrees to form the global quadtree. The solution is solely based on external storage. Their experimental results show that construction time is reduced when a sample is used as opposed to using an entire data set to build the base quadtree.

The SQMR focuses on improving quadtree construction time by distributing the work load. However since the quadtree is consolidated into a global tree, querying is left to the mappers only. This is ideal when using external storage but isnt achievable with an in-memory approach as the tree is too big to fit in memory.

1.4.4 Other researches

X-Switch: An Efficient, Multi-User, Multi-Language Web Application Server is a Web application server that over comes the bottleneck related to process re-initialization by using persistent processes [40]. Mayumbo, Suleman and Maunder show that performance which is often a problem in server applications can be improved by reducing the need to re-initialize the Web application process with each access.

Chapter2

Preliminaries

In this chapter, we define the terms and notation used in this thesis. We also introduce mathematical models used to model epidemics. We then proceed to discuss data assimilation algorithms that can be used for parameter estimation of epidemic models. Further on, we proceed to discuss the pros and cons of these algorithms. We also discuss the role of spatial Big-data in data assimilation. Finally, we outline the problems that the rest of this thesis focuses on.

2.1 Mathematical models for epidemics

An epidemic is a rapid spread of an infectious disease from person to person to a large number of people in a given population. One of the most famous examples of an epidemic is the great plague of London which led to many deaths in the period 1665-1666. Likewise, it is of great importance to be able to model how such an infection spreads through the population in order to understand the dynamics of such a disease. Amongst the many models proposed are those based on dividing the host population (humans in this case) into a small number of compartments, each containing individuals that are identical with regards to their status with respect to the disease in question. Once such

mathematical model is the Susceptible Infectious Removed (SIR) model.

2.1.1 SIR based models

The SIR model is mathematical model that uses three compartments to classify hosts at time t as;

(i) *Susceptible*($S = S(t)$): The number of individuals that have no immunity to the infectious disease and may become infected if exposed

(ii) *Infectious*($I = I(t)$): The number of individuals that after exposure are now infected with the disease

(iii) *Removed*($R = R(t)$): The number of individuals who are immune to the infection and thus can not affect the transmission dynamics in any way.

The total host population is of size $N = S+I+R$. N, S, I and R are logically integers since they represent the number of hosts in each compartment. However, if N is sufficiently large enough we can treat S, I and R as continuous variables and express how hosts move from one compartment to another as a set of differential equations.

$$\frac{dS}{dt} = -\beta S(t)I(t), \tag{2.1a}$$

$$\frac{dI}{dt} = \beta S(t)I(t) - \gamma I(t), \tag{2.1b}$$

$$\frac{dR}{dt} = \gamma I(t) \tag{2.1c}$$

where β is the transmission rate (rate of getting infected) and γ is the recovery rate. Assuming that all hosts are initially without immunity then ($S(0) = N$) and a newly infected host will infect a susceptible individual at a rate βN during the infective hosts infectious period $1/\gamma$. Furthermore the infective host can be expected to infect $\beta N/\gamma$

hosts. There are many extensions to the SIR model but in this thesis we will focus more on the SEIR and SEIRS models.

SEIR model

The $SEIR$ introduces another compartment for hosts, E, that have been *exposed* to the infectious disease and are infected but are not yet infective (can not cause other infections). Short periods of exposure have less impacts on the dynamics of an epidemic. However longer periods do make a significant difference to model predictions. Thus we can now model the transitions from one compartment to another as:

$$\frac{dS}{dt} = -\beta S(t)I(t), \tag{2.2a}$$

$$\frac{dE}{dt} = \beta S(t)I(t) - \epsilon E(t), \tag{2.2b}$$

$$\frac{dI}{dt} = \epsilon E(t) - \gamma I(t), \tag{2.2c}$$

$$\frac{dR}{dt} = \gamma I(t) \tag{2.2d}$$

where κ is the rate of becoming infective after exposure and composition of the total host population is $N = S + E + I + R$.

SEIRS model

When analyzing epidemics over a long period of time, it becomes important to consider the loss of immunity by hosts that had initially acquired immunity after infection and recovery. Such models incorporate the rate of loss of immunity and thus hosts move from the R compartment and back to the S compartment at a rate κ. We can express the transitions as:

$$\frac{dS}{dt} = -\beta S(t)I(t) + \kappa R(t), \tag{2.3a}$$

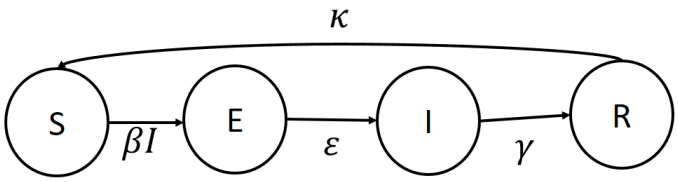


Figure 2.1: SEIR compartmental model

$$\frac{dE}{dt} = \beta S(t)I(t) - \epsilon E(t), \tag{2.3b}$$

$$\frac{dI}{dt} = \epsilon E(t) - \gamma I(t), \tag{2.3c}$$

$$\frac{dR}{dt} = \gamma I(t) - \kappa R(t) \tag{2.3d}$$

Figure 2.1 is an illustration of the transmission of infection in an SEIRS model.

2.1.2 ABC rejection sampling algorithm

Data assimilation techniques rely on learning the state of a system using mathematical models. As such, many models have been developed which capture the related structure and parameters of the model with appropriate values to considerably represent the process. However learning the parameter values for real-world problems can be more than challenging. There many methods for parameter learning and estimations. However most of these method assume that the likelihood associated to the parameters is tractable and cheap to computer. This is however a challenge when dealing with complex models whose likelihood may be intractable.

In such cases Approximate Bayesian Computation [2, 54, 59] has demonstrated to

be a good method for the estimation of parameters and state. Likelihood free methods for parameter estimation spring from the logic that given data D_o from an observation, we can use this data to fine tune our model for simulating the environment by drawing parameter sets θ, defining the state of the system, such that the simulated data is the same or as close as possible to the observed data(See Equation 2.4).

$$p(\theta|D_o) \sim (D_o|\theta)\pi(\theta) \tag{2.4}$$

The likelihood function $p(D_o|\theta)$ is at times not derivable. In such a case we would like to have a likelihood-free computation that can be used to estimate θ. Thus we develop a model that generates data that is close to the observed data and accept the parameters as coming from the true distribution that defines θ.(see Algorithm 1)

Algorithm 1 Pure rejection sampling

1: Draw $\theta \sim \pi(\cdot)$

2: Simulate $x \sim f(\cdot|\theta)$

3: If $s(y) = s(x)$ then accept θ

4: Repeat lines 1, 2 and 3 until N samples are drawn

However not so many models and simulations will produce results that match the criteria of algorithm 1. It is often not possible to match the observed data exactly. For this reason the rejection criteria is relaxed a bit by accepting θ such the simulations generates results that are slightly different from the exact value with a certain level tolerance. Such algorithms like Algorithm 2 use the assumption that

$$p(\theta|D_o) \approx p(\theta|d(D_s, D_o) \leq \epsilon) \tag{2.5}$$

where there is a tolerance $d(D_s, D_o) \leq \epsilon$ as the basis of relaxation of the exactness constraint. The relaxation of the rejection criteria also reduces the time required to generate

samples from a distribution which is not the true posterior distribution of interest, but a distribution which is hoped to be close to the real posterior distribution of interest.

Algorithm 2 Rejection sampling

1: Draw $\theta \sim \pi(\cdot)$

2: Simulate $x \sim f(\cdot|\theta)$

3: If $\rho(y, x) < \epsilon$ then accept θ

4: Repeat lines 1, 2 and 3 until N samples are drawn

More often than not the Euclidean distance is chosen as a measure of the distance of the simulations from the observations. Thus

$\theta^{\star}\big|\|D_o - D_s\|| < \epsilon.$

This implies that small values of ϵ result in selection of parameters that closely approximate the true posterior. However, smaller choices of ϵ will lead to higher rejection rates and is particularly a problem when dealing with high-dimensional D_o, where it is often unrealistic to expect a close match between all components of D_o and the simulated data D_s, even for a good choice of θ. In this case, it makes more sense to look for a good agreement between particular aspects of D_o, such as the mean, or variance, or auto-correlation, depending on the exact problem and context.

If the data is continuous or highly dimensional, then data may be summarized using a lower dimensional set of summary statistics S such that $p(D_o|S, \theta)$ is independent of θ and $p(\theta|D_o) = p(\theta|S(D_o))$. Then S is said to be a sufficient statistic(Marjoram et al., 2003). Algorithm 3 is an approximate Bayesian computation that makes use of the approximation.

Algorithm 3 ABC rejection sampling

1: Draw $\theta \sim \pi(\cdot)$

2: Simulate $x \sim f(\cdot|\theta)$

3: If $\rho(S(y), S(x)) < \epsilon$ then accept θ

4: Repeat lines 1, 2 and 3 until N samples are drawn

2.1.3 Markov chain Monte Carlo ABC algorithm

Markov chain Monte Carlo (MCMC) sampling is a general technique that filters proposed values for θ to arrive at a sample of values drawn from the desired posterior distribution. It has been used in various sampling techniques but most notable in the Metropolis-Hastings algorithm. The Metropolis-Hastings [16, 34] algorithm first selects some initial value θ_0 for θ. It then considers it as a candidate value θ^* from a proposal distribution $q(\cdot|\theta_0)$ conditioned on the initial value θ_0. Let the proposal distribution q be Gaussian and thus θ^* be drawn from q,i.e $\theta \sim N(\theta_0, \sigma^2)$ and q follows a *Gaussian distribution* with mean θ_0 and variance σ^2. We then reject of accept θ^* based on rejectio rate determined by the likelihood. We then acceptnθ^* and set $\theta_1 = \theta^*$, or we reject it and set $\theta_1 = \theta_0$. This process is repeated until we obtain a chain of values $\theta_0, \theta_1, \ldots, \theta_m$ that we can assume are a sample from the posterior distribution $\pi(\theta|Y)$. The Metropolis-Hastings algorithm can be very efficient, especially when the prior distribution $\pi(\theta)$ differs substantially from the posterior distribution $\pi(\theta|Y)$. However, computing the acceptance probabilities to generate the chain $\theta_0, \theta_1, \ldots, \theta_m$ requires an expression for the likelihood. To overcome the need for likeliood, MCMC can be coupled with ABC as shown in Algorithm 4. MCMC ABC has been used by [33, 43] before.

When selecting the proposal for θ^* not only must we meet the acceptance probability of the Metroplois-Hastings sampler, we must also generate data that is sufficiently close

Algorithm 4 MCMC ABC algorithm of Marjorum et al

1: $\theta^0 \leftarrow$ ABC rejection procedure

2: **for** $i = 1 \dots N$ **do**

3: Draw $\theta^* \sim q(\cdot|\theta^{i-1})$

4: Simulate $x^* \sim f(\cdot|\theta^*)$

5: Compute: $w = min\left(1, \frac{\pi(\theta^*)q(\theta^{i-1}|\theta^*)}{\pi(\theta^{i-1})q(\theta^*|\theta^{i-1})}\delta_\epsilon\right)$, where $\delta_\epsilon = \mathbb{I}_{[0,\epsilon]}[\rho(y, x^*)]$

6: **if** $uniform(0, 1) < w$ **then**

7: $\theta^i = \theta^*$

8: **else**

9: $\theta^i = \theta^{i-1}$

10: **end if**

11: **end for**

to the observed data, that is, with a small enough distance ϵ. Meeting these two criteria has drastic effects on the MCMC ABC algorithm. It is known that most MCMC series of execution are prone to getting stuck. The ABC MCMC algorithm is no exception to this and is especially greatly likely to get stuck because of the two criteria that the proposal θ^* must meet as explained. For this reason, the rejection rate of ABC MCMC can be extraordinarily high thereby disproportionally large computing cycles for even relatively simple problems. Another important aspect of the MCMC ABC algorithm is the complications which arise when trying to parallelize it. Consequently we will not consider the ABC MCMC algorithm further. We instead focus on the SMC ABC algorithm.

2.1.4 Sequential Monte Carlo ABC algorithm

Sequential Monte Carlo sampling uses a particle filter as opposed to drawing samples singly. Furthermore it is easier to parallelise as compared to MCMC. It also differs from the MCMC approach in that, rather than drawing possible θ^* one at a time, these algorithms work with large pools of candidates, called particles, simultaneously. This makes the parallelisation of SMC particle filters attractive. Through each iteration of the algorithm, the pool of particles draws closer and closer to a sample drawn from the desired posterior. Particle filtering algorithms start by generating a pool of N candidate values for θ from by sampling from a prior distribution $\pi(\theta)$. Then, in subsequent iterations, particles are chosen randomly from this pool, and the probability of any particle being sampled depends on a weight assigned to that particle. Immediately after initialization, all particles have the same importance and thus are assigned equal weight $1/N$. The major difference between SMC algorithms is the way the weights are calculated and assigned to particles in the pools as the filtering progresses in subsequent iterations. Algorithm 5 shows an ABC based approach for calculating weights to use SMC with ABC as proposed by [30, 57].

2.2 Spatial indexing data structures

Geographic Big-data analytics applications performance is hampered by information retrieval from spatial data. Retrieval of necessary information from spatial data is associated with what are known as spatial queries in database management systems. A naive method for spatial queries is to check all the points. Consider a situation where a given geographic point needs to find all observations from remote sensing that lie within a given distance to it. In the absence of any organization, this query requires checking

Algorithm 5 SMC ABC algorithm

1: Initialise $\epsilon_1, \ldots, \epsilon_T$ and specify the initial importance sampling distribution $\pi\theta(\cdot)$

2: **for** $t = 1 \ldots T$ **do**

3: **for** $i = 1 \ldots N$ **do**

4: If $t = 1$ sample θ^{**} from $\pi\theta(\cdot)$. If $t > 1$ sample θ^{*} from the previous population $\{\theta_i^{t-1}, W_i^{t-1}\}_{i=1}^{N}$ and perturb the particle $\theta^{**} \sim Kt(\cdot|\theta^{*})$.

5: Generate a dataset $x^{**} \sim f(\cdot|\theta^{**})$.

6: If $\rho(y, x^{**}) > \epsilon_t$ then go back to step 4.

7: Set $\theta_t^i = \theta^{**}$ and re-weight

$$w_t^i = \begin{cases} \dfrac{\pi(\theta_t^i)}{\pi_0(\theta_t^i)}, & \text{if } t = 1 \\ \dfrac{\pi(\theta_t^i)}{\sum_{j=1}^{N} W_{t1}^j Kt(\theta_t^i|\theta_{t-1}^j)}, & \text{if } t > 1 \end{cases}$$

8: **end for**

9: Normalise the weights $W_t^i = \frac{w_t^i}{\sum_{j=1}^{N}} w_t^j$ for $i = 1, \ldots, N$.

10: Update the tuning parameters of K_{t+1} using the set of particles $\{\theta_t^i, W_t^i\}_{i=1}^{N}$.

11: **end for**

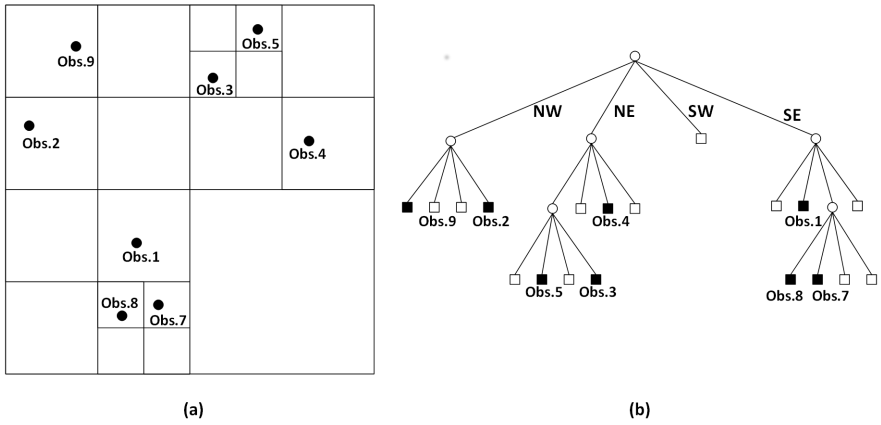


Figure 2.2: a) Data points by spatial disposition and b) the equivalent quadtree representation

the distance from the query point to each point in the observations one by one. This is computationally expensive. Spatial indexing reduces multidimensional data into a simplified form which can be used for much efficient querying. There are three major types of spatial queriesrange query, spherical query and nearest neighbor query. Since the spherical query is commonly required in a data assimilation implementation, we concentrate on this type of query in this thesis.

2.2.1 Quadtrees

A quadtree is a hierarchical spatial tree data structure that uses a recursive decomposition of two-dimensional space [45]. A region quadtree represents a space in two dimensions by decomposing the region into four sub-regions of equal size (See Figure 2.2). Each internal node can have four child nodes of the four sub-regions corresponding to northwest, northeast, southwest, and southeast quadrants. Leaf nodes are either empty (i.e., white) or contain a data point (i.e., black). Each non-leaf node has at least two descendant leaf nodes that contain data points.

Data insertion

A new record or data point (i.e., an observation) can be inserted into a region quadtree by recursively determining which node the point occupies until a leaf is found. If the leaf contains a data point, then its corresponding region must be subdivided repeatedly until these two points no longer occupy the same sub-region. The shape of a region quadtree is independent of the order of insertion of data points.

Proximity search with spherical Query

The quadtree data structure is used for proximity search with spherical querysuch as asking all observation records within 10 km of a particular grid point in a system model of data assimilation. Required data points (i.e., observations) can be found by recursively descending the tree by determining which sub-region intersects the sphere.

Parallelization

The quadtree data structure is considered to be suitable for parallel and distributed implementation. Although load balancing is one of the crucial issues in distributed processing, a simple input partitioning method could be equally applied to distribute quadtree processing. Parallel processing of quadtrees using different architectures is discussed in a number of studies [4, 8, 25, 29]. Quadtrees are hierarchical data structures that are used for space partitioning. They were first proposed by Klinger [26]. In his proposal Klinger defines quadtrees as pointer based structures used for space decomposition. The space decomposition is done by recursively partitioning the space into four regions (leaves) until a given height of the quadtree called the resolution r, is reached. The quadrants are often labeled as SW, SE, NW and NE. Searching for data is achieved by

traversing the quadtree. For example, given data points which are spatially distributed as in Figure 2.2(a) and also that the number of points that can be assigned to each leaf or bucket of the quadtree is one, the resulting region quadtree is illustrated in Figure 2.2 (b). The decomposition is done by recursively dividing the region into four equal parts until only one data point resides in each region. Leaves that have data are highlighted as black leaves. Further discussion on quadtrees can be found in Samets book [45].

Pointer based quadtree representations appear most often in literature as base representations for describing quadtree algorithms. Pointer-based quadtree representations maintain the explicit tree structure whereas linear representation replaces the tree structure with a sorted linear list containing only the leaf nodes from the original tree [50]. Pointer-based quadtrees are often used in applications where the entire tree can be maintained in main memory. An alternative form of quadtrees called Linear Quadtrees was proposed by Gargantini [11], Abel and Smith [1], Oliver and Wiseman [41] and Woodwark [64]. Linear quadtrees are better suited for applications which use quadtrees that demand large memory and hence maintained in disk files with partial loads into main memory when needed [10]. Even though there are two approaches for quadtree representation which appear to differ, algorithms yielding a particular run time complexity for one representation can usually be transformed to an equally efficient form for the other representation [11].

2.2.2 Linear quadtrees

Linear quadtrees are also ideal for situations were a persistent form of the quadtree is stored on disc. In the Linear representation the spatial address of the leaf also called the locational code is stored together with the level of the leaf as an ordered pair (n, l),

where n is the *locational code* and l is the level or height of the leaf node. For efficient computation of locational codes, dilated integers offer a plausible solution.

2.2.3 Dilated integers arithmetic for *O(1)* basic operations

In the Linear representation of quadtrees, Gargantini [11] highlights that the locational code can be viewed as an interleaved coordinate $pair(x, y)$, where $x, y \in 0, 1, \ldots 2r$. The locational code is then

$$n = y_{r-1}x_{r-1} \ldots y_1x_1y_0x_0 \tag{2.6}$$

for an integer having the following binary form of the coordinate representation.

$$y = y_{r-1}, \ldots, y_1, y_0, \tag{2.7a}$$

$$x = x_{r-1}, \ldots, x_1, x_0 \text{ and } x, y \in 0, 1 \tag{2.7b}$$

Using this representation Schrack proposes a concept of dilated integers and defines a dilated integer as an integer of a fixed length for which only the bits in the odd positions of its binary representation are significant [50]. For the purpose of our discussion we focus on encoding of locational codes into normalized dilated integers and also addition/translation of dilated integers. Further detail can be found in Schrack's paper [50].

Encoding of dilated integers

Given an integer k whose binary representation is $k = k_r, k_{r-1}, \ldots, k_1, k_0$ where $k_i \in 0, 1$, it's dilated integer form can be obtained through a dilation operation

$$qdilate(k) = kd \tag{2.8}$$

and

$$kd = k_r, g_{r-1}, k_{r-1}, \ldots, g_1, k_1, g_0, k_0 \text{ where } k_i, g_i \in 0, 1. \tag{2.9}$$

A dilated integer is said to be a normalized dilated integer if all $g_i = 0$.

Locational codes

Using the normalized dilated integer form it becomes possible to encode locational codes and perform basic addition and subtraction on them in *O(1)* time. For this Shrack uses the following operators: + ordinary addition of two integers,

— bitwise OR,

$\wedge$ bitwise AND,

$\ll n$ left shift n times,

$\gg n$ right shift n times.

Given a locational code $(x, y) \rightarrow n$, and an operation $Loc(x, y) = n$

$n = qdilate(x)|(qdilate(y) \ll 1)$

Addition/translation of dilated integers

In order to perform addition Shrack's algorithm utilizes two binary constants also referred to as masks t_x and t_y. The masks are basically used to normalize the result of the addition of two normalized dilated integers.

$t_x = 01, \ldots, 01, 01$ which is basically '01' repeated r times and

$t_y = t_x \ll 1$. Using these two masks addition/translation of a locational code of a quadtree denoted as $\oplus_q$ becomes

$m_q = n_q \oplus_q \Delta n_i = (((n_q|t_y) + (\Delta n_i \wedge t_x)) \wedge t_x)|(((n_q|t_x) + (\Delta n_i \wedge t_y)) \wedge t_y)$

where n_q is the locational code to be translated and Δn_i is the amount of translation.

m_q is the resulting locational code. In the rest of the thesis we adopt the notation used in Schrack's paper.

Prediction of multi-lineage diseases is very important. However most studies done focus on single lineage disease estimation and as a result use filtering techniques such as particle filters. From the complexity of the models discussed in this Chapter it is clear that a framework for estimating complex dynamics of diseases is needed. In addition simulation of such diseases is quite expensive computationally. This problem can be sorted out by running the simulations in parallel. This makes the SMC ABC algorithm ideal for extension in the needed framework. Furthermore the need for alleviating the bottleneck related to spatial data IO is evident. We thus discuss proposed solutions to these problems in the remaining chapters.

Chapter3

Approximate Bayesian Computation Inference Procedure with Application to Influenza

In this chapter, we propose an approach for estimating the dominant lineage of influenza B. We propose a mathematical model and an approximate Bayesian computation procedure for learning the dynamics of influenza B. Like many other models that show complex and intractable likelihoods, the prediction of the lineage dynamics of influenza B for the next season is one of the most difficult obstacles for constructing an appropriate influenza trivalent vaccine. Seasonal fluctuation of transmissibility and epidemiological interference between the two major influenza B lineages make the lineage dynamics complicated. This renders the commonly used data assimilation techniques like particle filters and Kalman filters inapplicable. To overcome this challenge, we discuss a likelihood free procedure for epidemic estimation in this Chapter. We then proceed to validate our procedure and model by applying it on a real world problem. We construct the parsimonious model describing the lineage dynamics of influenza B while taking into account seasonal fluctuation of transmissibility and epidemiological interference. We then use

our proposed procedure to learn parameter states and use these to predict the dominant lineage for the following year.

3.1 Proposed procedure

Multi-lineage disease epidemic models are often complex. As a result, model-based inference is complicated to implement due to the difficulty of obtaining an analytical solution for the likelihood function. In such cases ABC gives us a good approximation of the posterior distribution. To do this we need to develop a mathematical model for running simulations for the ABC. We propose using individual based models (IBM) as they are better capable of capturing complex dynamics. IBMs allow for simulation of epidemics taking into account individual features and interactions. Using SMC ABC and the IBM we learn the parameters that describe the epidemic. We then use these parameters to predict the most favorable time for the next cycle of the epidemic to start. We then run simulations with learnt parameters and introduce newly infecteds based on the predicted time of emergence. For an outline of how our procedure works see Figure 3.1.

3.1.1 Stochastic individual based mathematical model

To model a multi-lineage epidemic in our inference procedure we use an individual based model in which each individual is assigned an epidemiological state at each time step t. The epidemiological state of an individual at a particular point in time t is defined as

$$H_{n,t} = [D_{n,1,t} D_{n,2,t} \cdots D_{n,l,t}] \tag{3.1}$$

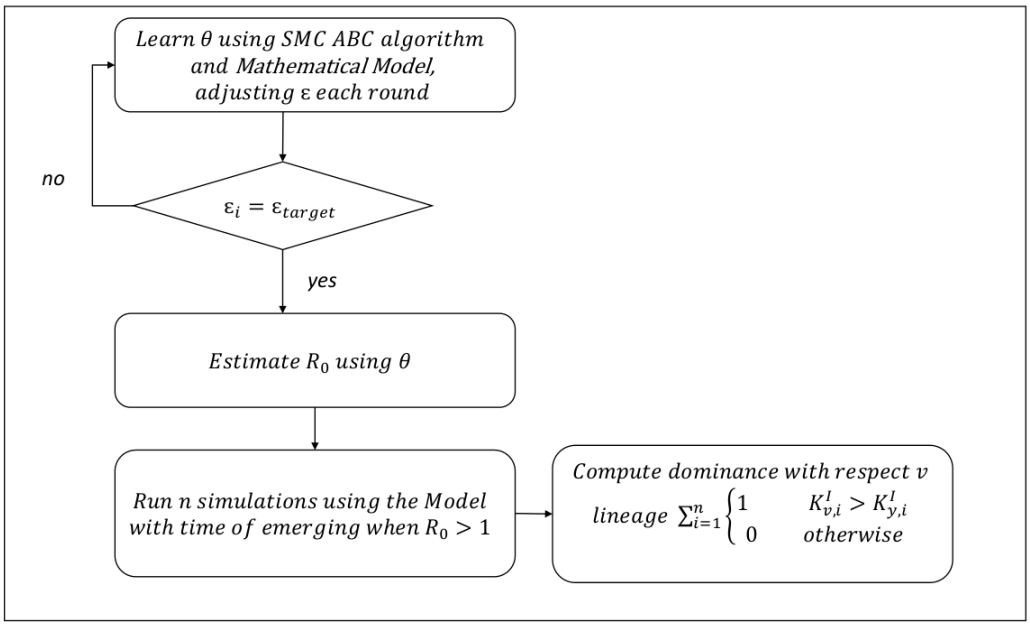


Figure 3.1: Proposed procedure for epidemic estimation

where n identifies an individual and l is the total number of lineages. Therefore at a particular point in time t, an individual has state $H_{n,t} \in \mathbb{R}^{1\times l}$ and $n < N(t)$ the total population.

The state of the entire population N at a time t is represented as

$$N_t = \begin{bmatrix} H_{1,t} \\ H_{2,t} \\ \vdots \\ H_{n,t} \end{bmatrix} = \begin{bmatrix} D_{1,1,t} & D_{1,2,t} & \cdots & D_{1,l,t} \\ D_{2,1,t} & D_{2,2,t} & \cdots & D_{2,l,t} \\ \vdots & \vdots & \ddots & \vdots \\ D_{n,1,t} & D_{n,2,t} & \cdots & D_{n,l,t} \end{bmatrix} \tag{3.2}$$

In this thesis we applied our model to influenza B which has two lineages Victoria and Yamagata and therefore $l = 2$. Based on the natural history of influenza B, we classified the host population into four classes by infection state against each lineage; susceptible S, exposed E, infectious I, and recovered R (See Figure3.1.1). A total of $4^2 = 16$ infection states were considered. We denote an individuals state with respect to Victoria as $D_{n,v,t} \in$

$\{S_v, E_v, I_v, R_v\}$ and state with respect to Yamagata as $D_{n,y,t} \in \{S_y, E_y, I_y, R_y\}$. For example, individual number 3 infected with Yamagata and susceptible to Victoria at time step 4 will be denoted as $H_{3,4} = [S_v I_y]$.

We represent the population state as

$$N_t = \begin{bmatrix} H_{1,t} \\ H_{2,t} \\ \vdots \\ H_{n,t} \end{bmatrix} = \begin{bmatrix} D_{1,v,t} & D_{1,y,t} \\ D_{2,v,t} & D_{2,y,t} \\ \vdots & \vdots \\ D_{n,v,t} & D_{n,y,t} \end{bmatrix} \tag{3.3}$$

Let $K^c_{l,t}$ denote the total number of people in a particular state with respect to a lineage and state of a lineage be $k_l \in \{S_v, E_v, I_v, R_v, S_y, E_y, I_y, R_y\}$. Then,

$$K^c_{l,t} = \sum_{n=1}^{N(t)} L_{n,l,t} \tag{3.4}$$

where,

$$L_{n,l,t} = \begin{cases} 1, & \text{if } D_{n,l,t} = k_l \\ 0, & \text{Otherwise} \end{cases} \tag{3.5}$$

3.1.2 State transition algorithm

We described the transmission process of the Yamagata lineage and the Victoria lineage using the compartmental SEIR model and Equation 3.6.

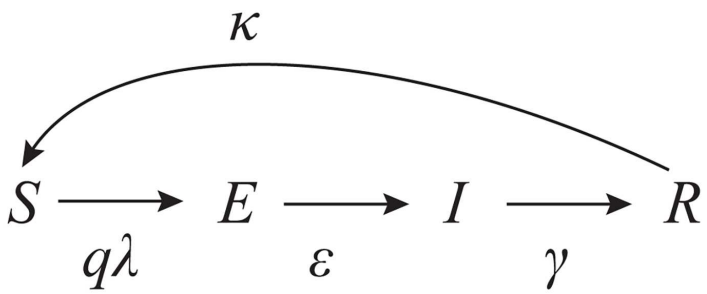
$$\frac{\delta S_v S_y}{\delta t} = \kappa_v R_v S_y + \kappa_y S_v R_y - \lambda_v S_v S_y - \lambda_y S_v S_y \tag{3.6a}$$

$$\frac{\delta S_v E_y}{\delta t} = \kappa_v R_v E_y + \lambda_y S_v S_y - \lambda_v S_v E_y - \epsilon_y * S_v E_y \tag{3.6b}$$

$$\frac{\delta S_v I_y}{\delta t} = \kappa_v R_v I_y + \epsilon_y S_v E_y - \lambda_v S_v I_y - \gamma_y S_v I_y \tag{3.6c}$$

$$\frac{\delta S_v R_y}{\delta t} = \kappa_v R_v R_y + \gamma_y S_v I_y - \alpha_y \lambda_v S_v R_y I_v - \kappa_y S_v R_y \tag{3.6d}$$

(a) Single strain model structure



(b) Full model structure

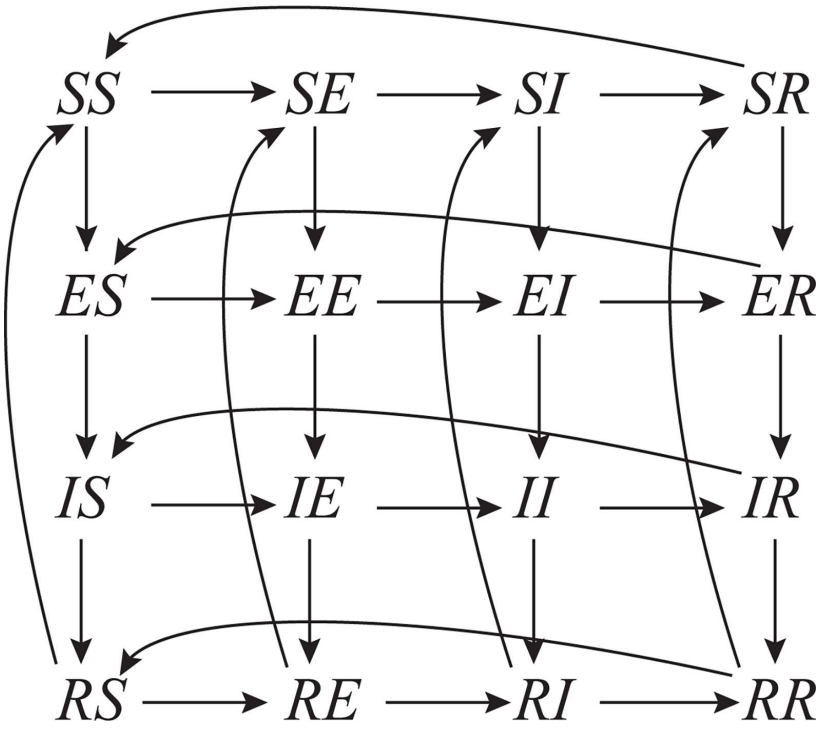


Figure 3.2: State transition for the Yamagata and Victoria lineages

$$\frac{\delta E_v S_y}{\delta t} = \kappa_y E_v R_y + \lambda_v S_v S_y - \lambda_y E_v S_y - \epsilon_v E_v S_y \tag{3.6e}$$

$$\frac{\delta E_v E_y}{\delta t} = \lambda_y E_v S_y + \lambda_v S_v E_y - \epsilon_v E_v E_y - \epsilon_y E_v E_y \tag{3.6f}$$

$$\frac{\delta E_v I_y}{\delta t} = \epsilon_y E_v E_y + \lambda_v S_v I_y - \epsilon_v E_v I_y - \gamma_y E_v I_y \tag{3.6g}$$

$$\frac{\delta E_v R_y}{\delta t} = \alpha_y \lambda_v S_v R_y + \gamma_y E_v I_y - \epsilon_v E_v R_y - \kappa_y E_v R_y \tag{3.6h}$$

$$\frac{\delta I_v S_y}{\delta t} = \kappa_y I_v R_y + \epsilon_v E_v S_y - \lambda_y I_v S_y - \gamma_v I_v S_y \tag{3.6i}$$

$$\frac{\delta I_v E_y}{\delta t} = \lambda_y I_v S_y + \epsilon_v E_v E_y - \epsilon_y I_v E_y - \gamma_v I_v E_y \tag{3.6j}$$

$$\frac{\delta I_v I_y}{\delta t} = \epsilon_y I_v E_y + \epsilon_v E_v I_y - \gamma_y I_v I_y - \gamma_v I_v I_y \tag{3.6k}$$

$$\frac{\delta I_v R_y}{\delta t} = \gamma_y I_v I_y + \epsilon_v E_v R_y - \gamma_v I_v R_y - \kappa_y I_v R_y \tag{3.6l}$$

$$\frac{\delta R_v S_y}{\delta t} = \kappa_y R_v R_y + \gamma_v I_v S_y - \alpha_v \lambda_y R_v S_y - \kappa_v R_v S_y \tag{3.6m}$$

$$\frac{\delta R_v E_y}{\delta t} = \alpha_v \lambda_y R_v S_y + \gamma_v I_v E_y - \epsilon_y R_v E_y - \kappa_v R_v E_y \tag{3.6n}$$

$$\frac{\delta R_v I_y}{\delta t} = \epsilon_y R_v E_y + \gamma_v I_v I_y - \gamma_y R_v I_y - \kappa_v R_v I_y \tag{3.6o}$$

$$\frac{\delta R_v R_y}{\delta t} = \gamma_y R_v I_y + \gamma_v I_v R_y - \kappa_y R_v R_y - \kappa_v R_v R_y \tag{3.6p}$$

The infection probability of a susceptible host is the product of the susceptibility of host q and the force of infection, $q\lambda$. Susceptibility of host q depends on infection states against both lineages as described below,

$$q_{victoria} = \begin{cases} 1, & \text{if } H_{n,t} \in \{S_v S_y, S_v E_y, S_v I_y\}, \\ \alpha_{Yamagata \to Victoria}, & \text{If } H_{n,t} = S_v R_y, \\ 0, & \text{otherwise} \end{cases}$$
$$q_{yamagata} = \begin{cases} 1, & \text{if } H_{n,t} \in \{S_v S_y, E_v S_y, I_v S_y\}, \\ \alpha_{Victoria \to Yamagata}, & \text{If } H_{n,t} = R_v S_y, \\ 0, & \text{otherwise} \end{cases} \tag{3.7}$$

For example, the susceptibility against Victoria is 1 if the host is susceptible to Victoria (the infection state for Victoria is S_v) and does not have any immunity against Yamagata (the infection state for Yamagata is S_y or E_y or I_y). If the host is susceptible to Victoria and has immunity against Yamagata ($S_v R_y$), the susceptibility to Victoria decreases to by cross-immune reaction. The force of infection at time t, λ_t, is determined by the number of infected hosts $K^I_{l,t}$ and the specific humidity h_t at time t as follows,

$$\lambda_{l,t} = \gamma_l[1 + exp(a_l - b_l h_t)]\frac{K^I_{l,t}}{N}, l \in \{Victoria, Yamagata\} \tag{3.8}$$

Here N denotes the host population size, *lambda* is the lineage specific recovery rate, the term $1 + exp(a_n - b_n h(t))$ describes the transmission coefficient determined by humidity, where h is specific humidity, and a and b are lineage specific parameters. We followed Shaman et al [48, 49] regarding the model of the relationship between specific humidity and transmissibility. We extrapolated $h(t)$ using the daily observed data of specific humidity in Tokyo, Japan, collected by the Japan Meteorological Agency (http://www.jma.go.jp/jma/indexe.html). $I_n(t)$ denotes the number of infected hosts with lineage n, for example, $I_{victoria}(t) = I_v S_y(t) + I_v E_y(t) + I_v I_y(t) + I_v W_y(t) + I_v R_y(t)$. The host infection state becomes E after infection, and the host obtains infectiousness

Algorithm 6 Infection state transition algorithm

1: **for** t=1 ... T **do**

2: $\lambda_{l,t} = \gamma[1 + exp(a_l - b_l h_t)]\frac{K^I_{l,t}}{N}, l \in \{Victoria, Yamagata\}$

3: **for** $j = 1 \ldots N$ **do**

$$prob = \begin{cases} \lambda_{l,t} \times q_l(\text{computed with equation 3.7} \quad) & \text{If } D_{j,l,i} = S_l \\ \epsilon & \text{If } D_{j,l,i} = E_l \\ \gamma & \text{If } D_{j,l,i} = I_l \\ \kappa & \text{If } D_{j,l,i} = R_l \end{cases}$$

4: **if** $uniform(0, 1) < prob$ **then**

$$D_{j,l,t+1} = \begin{cases} E_l & \text{If } D_{j,l,i} = S_l \\ I_l & \text{If } D_{j,l,i} = E_l \\ R_l & \text{If } D_{j,l,i} = I_l \\ S_l & \text{If } D_{j,l,i} = R_l \end{cases}$$

5: **else**

6: $D_{j,l,t+1} = D_{j,l,t}$

7: **end if**

8: **end for**

9: **end for**

and the infection state becomes I with probability ϵ. I_n recovers with probability λ and obtains immunity. The immune response wanes due to the evolution of antigenicity. The emergence of new distinct lineages from existing lineages is not observed for a long time [61]. We assume that the evolutionary dynamics of influenza B within the same lineage is stable and the probability of waning immunity is constant over time, κ. The parameters ($\alpha, a, b, \epsilon, \lambda$, and κ) are lineage specific.

3.1.3 Parallelization of SMC ABC algorithm

To reduce the amount of time required to do the prediction we take advantage of SMC ABC algorithm. We run multiple simulations in parallel. The simulation forms the most expensive part of the learning. Thus one process acts as the master and distributes the work to slaves. i.e the simulations. It then receives the results from the slaves and aggregates them to finish off the rest of the SMC ABC algorithm.

3.1.4 Prediction of time of emergence

From the parameters learnt from the SMC ABC we proceed to predict the most favorable time of emergence for the next cycle. In our model, $R_{0,n}$ can be described by 3.9:

$$R_{0,Victoria}(t) = \frac{SS + \alpha_{Yamagata \to Victoria} SR}{N} \times \int_{\tau=t}^{\infty} \gamma[1 + exp(a_{Victoria} - b_{Victoria} h(\tau))] exp(1 - \gamma\tau) d\tau \tag{3.9a}$$

$$R_{0,Yamagata}(t) = \frac{SS + \alpha_{Victoria \to Yamagata} RS}{N} \times \int_{\tau=t}^{\infty} \gamma[1 + exp(a_{Yamagata} - b_{Yamagata} h(\tau))] exp(1 - \gamma\tau) d\tau \tag{3.9b}$$

3.1.5 Prediction of dominant lineage

Using the posterior distributions obtained by ABC, we simulate IBM for one epidemic season and compare the results with empirical data of lineage specific confirmed cases. To measure the accuracy of the prediction, we conducted IBM several times and counted the number of simulations that showed the same dominant lineage as the empirical data. The average specific humidity at specific time points over the epidemic season was used for prediction. For prediction we consider three scenarios,

i predict using empirical data for the emergence dates of lineages,

ii predict without using empirical data for the emergence dates of any lineage, and

iii estimate the emergence dates of both lineages and predict the dominant lineage with the estimated emergence dates.

In scenario ii), we simulated IBM while varying the emergence timing of both lineages from the beginning to the end of the epidemic season. Regarding iii), we assume the emergence timing for a lineage is equivalent with the time when the lineage specific basic reproduction number $R_{0,n}$ exceeds one(See Equation 3.9).

3.2 Experimental results on application to prediction of influenza B dominant lineage

3.2.1 Data

We analyzed the weekly reports of the number of cases of human influenza B virus in Japan from the 2010-2011 season to the 2015-2016 season, collected by the National Institute of Infectious Diseases, Japan (http://www.nih.go.jp/niid/en/influenza-e.html).

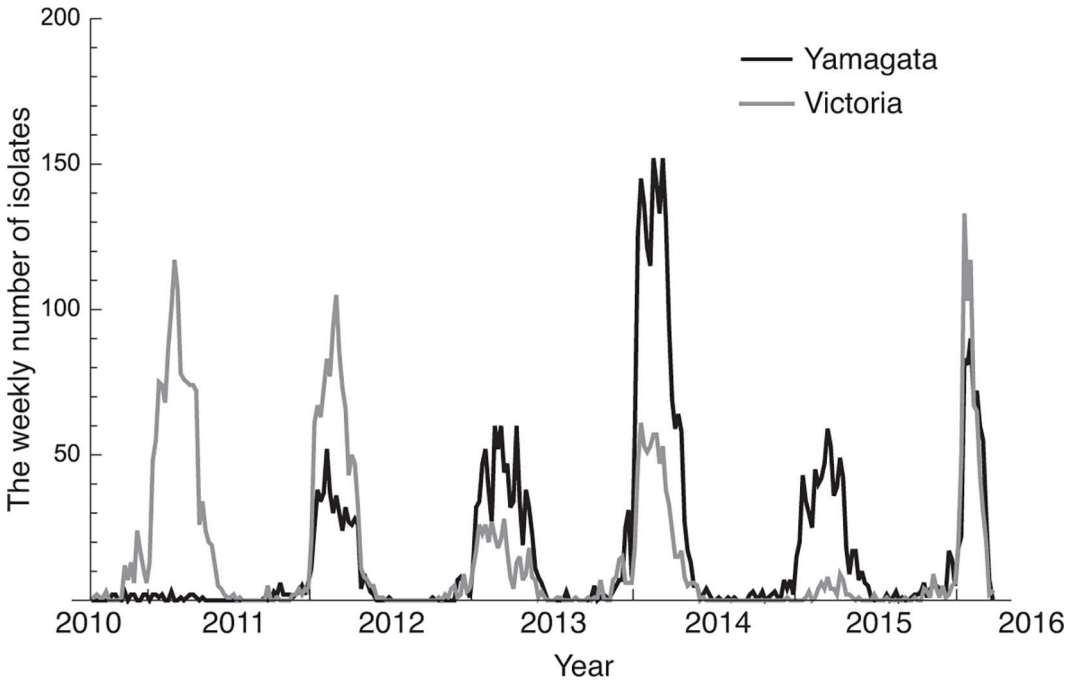


Figure 3.3: Weekly reports of the number of cases of human influenza B virus in Japan from the 2010-2011 season to the 2015-2016

The following analyses are based on the data which we accessed on 12th April 2016. Cases where the lineage was not available were excluded.

We estimate the timing when $R_{0,n}$ exceeds one using estimated parameters.

3.2.2 Method

We employed individual-based Monte Carlo simulation (IBM) with host populations of 10,000 where I_{sim} denotes the simulation result of the number of infected individual and I_{obs} denotes the field data. p is the parameter for the adjustment of the population size. The parameter sets were accepted when the distance, D, is smaller than 0.44. We assume that the sampling probability of cases for laboratory testing and the host population size are constant over time and confirmed cases are proportional to the number of infected hosts. We set the prior distributions as uniform distributions for all parameters, the

ranges of the priors are [0, 1] for $\alpha_{Victoia \to Yamagata}$, [0, 1] for $\alpha_{Victoia \to Yamagata}$, [0, 5] for $a_{Victoria}$, [0, 5] for $a_{Yamagata}$, [0, 5] for $b_{Victoria}$, [0, 5] for $b_{Yamagata}$, [0, 0.001] for $\kappa_{V}ictoria$, [0, 0.01] for $\kappa_{Yamagata}$, [0, 10] for p, [0, 10000] for S_vS_y, [0, 10000] for S_vR_y, [0, 10000] for R_vS_y, [0, 10000] for R_vR_y. We normalized S_vS_y, S_vR_y, R_vS_y, and R_vR_y as $S_vS_y + S_vR_y + R_vS_y + R_vR_y = 10000$. We introduce the infected people at the beginning of each epidemic season as the initial condition during the IBM simulation process. We defined the beginning of epidemic season as the timing when the number of isolation exceeds 7. The number of infected people in the beginning of epidemic season was adjusted by p.

To estimate these parameters we implemented Approximate Bayesian Computation (ABC) [53] using our model. Models describing the interaction between nonlinear dynamics, i.e. epidemiological interference, are difficult to solve analytically. The procedure of ABC that we conducted is,

1. i) we simulated IBM with a parameter set determined by prior distributions,

2. ii) the simulation results were compared to the time-series data of lineage specific confirmed cases and we recorded the parameter sets if the distance between the simulation results and the observed data was smaller than a threshold,

3. iii) we estimated prior distributions from the recorded parameter sets

We defined the distance between simulation results and observed data as D:

$$D = \frac{p\sum_t I_{sim}(t) - I_{obs}(t)}{\sum_t I_{obs}(t)}, \tag{3.10}$$

3.2.3 Results

Our model captured the lineage dynamics of both Victoria and Yamagata from the 2010-2011 season to the 2014-2015 season well (figure 3.4). Table 1 summarized the estimated values of parameters; with most parameters being similar between Yamagata and Victoria except b and κ. The amplitude of seasonal fluctuation of transmission rate for the Victoria lineage, b for Victoria, is higher than that for Yamagata. $\kappa_{Yamagata}$ is much higher than $\kappa_{Victoria}$, the average sojourn time until the loss of immunity is 1.15 years for Victoria and 0.079 years for Yamagata.

Using the posterior distribution of parameters in our model we predicted the dominant lineage for the 2015-2016 season. The number of isolates for the Yamagata and Victoria lineages were close during the 2015-2016 season in Japan; 694 isolates of the Yamagata lineage and 663 isolates of the Victoria lineage were reported by 12th April 2016. Although the emergence timing plays a key role in determining the dominant lineage, at this moment we do not know the future emergence timing. The average accuracy obtained by varying the emergence timings of Yamagata and Victoria is 0.614. Figure 4a shows the sensitivity analysis of the accuracy of prediction for the dominant lineage. The accuracy was improved to 0.688 if we use the actual emergence timing. We showed that understanding the emergence timing is key for the prediction of the dominant lineage. We also tried to narrow down the considerable range of the emergence timing using the lineage specific basic reproduction number $R_{0,n}$. Figure 5 shows $R_{0,n}$ during 2015-2016 season. The calendar week when the $R_{0,n}$ exceeds one is the 46th week (95% highest posterior density (HPD): 43rd - 48th week) for Victoria and the 47th week (95%HPD: 44th -50th week) for Yamagata. The actual emergence time, determined as the time when the weekly isolation number exceeds 6, is the 46th and 43rd week for Victoria and Ya-

magata, respectively. Estimated emergence timing by $R_{0,n}$ can improve the accuracy of prediction, 64.6 percent of 1000 simulation runs with estimated timing shows the correct dominant strains.

Due to the similar number of isolates between Yamagata and Victoria in the 2015-2016 season, it was difficult to determine the dominant strain. We also compared our prediction to the frequency of Victoria lineage isolates among all the influenza B cases (figure 4b). Our model can predict the frequency of lineage as well. The predicted frequency of the Victoria lineage using the observed emergence timing by $R_{0,n}$ is 0.58 (95%HPD: 0.01-0.99), the predicted frequency using estimated emergence timing by $R_{0,n}$ is 0.61 (95%HPD: 0.01-0.99), the average predicted frequency of Victoria among varied emergence timings is 0.64 (95%HPD: 0.12-0.97), and the observed frequency of Victoria was 0.51 from the field data. See Figure 3.8 and 3.9.

3.3 Discussion

In this Chapter we proposed mathematical model and a procedure for estimating the dynamics of a multi-lineage disease. Our procedure takes advantage of high performance computing by parallelising the simulations which form the most expensive part of ABC. Furthermore the use of ABC is promising as it allows for learning parameters when dealing with highly complex models. We applied our procedure to influenza B and estimated the dynamics of two major influenza B lineages using a parsimonious mathematical model. Our estimates of lineage-specific reproduction numbers agree with phylodynamic analysis [61]; the average reproduction number of Victoria is larger than that of Yamagata. The seasonal fluctuation of the reproduction number of Victoria is also larger than that of Yamagata. Our estimate of the reproduction number takes into

account both cross-reactivity of immunity between Victoria and Yamagata and waning immunity. If we misestimated these two factors, estimated values would be far from the estimate by phylodynamic analysis. Phylodynamic studies show that the time series change of the genetic diversity of Victoria lineage requires that a plausible model must take into account evolution at the strain level [3, 20, 31, 42, 51, 53] which our model did. This confirms the validity and applicability of our proposed model and procedure. In summary, we developed a parsimonious mathematical model describing the lineage dynamics of influenza B. Using the weekly number of lineage specific isolates we estimated the reproduction number, the waning rate of immunity, and the strength of cross immune reaction. We applied our model using our procedure and our prediction suggested that models taking into account epidemiological interference due to cross-immune reaction and the seasonality of transmission can predict the lineage dynamics of diseases like influenza B for the next year as long as an applicable procedure is available.

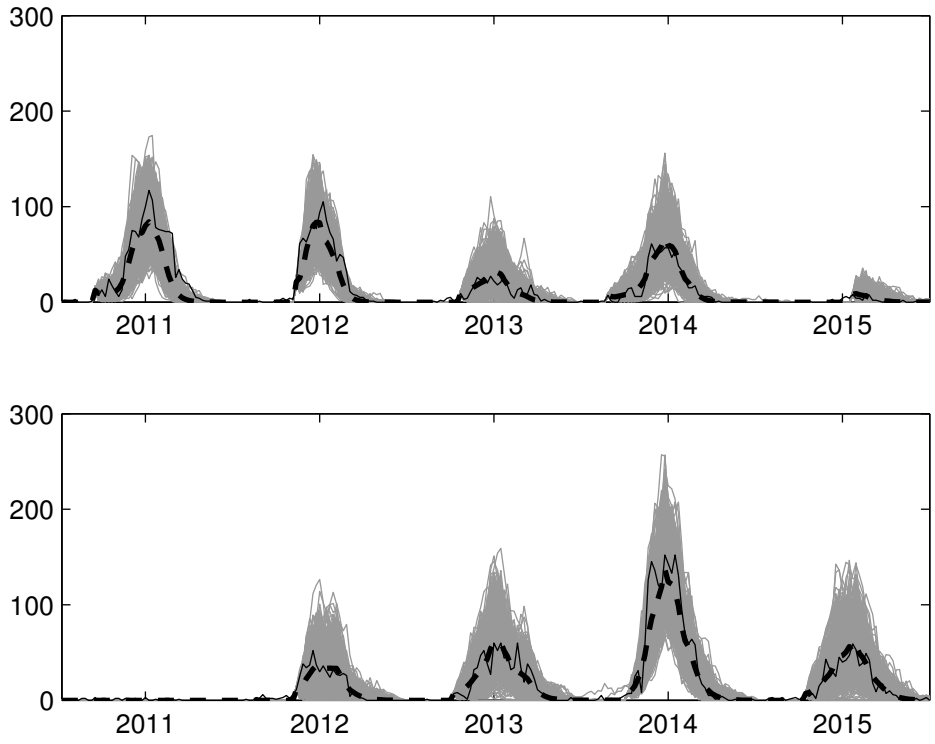


Figure 3.4: Comparison between the accepted simulations by ABC and the data from the 2010-2011 season to the 2014-2015 season. *Each gray line shows the accepted simulation run by ABC. The dashed line shows the average of the accepted simulation runs by ABC. The black line shows the data of the weekly reported number of isolates. The top panel shows the isolation of Victoria lineage and the bottom panel shows the isolation of Yamagata.*

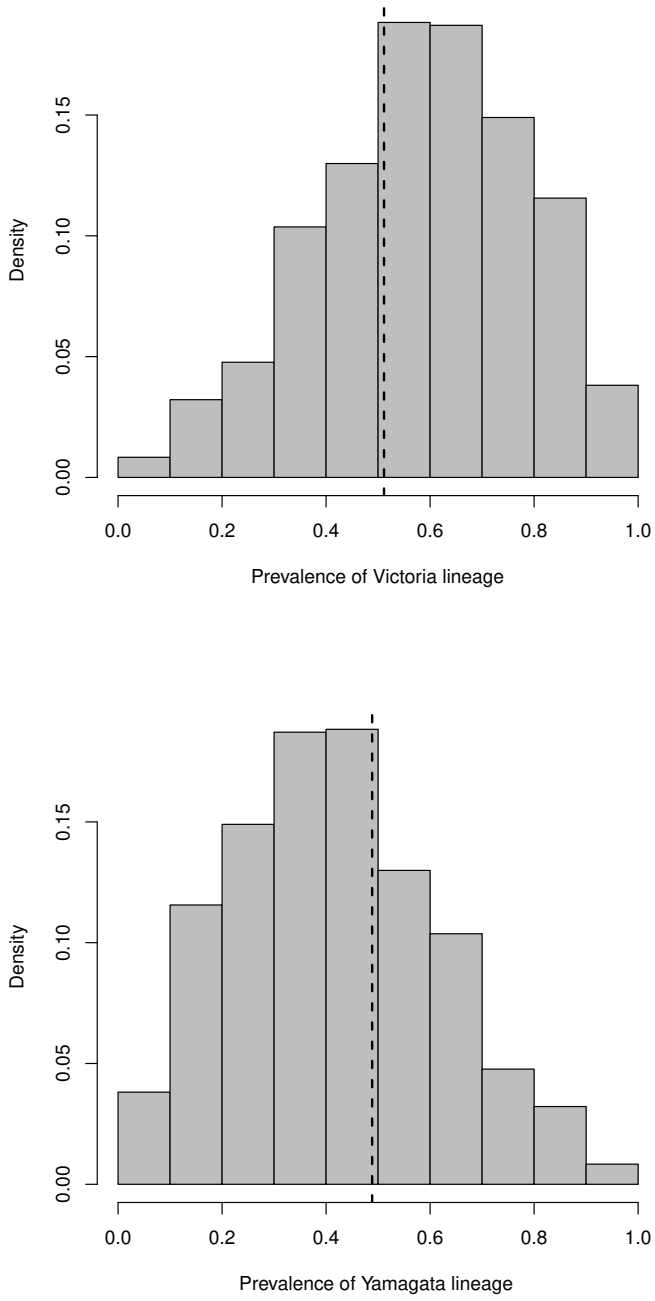


Figure 3.5: Cross-validation of our model estimation. The histogram shows the distribution of the predicted final epidemic size in the 2015-2016 season using the posteriors of parameters estimated from the weekly reported lineage-specific IBV cases from the 2010-2011 season to the 2014-2015 season. Dashed line shows the field data

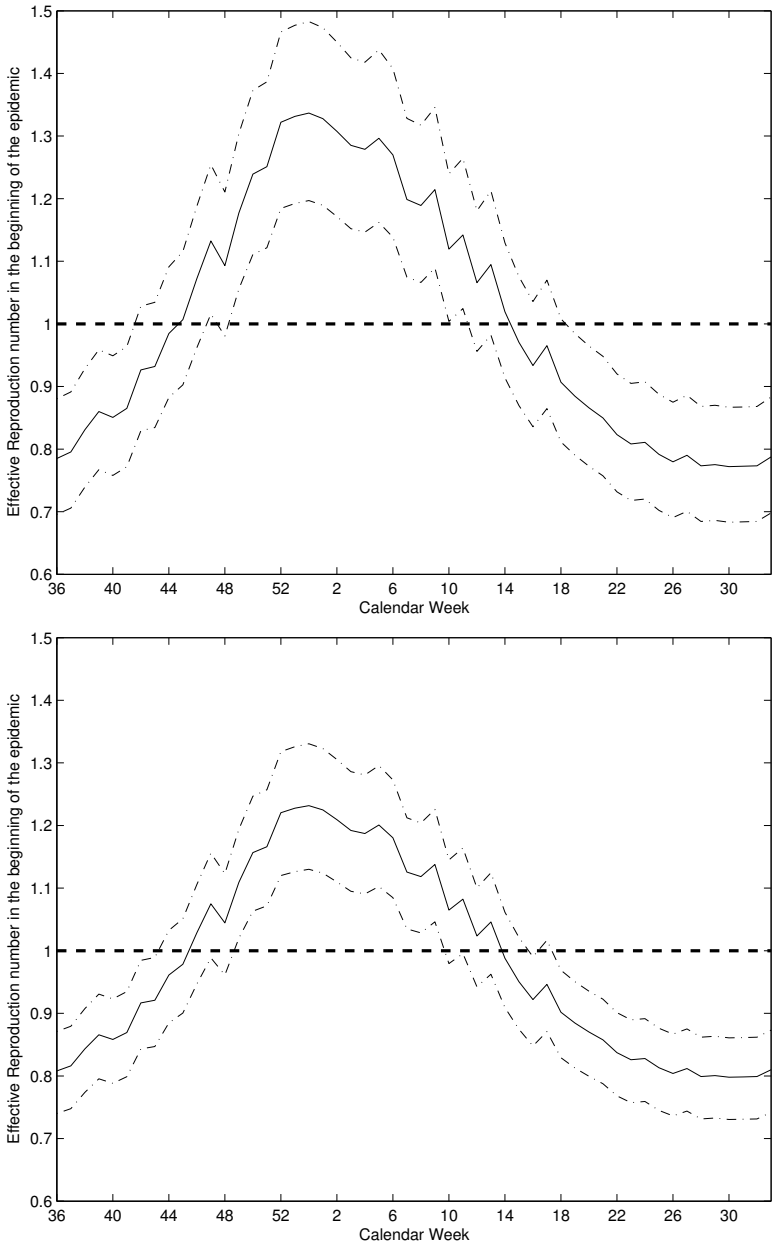


Figure 3.6: Estimated basic reproduction number at the beginning of the epidemic $R_{0,n}$ in the 2015-2016 season with varied emergence timings. (a) shows R_0,Victoria, and (b) shows R_0,Yamagata. The solid black line shows the median of the highest posterior density (HPD) and the curved dashed lines show the lower and upper bounds of 95% HPD.

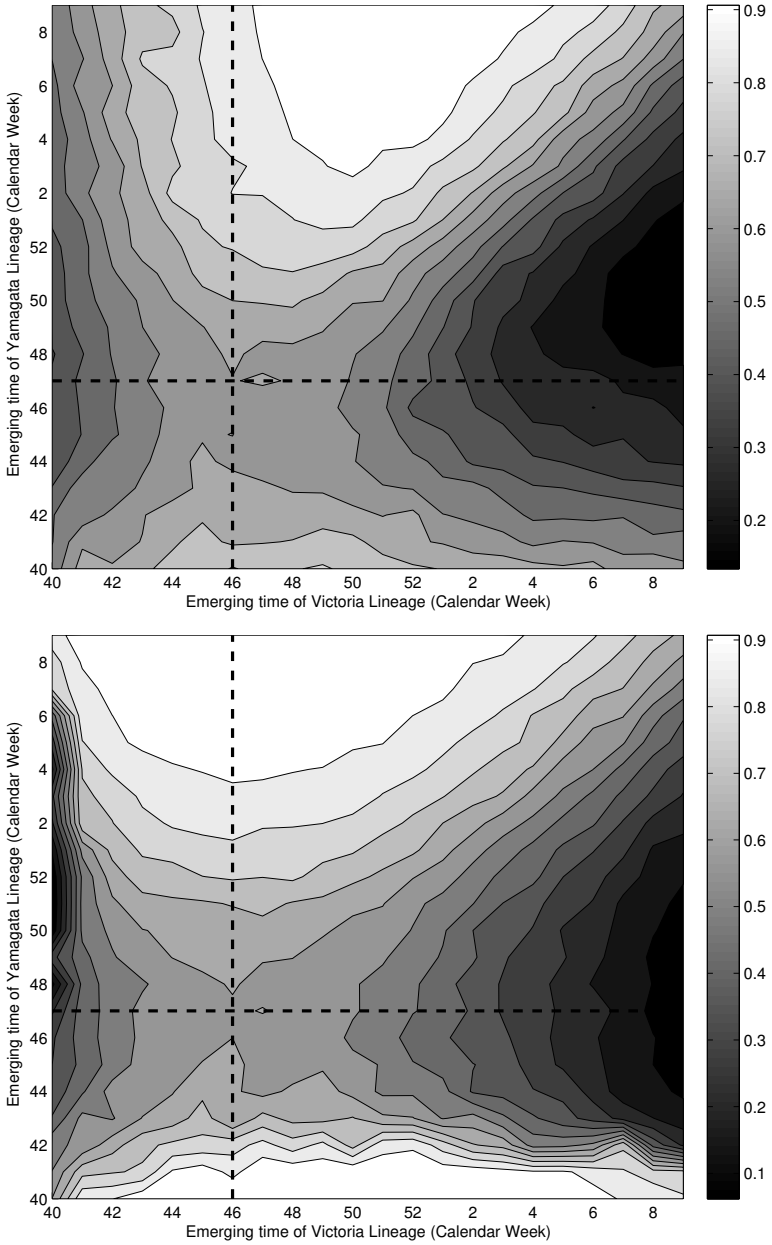


Figure 3.7: The model prediction of the epidemic in the 2015-2016 season with varied emergence timings. The straight gray lines show the actual emergence timing, and the dashed line shows the timing when the predicted $R_{0,n}$ exceeds one. (a) shows the probability that the model predicts the dominant strain is the Victoria lineage. (b) shows the model prediction of the frequency of the Victoria lineage among all isolations. The actual frequency of the Victoria lineage among all isolates in the 2015-2016 season is 0.51.

Parameters for Yamagata	$a_{Yamagata}$	$b_{Yamagata}$	$\kappa_{Yamagata}$ (year^{-1})	$\alpha_{Yamagata \to Victoria}$
Estimated values (95% HPD)	0.82 (0.77, 0.87)	0.77 (0.66, 0.87)	12.60 (3.96, 30.24)	0.51 (0.084, 0.92)
Parameters for Victoria	$a_{Victoria}$	$b_{Victoria}$	$\kappa_{Victoria}$ (year^{-1})	$\alpha_{Victoria \to Yamagata}$
Estimated values (95% HPD)	0.83 (0.74, 0.92)	1.05 (0.89, 1.20)	0.86 (0.65, 1.15)	0.62 (0.42, 0.80)

doi:10.1371/journal.pone.0166107.t001

Figure 3.8: Final estimated parameters

Data used for the estimation	**Posterior of each parameter**							
	$a_{Yamagata}$	$b_{Yamagata}$	$\kappa_{Yamagata}$	$\alpha_{Yamagata \to Victoria}$ (year^{-1})	$a_{Victoria}$	$b_{Victoria}$	$\kappa_{Victoria}$	$\alpha_{Victoria \to Yamagata}$ (year^{-1})
2010–2015	0.82 (0.77, 0.87)	0.77 (0.66, 0.87)	12.60 (3.96, 30.24)	0.51 (0.08, 0.92)	0.83 (0.74, 0.92)	1.05 (0.89, 1.20)	0.86 (0.65, 1.15)	0.62 (0.42, 0.80)
2010–2014	0.77 (0.74, 0.80)	0.89 (0.84, 0.95)	14.97 (7.3, 26.65)	0.68 (0.15, 0.99)	0.91(0.8, 1.01)	1.04 (0.98, 1.09)	0.73 (0.62, 0.84)	0.57 (0.50, 0.63)
2011–2015	0.83 (0.78, 0.88)	0.84 (0.74, 0.94)	14.60 (6.57, 27.01)	0.61 (0.10, 0.99)	1.02 (0.91, 1.12)	1.20 (1.03, 1.38)	0.73 (0.62, 0.84)	0.70 (0.55, 0.82)
2010–2013	1.00 (0.92, 1.17)	0.92 (0.85, 0.99)	8.40 (1.93, 33.95)	0.38 (0.02, 0.97)	1.00 (0.92, 1.08)	0.80 (0.65, 0.95)	0.95 (0.77, 1.20)	0.44 (0.29, 0.62)
2011–2014	0.84 (0.61, 1.10)	1.14 (0.81, 1.56)	21.90 (5.11, 36.14)	0.82 (0.34, 0.99)	1.36 (1.06, 1.70)	0.98 (0.52, 1.56)	0.36 (0.27, 0.51)	0.49 (0.24, 0.79)
2012–2015	0.88 (0.79, 0.96)	0.66 (0.54, 0.79)	5.84 (2.37,19.71)	0.95 (0.69, 0.99)	0.69 (0.57, 0.84)	1.12 (0.87, 1.40)	1.35 (0.69, 3.58)	0.84 (0.45, 0.99)

doi:10.1371/journal.pone.0166107.t002

Figure 3.9: Estimated parameters with differing length of learning period

Chapter4

Relaxing the Data Access Bottleneck of Geographic Big-data Analytics Applications Using Distributed Quadtrees

In this Chapter, we propose a distributed quadtree architecture to alleviate the I/O bottleneck in spatial big-data centric applications. We make an implementation of distributed spatial indices, specifically quadtrees, on a distributed computing system in the shared-nothing memory approach. We discuss static and dynamic partitioning and allocation strategies for data and queries across distributed nodes. Using scale-down parallel data load and search experiments with a small distributed processor system as proof-of-concept, we show that the proposed approach with a collection of small indices of distributed shared-nothing memory is more efficient than the conventional approach with a single processor with a large external index. We also observed that the proposed tree-based partitioning and assignment strategy using sampling reduces query time than other conventional partitioning strategies used in databases. We also discuss how to allocate a collection of small tree indices among distributed processors. These results suggest that the use of parallelized access to databases with spatial indexing functions

can enhance the throughput of large-scale data-centric applications.

4.1 External storage quadtree

The single in-external-storage quadtree uses a basic linear implementation of a quadtree. Using a quadtree and recursive decomposition of space each data point is assigned a unique locational code which is related to its spatial positioning. These codes are then stored persistently on external storage. The locational codes are then accessed from the persistent storage when serving query requests. To overcome this we implement our global quadtree as a distributed in-memory quadtree. We also use a tree based approach in our implementation and extend it further by using persistent processes to reduce the costs related to process re-initialization. Furthermore, we use the tree based directory to strategically redirect queries. Our approach is similar to that used by Noh and Min [9]. However while Noh and Min focus on using the quadtree for improving the time to build a quadtree, we exploit the tree partitioning approach to improve query time and use it as a directory structure for the distributed data servers. The directory is used for both record assignment and query redirection in a shared-nothing environment.

An alternative to the permanent storage approach to querying the spatial data is decomposing it into smaller partitions which can fit in a collection of computers that collectively can store the data in ram but are singly inadequate. Key to this is the use of persistent processes for managing data access. To achieve this a directory of which server is responsible for which region of the data space is required. Such a directory is used during data insertion or query servicing to determine which server the request or data should be forwarded to. The following are possible data space partitioning strategies.

4.2 Architecture for the distributed in-memory quadtrees

Fig.2 shows the basic architecture of the distributed in-memory quadtree in a shared nothing environment. The master server partitions and distributes data to the data servers and builds a tree directory based on a data partitioning strategy. The directory is then used for query redirection. Of importance is the approach for partitioning the data and also how to assign the partitions to the servers. How do we distribute set of data points $P = \{p_1, p_2 \ldots, p_n\}$ amongst a set of servers $S = \{s_1, s_2 \ldots, s_k\}$ and minimise Standard deviation of S.

4.2.1 Random partitioning algorithm

In the random partitioning strategy data point p_i is assigned to a data server s_i picked from a pool of servers at random illustrated in algorithm 7. Each server then builds a quadtree from the points that are forwarded to it by the master. The master however does not keep track of which server has which data points as this would be computationally expensive. For this reason, to perform a search query, the master server broadcasts the query to all the data nodes. Each and every data node in turn searches its quadtree and returns the set of points which satisfy a search criteria. This data partitioning strategy is not location aware. It however results in evenly sized partitions.

Algorithm 7 Random partitioning algorithm

$Input : P = \{p_1, p_2 \ldots, p_n\}, S = \{s_1, s_2 \ldots, s_k\}$

for $i = 1 \ldots n$ **do**

$s_k \leftarrow p_i$ (randomly select server s_k)

end for

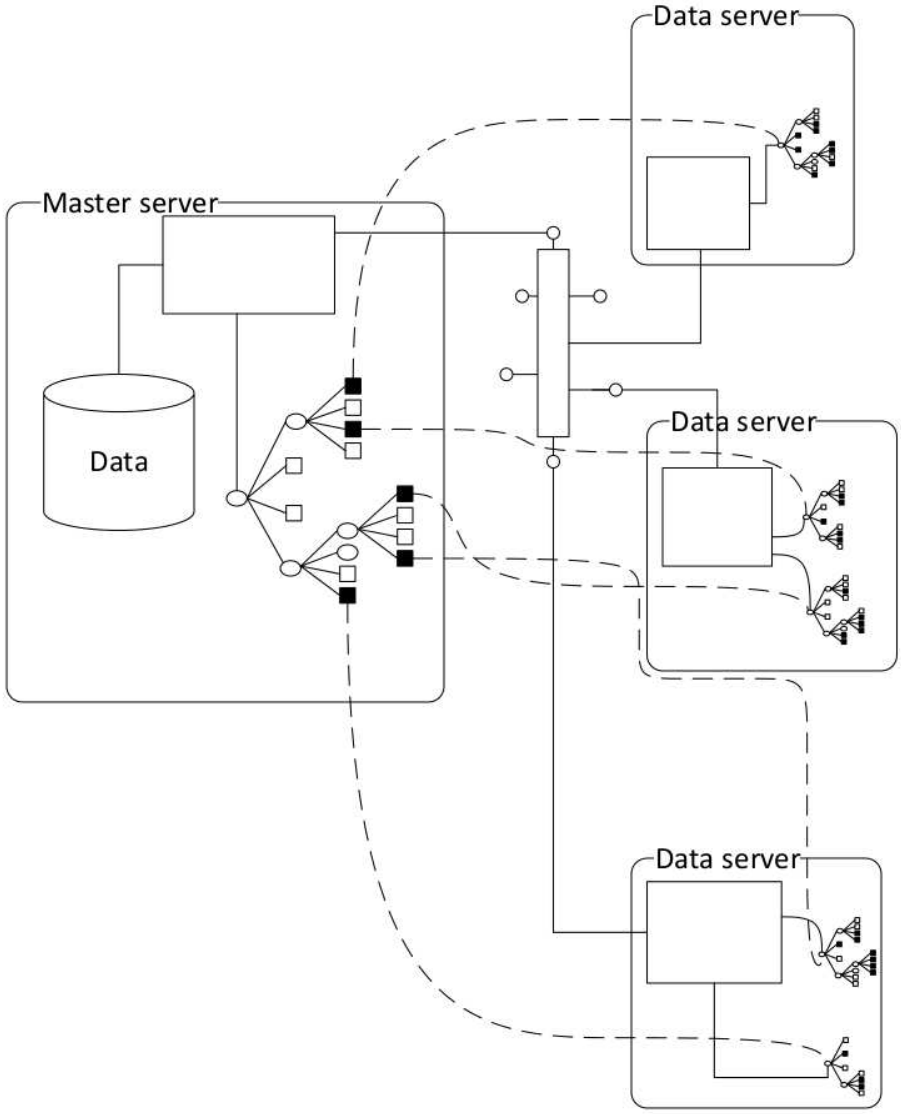


Figure 4.1: Architecture of the distributed quadtree using a tree based directory

4.2.2 Simple one-dimensional partitioning algorithm

In this partitioning approach the data is partitioned using one of its dimensions, d. Firstly the area encompassing the data points is evenly divided by either along dimension d. Each resulting sub area is then allocated to a data node by the master. To populate the quadtree the master choses a data server to forward a data point to by checking which region the point belongs to using the data points d-dimension value outlined in algorithm8. To process a query the master first checks which of the regions intersect the query range. The request is then forwarded only to the data servers whose regions intersect the query range. For this reason this data partitioning strategy is location aware. It however does not detect clusters and skewed data and can easily lead to unevenly sized partitions.

Algorithm 8 Random partitioning algorithm

$Input : P = \{p_1, p_2 \ldots, p_n\}, S = \{s_1, s_2 \ldots, s_k\}$

$interval \leftarrow max(P(d)) - min(P(d))$

for $i = 1 \ldots n$ **do**

$k = \left\lceil \frac{p_i(d) - min(P(d))}{interval} \right\rceil$

$s_k \leftarrow p_i$ (randomly select server s_k)

end for

4.2.3 Distributed quadtree based partitioning algorithm

Instead of keeping a large quadtree in the external memory as in the above approach, we propose approach partitioning the imaginary large quadtree for the whole data set into an exclusive collection of small subtrees. Then, distribute them on a loosely coupled collection of distributed processors with shared-nothing memory. To make dispatching

of queries from many clients easier, we use the upper subtree of the original tree, called the master quadtree ($Q_{|m/k|}$), as a directory for dispatch. k is the number of servers and m is a number of points sampled from P that uses only a small working memory for constructing the quadtree with node capacity $|m/k|$. We first draw a small random sampling from the whole data, and then construct a small quadtree based on this sample.

The rest of all lower subtrees,Q_i^s are called the data quadtrees, and allocated to all data servers. Since we have more data subtrees than data processors, each data server maintains more than one subtree each of which contains a partition of the data set.

Next, by associating the range and the expected sizes to each imaginary subtree, the master server allocates each subtree to one of the data servers. Using this small quadtree as the directory for dispatching, the master server can redirect each data point or query to the appropriate server having its data range by reading all data points or all queries from the source(See Algorithm 9). We construct the linear quadtree using a function consQuadTree(Size,Capacity).

Data distribution forms an important aspect when it comes to partitioning spatial data for distribution. Skewed data can lead to poor performance as most data can in the worst case be assigned to one data server. It is therefore important to take into account the distribution of the data before assigning it to data nodes. Random assignment and simple one-dimensional partitioning is oblivious of skewed data. On the other hand simple partitioning takes into account the proximity of data points but does not take into account clustered and skewed data. To partition the data taking into account both the amount and spatial distribution, tree data structures form a good candidate as they can both discover clusters as well as balancing the number of data points assigned to each processing node. We use the quadtree data structure to partition the data and also

Algorithm 9 In-Memory distributed quadtree algorithm

1: sample ← sample n_s points from P

2: Array $Q \leftarrow constQuadTree(sample, |n_s/k|)$ ▷ Construct linear quadtree with capacity n/k

3: $sort(Q)$

4: **for** i=1 ... n **do** ▷ Assign all the data points in P

5: L ← 0 and $R \leftarrow n - 1$

6: **if** $L > R$ **then**, ▷ terminate as unsuccessful

7: **else**

8: $m = \lfloor (L + R)/2 \rfloor$

9: If $Q[m] < l_p$, $L = m + 1$ and go to step 8

10: If $Q[m] > l_p$, $R = m1$ and go to step 8

11: If $Q[m] = l_p$, distributed point p_i to $Q[m]$

12: **end if**

13: **end for**

use this quadtree as a directory when processing queries.

Once the partitioning has be done, the partitions need to be assigned to data servers. Of importance during the assignment is an even distribution of records across the data servers. Packa set of data quadtrees $Q^s = Q_1^s, Q_2^s, \ldots, Q_i^s$, each with size $t_i, i = 1, 2, \ldots, n$, into identical servers s_k each of capacity C. To do this we considered two approaches:

4.2.4 Ascending-Descending partition assignment algorithm

. In this trivial approach the leaves (partitions) of the master quadtree are first sorted in order of size i.e. the total number of records they contain. The partitions are then allocated to the data servers by traversing the sorted list of partitions from either biggest to smallest (descending) or smallest to biggest (ascending). When a data server reaches its maximum capacity it is skipped and no more partitions are allocated to it.

Example

Lets assume we have 40 sample data points to allocate to servers s_1 and s_2. We would want to distribute approximately 20 data points to each server. Assuming further that we got partitions of $6, 7, 6, 12$ and 9. We begin by first sorting them in descending order giving us $12, 9, 7, 6, 6$. We then fix the order of the servers as s_1, s_2. We then traverse the fixed order of servers in alternating rounds of ascending and descending order. In the first round 12 goes to s_1 and 9 goes to $s_2, s_1 = 12$ and $s_2 = 9$. In the second round 7 goes to s_2 and 6 goes to $s_1, s_1 = 12, 6$ and $s_2 = 9, 7$. Finally we assign 6 to $s_1, s_1 = 12, 6, 6$ and $s_2 = 9, 7$.

4.2.5 Relaxed capacity first fit decreasing bin packing algorithm

The bin packing problem is one of the classical problems defined as intractable. Given an infinite amount of bins of unit size the bin packing problem tries to pack the items in as few bins as possible. There are heuristics that however offer algorithms that proffer solutions to the intractable problem. One of these being the *first fit decreasing algorithm (FFD)* which is simple to extend for our purpose and offers performance which is comparable to its competitors [27] like the *best fit decreasing algorithm* (BFD). Both algorithms are guaranteed to return 11/9 of the optimal number of bins [21]. Further detail about the bin packing algorithms can be found in [21, 27].

We draw our attention to two important technical observations about FFD performance [21]. Firstly, suppose we have n items that have been sorted in descending order of size; $Q_1^s > Q_1^s > ... > Q_k^s$. If optimal packing uses k bins, then all bins in the FFD after k have items of $size <= 1/3$. Secondly, the number of items FFD puts in bins after k is at most $k - 1$. In the case of our partition assignment requirement, k is the number of available data servers. After the bin packing algorithm has done the allocation we relax the constraint of the capacity of the bins and assign the $k - 1$ items in the bins after k to the k bins using a best fit decreasing approach i.e assign the biggest item of the $k - 1$ items to the least full bin of the k bins. Because FFD uses a decreasing order we are guaranteed that the items in bins after k are the smallest items and would thus give us an approximate near best solution to distributing the items. In addition we are also guaranteed that no two of the $k - 1$ items need to be assigned to the same bin as there are k bins and $k - 1$ items.

Example

Going back to our example given $6, 7, 6, 12$ and 9 and bin capacity 20, FFD first sorts these in decreasing order giving us $12, 9, 7, 6, 6$ and then fixes the order of bins. FFD will then proceed by placing 12 in the first bin and 9 in the second bin as it doesnt fit in the first bin. 7 would then be place in the first bin and 6 in the second bin. The last 6 doesnt fit in neither bin 1 nor bin 2 and thus would be assigned to bin 3. After this all the n items would have been processed. The optimal number of bins k in our example is 2 similar to the ascending-descending example. So with a relaxed capacity constraint we then assign the 6 from the third bin to the second bin as it is the list full. The result being bin $1 = 12, 7$ and bin $2 = 9, 6, 6$.

4.3 Experimental results

The primary goal of the proposed solution is to distribute spatial data across a set of servers in a shared nothing environment. Generally the data assimilation process accessing the data would in the real world run on a computationally superior computer. Thus we use a scale down model to try and recreate this asymmetric computational power. We chose to use Raspberry Pis as data servers and an Intel Xeon based computer for data access in order to create the computational power asymmetry. Roughly speaking, the performance of a Raspberry pi is equivalent to 0.041GFLOP and comparable to a 300MHz Pentium II of 1997-1999, while its power consumption can be estimated at 3W. An Intel Xeon X5365 CPU performs around 38 GFLOPS with power consumption of 150W. A Raspberry pi is therefore roughly speaking 1/1000 times slower in GFLOPs when compared to a server type Intel Xeon processor like the X5300 series CPU. Its power consumption is however 1/50 of power consumption in TDP when compared to

Dataset	Number of points	Size of area
WMO surface observations and Upper-air stations	13028	20,000(km) by 40,000(km)

Table 4.1: Summary of Observational Data set used

the X5300 series CPU.

4.3.1 Data

Data describing a high precision spatial distribution of all the surface and upper-air stations in operation which are used for synoptic purposes was downloaded from the World Meteorological Organization website [35]. The data has 13028 stations encompassing an area of approximately 20000(km) by 40, 000(km). For the experiments reported in this chapter the data was generated by assigning synthetic meteorological records to the actual coordinates of the observation stations to simulate observations. The synthetic meteorological data follows the format specified by CISCL research data archive for TDL U.S. and Canada surface and airways hourly observations data. Therefore even though the meteorological values are synthetic the spatial distribution as well as the size of each record follows the real world distribution.

4.3.2 Method

We used ten type B+ Raspberry Pis interconnected using a network switch. Nine of the Raspberry Pis served as data servers and one as the master server. The Raspberry Pis were of type B first generation, CPU ARM1176JZF-S, 700MHz, 512 RAM, 16KB L1 cache and 16GB SSD.

Partition Distribution and Assignment strategies experiment

In the first experiment we investigated how the size of sub trees allocated to the data servers varies with change in sub tree size. We used the *ascending-descending* and *relaxed capacity first fit bin packing* as partition assignment strategies. Each set up was repeated 10 times for sub tree sizes off 50, 100, 150 and 200. We then calculated the standard deviation of data size per data server as a measure of the difference in size of the resulting sub trees allocated to each data server.

Range query response time experiment

The second experiment focused on the monitoring of the total response time i.e. the time from the sending of the first range query to the receiving of the last response. We used an external quadtree, distributed quadtrees using random, simple one dimensional assignment and tree based partitioning. For each setup the number of queries was increased from 1 to 100,000. Each query searched for points within a 100,000 meters radius. The query points were drawn from a uniform random distribution within the data space see Table4.1. For each number of queries the experiment was repeated 10 times and the average response time of the 10 recorded.

4.3.3 Results

Partition Distribution Assignment strategies

The results show that relaxed capacity first fit bin packing has a lower standard deviation 20% smaller than the ascending-descending approach see Figure 4.4. With trees of maximum size 50, relaxed capacity first fit shows a deviation of about 10% in data allocation size. It hence demonstrates a much more even distribution of data as

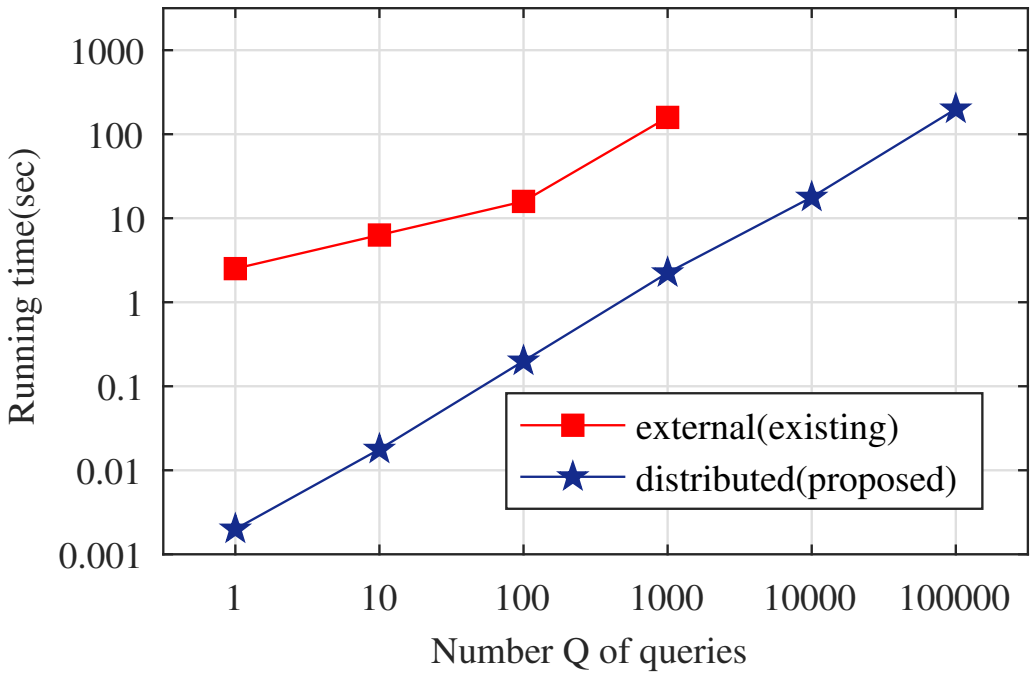


Figure 4.2: Query response time comparing external and tree based distributed quadtree compared to the ascending-descending approach.

Range query

Tree based partitioning achieves the best performance in terms of query response time when compared to simple partitioning and random partitioning strategies. Query redirecting using a tree directory is almost 10 times faster than random partitioning. It is almost twice faster than simple one dimensional partitioning. When compared to the external storage quadtree, the distributed in-memory quadtree using a tree directory performs almost a 100 times better.See Figure 4.3 and 4.2.

4.4 Discussion

Geographic big-data analytics applications are becoming more and more important. With the current development in processing power I/O is becoming a notable bottleneck.

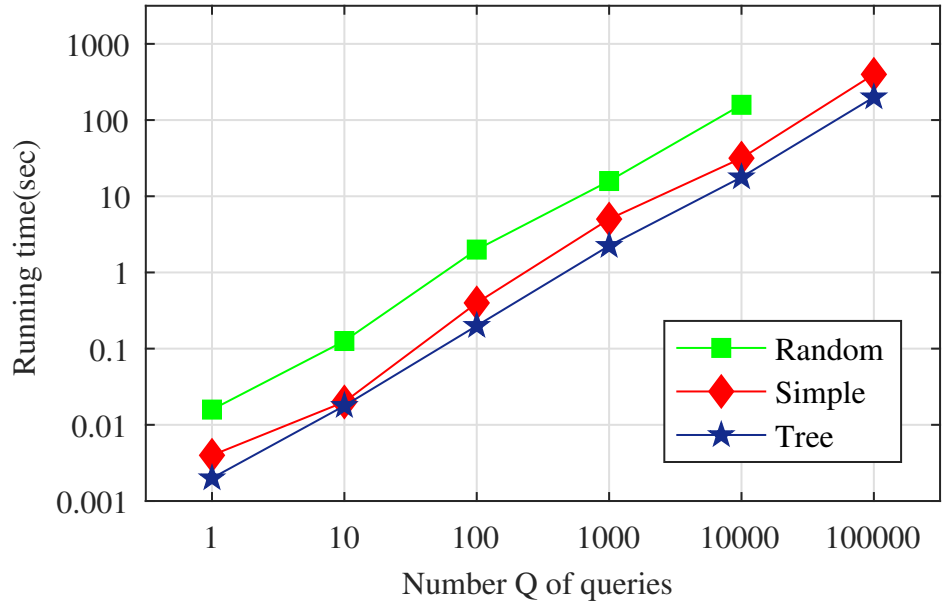


Figure 4.3: Query response time for distributed in-memory quadtrees

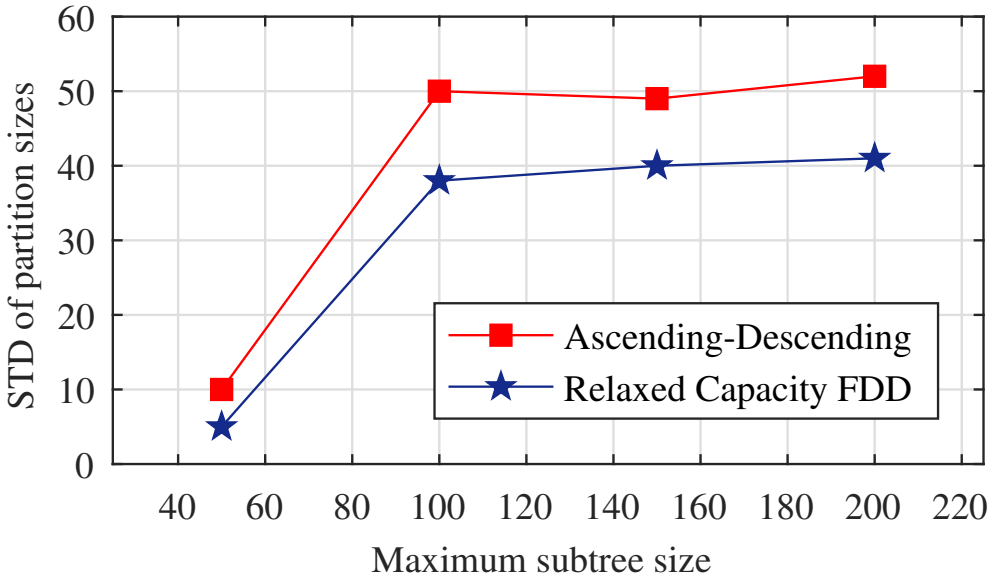


Figure 4.4: Variation in partition size with sub tree size

In this chapter we proposed a solution to the I/O problem related to spatial data that uses a distributed in-memory processing approach. The proposed architecture demonstrated that distributing a quadtree into several servers improves the query performance by more than a hundred fold. In addition we also showed that careful selection of partition assignment strategies results into much more evenly distributed data. This is very important as the quadtree is queried from memory. In addition using a master quadtree helps reduce the impact and effect of clustered and skewed data. This approach is also location aware and thus gives good performance for both data insertion and query servicing. This result is of significance as improving I/O ultimately improves the performance of a data assimilation process as demonstrated in [5, 37].

Chapter5

Speedup of Construction of Distributed Quadtrees Using Dilated Integers and Hashmaps

In this chapter, we propose an improvement to speedup construction of the distributed quadtree proposed in chapter 4. We take advantage of the static nature of the master quadtree and use hashmaps and dilated integers to speed up traversal of the directory. We successfully speedup the process of query redirection during the construction of the distributed quadtree as well as query redirection during a data retrieval process. We conduct experiments for construction and data querying and show that both construction and querying performance improves threefold when you compare the new approach to the previously proposed approach. In addition further experiments show that the proposed new approach is much less sensitive to data skewness. We conclude the Chapter with discussion of the results.

We use the architecture proposed in Chapter 4. In this architecture data is first sampled to construct a base quadtree. Each leaf of this quadtree is then assigned to a data server. The base quadtree then acts as a directory data structure for both searching

and insertion of data. The main focus of this Chapter however is to reduce the amount of time required to locate a data server. Thus we construct a linear quadtree and take advantage of the static nature of the data to do a binary search over the tree's height.

5.1 Linear quadtree construction

We construct the master quadtree using sampled data and then use this quadtree to generate a linear representation. The sample based quadtree is static in nature and thus we use the depth of the deepest leaf node as the resolution, r, of the linear quadtree. It is also important to note that unlike an ordinary quadtree where only the black nodes are stored in our situation we store all leaf nodes even the non-black ones. This is because an empty leaf node doesn't imply no data can possibly be in that location but that data is less likely to be in that location. This is a result of the construction being done using sampled data. Consequently we still have to add such leaf nodes to the directory structure and therefore all leaf nodes are added to the linear representation. We use an architecture and procedure similar to that used in Chapter 4. We use a tree partitioning strategy to partition the data and use the *relaxed capacity first fit decreasing bin packing* which is an extension of the first fit decreasing bin packing algorithm [21, 27], for partition assignment. We also use an in-memory approach for storing the data on the servers. The main focus of this Chapter is however the process for locating data servers during data insertion after partition assignment and also locating the data servers during data querying.

5.2 Locating data servers using quadtrees, dilated integers and hashmaps

We use the same approach as in Chapter 4. We begin by partitioning the imaginary large quadtree for the whole data set into an exclusive collection of small subtrees. Then, we distribute them on a loosely coupled collection of distributed processors with shared-nothing memory. To make dispatching of queries from many clients easier, we use the upper subtree of the original tree, called the master quadtree ($Q^n_{|m/k|}$), as a directory for dispatch. k is the number of servers and m is a number of points sampled from P that uses only a small working memory for constructing $Q^n_{|m/k|}$. We first draw a small random sampling from the whole data, and then construct a small quadtree based on this sample.

The rest of all lower subtrees,Q^s_i are called the data quadtrees, and allocated to all data servers. Since we have more data subtrees than data processors, each data server maintains more than one subtree each of which contains a partition of the data set.

Once the partitioning has be done, the partitions need to be assigned to data servers. Of importance during the assignment is an even distribution of records across the data servers. During insertion and querying the directory quadtree is traversed to locate the data server managing the region a point belongs to. In following section we look at how to efficiently traverse the set of data quadtrees $Q^s = Q^s_1, Q^s_2, \ldots, Q^s_i$.

5.2.1 Basic directory traversal algorithm

In the basic algorithm we store the information for the data server in an array and perform a basic binary search to locate the designated data server servicing the region the point belongs to. This algorithm therefore does a binary search over the total number of nodes in the global quadtree.

Given an array Q of data quadtree elements with locational codes $l_0, l_1, \ldots, l_{n1}$, sorted such that $l_0 \leq \ldots \leq l_{n1}$, and point to search for p having a location code l_p, Algorithm 10 uses binary search to find the index of the data server for p.

Algorithm 10 Basic directory traversal algorithm

1: L $\leftarrow 0$ and $R \leftarrow n-1$

2: **if** $L > R$ **then**, terminate as unsuccessful

3: **else**

4: $m = \lfloor (L+R)/2 \rfloor$

5: If $Q[m] < l_p$, $L = m+1$ and go to step 2.

6: If $Q[m] > l_p$, $R = m1$ and go to step 2.

7: If $Q[m] = l_p$, the search is done; return m.

8: **end if**

Once the index is returned it is then used to retrieve the data server and the data point is then forwarded to the appropriate server.

5.2.2 Improved directory traversal algorithm

In order to locate a server that a data point should belong to we begin by first computing its dilated integer form. After this we find the longest matching prefix from the linear representation of the quadtree. This is done by using a binary search over the height of the quadtree. A hashmap is used to store the leaf nodes at each height of the quadtree. Essential to the prefix matching is the definition of a mask. We define the mask as

$$m_r = 1_r, 1_{r-1}, \ldots, 1_1, 1_0 \tag{5.1}$$

which is basically 1 repeated r times where r is the height of the quadtree and apply the following binary search algorithm:

Given a sorted array Arr of n hashmaps with hashmap at array index i having all leafs at height i of the quadtree and a $point(x, y)$ we apply the following procedure to locate the server, where L and H stands for lower and upper end of data array respectively: The iterative procedure searches for the longest bit pattern matching a data server and

Algorithm 11 Proposed directory traversal algorithm

1: $result \leftarrow null, n \leftarrow Loc(x, y), L \leftarrow 0$ and $H \leftarrow r - 1$
2: **if** $L > H$ **then**
3: terminate.
4: **else**
5: $m = \lfloor (L + H)/2 \rfloor$
6: $s = n \wedge (m_r \ll 2m)$
7: key = find(s), search for s in hashmap for level m
8: **if** $key \neq null$ **then**
9: result = key, L = m
10: **if** key = null **then**
11: R = m
12: **end if**
13: **end if**
14: go to step 2
15: **end if**

then picks it as the appropriate server to forward the data point to for storage. The *find(s)* sub procedure is implemented as a hash function. After the server is located the

data is sent to that server. The server then builds a local quadtree using data points forwarded to it. The global quadtree is thus distributed across many data servers and points during subsequent searches are located using the directory or master quadtree. A similar approach is used when retrieving the data during a query session. Following the discussion in Shrack's paper it is evident that algorithms for other search queries for quadtrees can be implemented using the dilated integers approach we therefor do not proceed to implement these algorithms.

5.3 Experimental results

The main focus of this chapter is the enhancement of the procedure for locating the data servers during data insertion and also when querying the distributed quadtree. Before data is forwarded to a data server, the server serving the region the data point belongs to has to be located. This is done by traversing the master quadtree. Consequently we perform experiments to measure the improvement in performance for locating data servers when using our proposed approach for traversal and compare it to using the previous approach for traversal. The previous approach uses a binary search over the leaves of the quadtree. The proposed approach uses a binary search over the height of the quadtree.

5.3.1 Data

For the experiments we use data downloaded from the World Meteorological Organization website [35], describing the surface and upper-air stations in operation used for synoptic purposes. This is similar to the data used in Chapter 4. The data has 13028 stations encompassing an area of approximately 20,000(km) by 40,000(km). For

the experiments this Chapter, we however synthesize the data by reproducing more point following the same distribution and increasing the number to 10^9 data points.

5.3.2 Method

We use 8 servers with Intel(R) Xeon(R) CPU E7-4890 v2 @ 2.80GHz processors and 1 TB total memory for all processors in a shared nothing environment. The program source code is implemented in java programming language.

Construction time with varying number of data points

In the first experiment we investigate how both the proposed and previous approaches perform as the number of data points in the distributed quadtree increase. We measure the time required to construct the quadtree starting with 10^3 points and increase this until 10^9 points. We keep the size of the sample data and data points per server during master quadtree construction constant. By doing this we keep the quadtree structure constant but vary the amount of data points to be inserted in the global quadtree. We run each setup 10 times and record the average of the 10 runs.

Construction time with respect to depth

In the second experiment we focus on how performance varies with the depth of the directory (master) quadtree. We thus vary the average depth of the quadtree by changing the maximum number of points for each leaf of the directory quadtree while keeping the number of sampled data points constant. We use 106 data points. Thus we keep the number of data points to be inserted in the quadtree constant and just vary the structure of the master quadtree in height. We run each setup 10 times and then

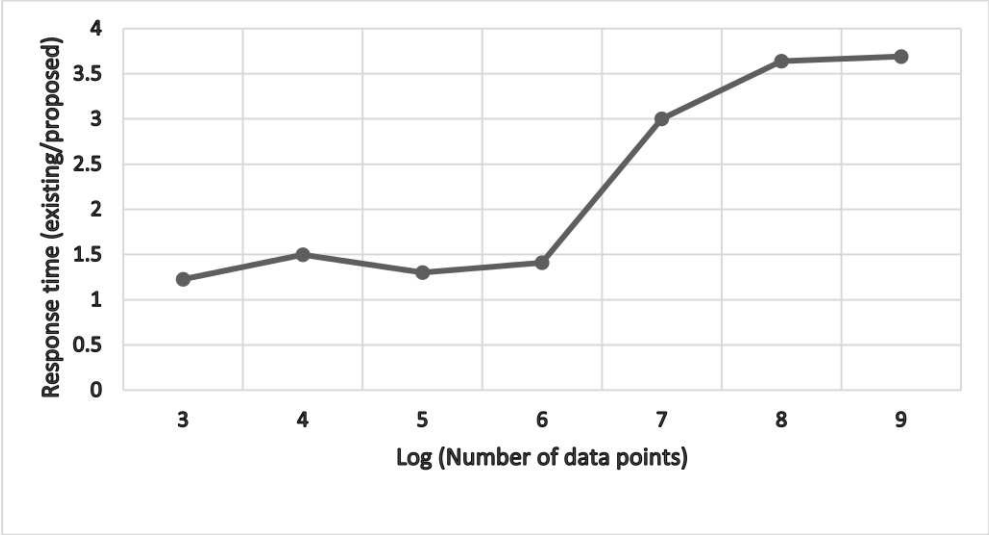


Figure 5.1: A comparison of the scaling of the construction time for the previous and proposed approach

record the average time required to construct the quadtree. Figure 3. A proportional comparison of the construction time of the previous approach to the proposed approach

5.3.3 Results

Construction time with varying number of data points

Both the proposed approach and previous approaches show similar performance when the number of data points is small. However as the number of data points increases the proposed approach starts to outperform the previous approach. Directory redirection improves more than threefold after reaching ten million data points as can be seen from Figure 5.1, which is plot of the running time of the existing approach divided by the running time of the proposed approach. Furthermore Figure 5.2 shows that the proposed

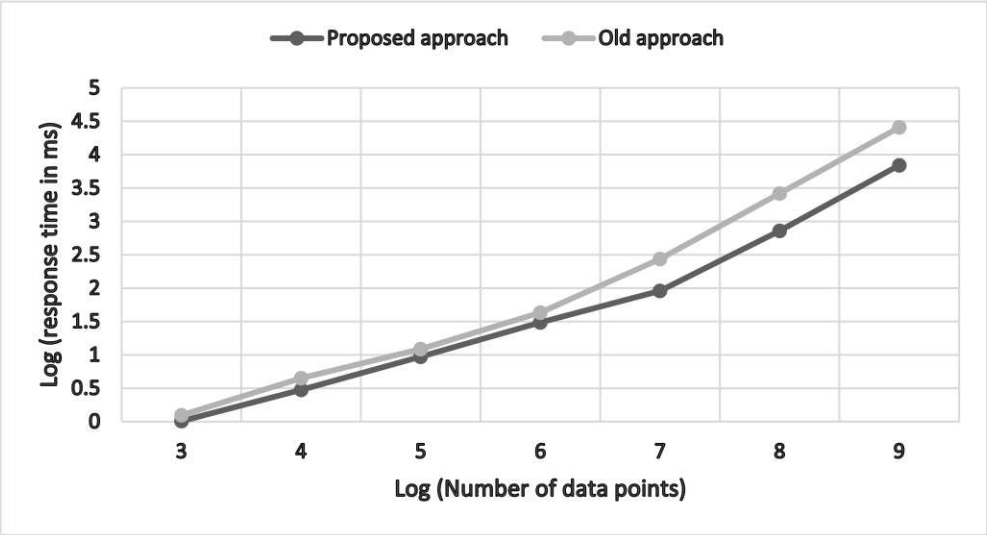


Figure 5.2: A comparison of the scaling of the construction time for the previous and proposed approach

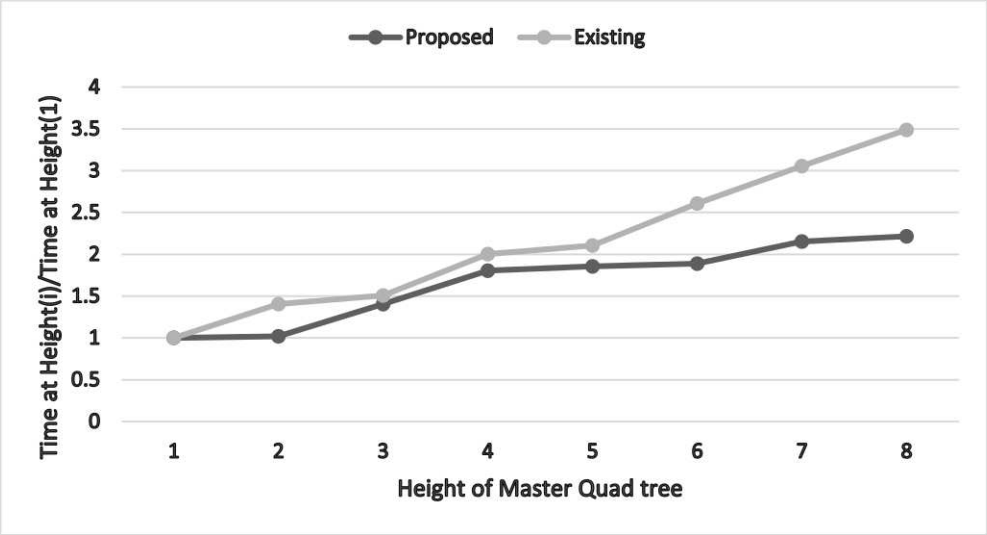


Figure 5.3: A comparison of the time to construct versus the height of the quadtree

solution has a better scaling when compared to the previous approach as it has a slower rise in time to complete construction as compared to the previous approach.

Construction time with respect to depth

In the second experiment we focus on how performance varies with the depth of the directory quadtree. The results illustrated in Figure 5.3 show that the proposed method scales much better than the existing method. At an average quadtree height of 8, for the directory quadtree, the previous approach shows a growth of 3.5 times initial construction time as compared to the proposed approach which shows a 2.2 times initial construction time growth. In both setups we keep the number of data points constant and just vary the size of the leaf nodes hence changes in performance are due to the search for data servers and this is attributed to the height of the quadtree.

5.4 Discussion

The use of spatial data in big-data analytics is increasing. One of the challenges related to use of such data is the bottleneck related to input/output. Amongst the proposed solutions to help alleviate this bottleneck is the use of distributed in-memory quadtrees. To further enhance the distributed in-memory quadtrees we proposed the use of dilated integers and hashmaps to traverse the master quadtree (directory). Our proposed approach demonstrated that using dilated integers together with hashmaps and searching the linear quadtree over height helps to further reduces the time required to search the directory quadtree. Consequently this helps reduce both the construction time and the time required to query such distributed quadtrees. Furthermore we also showed that searching the directory quadtree over height also offers a more scalable solution that is less sensitive to the height of the master quadtree when compared to the previous approach. This ultimately would result in a solution which is less sensitive to the skewed distribution of the spatial data points.

Chapter6

Conclusion

6.1 Summary of the results

Although the prediction of the dominant lineage of influenza B is important for vaccine strain selection, the complex lineage dynamics of influenza B makes this prediction difficult like it is for many other complex models. To this effect we proposed a mathematical model and procedure that uses a likelihood free technique, ABC. Further we parallelised the execution of the simulations to reduce the time taken to do the learning. We then applied our model to influenza B. Results from the experiments suggest that the prediction of the dominant lineage of influenza B may be possible if the epidemiological interference between lineages was quantified. Using the time-series data of the number of laboratory-confirmed influenza B cases per lineage and specific humidity, we estimated the $parameters(\alpha, a, b,$ and $\kappa)$ in the model described in Chapter 3. Based on previous study [65] we parameterized ϵ as $1/\epsilon = 0.6$ day and λ as $1/\lambda = 4.0$ day. The other parameters were estimated for each lineage. We also estimated the herd immunity against each lineage at the beginning of the period explored in this study, i.e., $S_vS_y, S_vR_y, R_vS_y, R_vR_y$ at the beginning of the 2010-2011 season.

Using this model we estimated the epidemiological and evolutional parameters with

the time-series data of the lineage specific isolates in Japan from the 2010-2011 season to the 2014-2015 season. The basic reproduction number is similar between Victoria and Yamagata, with a minimum value during one year as 0.82 (95% highest posterior density(HPD): 0.77-0.87) for the Yamagata lineage and 0.83 (95% HPD: 0.74-0.92) for Victoria, the amplitude of seasonal variation of the basic reproduction number is 0.77 (95% HPD:0.66-0.87) for Yamagata and 1.05 (95% HPD: 0.89-1.02) for Victoria. The duration for which the acquired immunity is effective against the Yamagata infection is shorter than Victoria, 424.1days (95% HPD:317.4-561.5days) for Victoria. The reduction rate of susceptibility due to immune cross reaction is 0.51 (95% HPD: 0.084-0.92) for the immunity obtained from the infection with Yamagata against the infection with Victoria and 0.62 (95% HPD: 0.42-0.80) for the immunity obtained from the infection with Victoria against the infection with Yamagata. Using these estimated parameters, we predicted the dominant lineage in 2015-2016 season. The accuracy of the prediction was 68.8% if the emergence timings of the two lineages are known and 61.4% if the emergence timings are unknown. The emergence timing of a lineage in the host population is a key for the prediction of lineage dynamics. Estimated seasonal variation of the lineage specific reproduction number can narrow down the range of emergence timing, with an accuracy of 64.6% if the emergence times are assumed to be the time at which the estimated reproduction number exceeds one.

The distributed quadtree architecture showed that tree based partitioning achieves the best performance in terms of query response time when compared to simple partitioning and random partitioning strategies. Query redirecting using a tree directory is almost 10 times faster than random partitioning. It is almost twice faster than simple one dimensional partitioning. When compared to the external storage quadtree, the dis-

tributed in-memory quad tree using a tree directory performed almost a 100 times better. Overall our results showed that use of dilated integers coupled with hashmaps can improve the performance of distributed spatial indexing structures used to help alleviate the data access bottleneck in big data spatial analytics.

6.2 Discussion and future work

We proposed a procedure for estimation of epidemics that proved to be quite effective. Furthermore we also proposed an architecture for distributed quadtrees that results in significant gains in performance.

Our proposed procedure for estimating epidemics and results showed that it is quite applicable. Results from this research are of importance they can help in planning mitigation of infectious diseases. Moreover we showed that ABC used in other works li [54] is quite effective and helpful for estimating using complex models. The use of spatial data in Big-Data analytics is increasing. One of the challenges related to use of such data is the bottleneck related to input/output. Amongst the proposed solutions to help alleviate this bottleneck is the use of distributed in-memory quadtrees. To further enhance the distributed in-memory quadtrees we proposed the use of dilated integers and hashmaps to traverse the master quadtree (directory). Our proposed approach demonstrates that using dilated integers together with hashmaps and searching the linear quadtree over height helps to further improve the time required to search the directory quadtree. Consequently this helps reduce both the construction time and the time required to query such distributed quadtrees. Furthermore we also showed that searching the directory quadtree over height also offers a more scalable solution that is less sensitive to the height of the master quadtree when compared to the previous approach. This ultimately

would result in a solution which is less sensitive to the skewed distribution of the spatial data points. Geographic Big-data analytics applications are becoming more and more important. With the current development in processing power I/O is becoming a notable bottleneck. The proposed architecture demonstrates that distributing a quad tree into several servers improves the query performance by more than a hundred fold. In addition we also showed that careful selection of partition assignment strategies results into much more evenly distributed data. This is very important as the quad tree is queried from memory.

Further, using a master quadtree helps reduce the impact and effect of clustered and skewed data. This approach is also location aware and thus gives good performance for both data insertion and query servicing. This result is of significance as improving I/O ultimately improves the performance of a data assimilation process as demonstrated in [5, 37].

For future work we intend to extend the idea and put into consideration reliability by implementing backup master servers and also process migration and redistribution in case of failure. We also plan to apply this approach to storage of data from the modelling of multi-strain epidemics data assimilation and use it for data assimilation. For future work we intend to extend the idea and put into consideration reliability by implementing backup master servers and also process migration and redistribution in case of failure. We also plan to apply this approach to storage of data from the modeling of multi-strain epidemics data assimilation and use it for data assimilation. We also intend to investigate an optimal hashing scheme to apply to the linear quadtree. We also plan to study the implementation of redundancy of the master server and threading options for better performance.

Bibliography

[1] D.J. Abel and J.L. Smith. A data structure and algorithm based on linear key for a rectangle retrieval problem. *Computer Vision, Graphics and Image Processing*, 24:1–13, 1983.

[2] M.A. Beaumont, W. Zhang, and D.J. Balding. Approximate Bayesian computation in population genetics. *Genetics*, 162(4):2025–2035, 2002.

[3] T. Bedford, A. Rambaut, and M. Pascual. Canalization of the evolutionary trajectory of the human influenza virus. *BMC biology*, 10:38, 2012.

[4] S.K. Bhaskar and A. Rosenfeld. Parallel processing of regions represented by linear quadtrees. *Computer Vision, Graphics, and Image Processing*, 42(3):371–380, 1988.

[5] M. Buehner, T. Miyoshi, A. Lorenc, K. Eugenia, and P.J. VanLeeuwen. Observations and data assimilation: Data assimilation methodology and diagnostic tools. World Weather Open Science Conference (WWOSC2014), Montreal, Canada, 16-21 August, 2014.

[6] C. Castillo-Chavez, H.W. Hethcote, V. Andreasen, S.A. Levin, and W.M. Liu. Epidemiological models with age structure, proportionate mixing, and cross-immunity. *Journal of Mathematical Biology*, 27:233–258, 1989.

[7] B. Cazelles and N.P. Chau. Using the Kalman filter and dynamic models to assess the changing HIV/AIDS epidemic. *Mathematical Biosciences*, 140:131–154, 1997.

[8] F. Dehne, A. Rau-Chaplin, and A.G. Ferreira. Hypercube algorithms for parallel processing of pointer-based quad trees. *Computer Vision and Image Understanding*, 66(1):1–10, 1995.

[9] V. Dukic, H.F. Lopes, and N.G. Polson. Tracking epidemics with Google Flu Trends data and a state-space SEIR model. *Journal of the American Statistical Association*, 107(500):1410–1426, 2012.

[10] R. Finkel and J. Bentley. Quadtrees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[11] I. Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910, 1982.

[12] W.P. Glezen. Editorial commentary: Changing epidemiology of influenza B virus. *Clinical Infectious Diseases : An Official Publication of the Infectious Diseases Society of America*, 59:1525–1526, 2014.

[13] J.R. Gog and B.T. Grenfell. Dynamics and selection of many-strain pathogens. *Proceedings of the National Academy of Sciences of the United States of America*, 99(26):17209–17214, 2002.

[14] N.C. Grassly and C. Fraser. Seasonal infectious disease epidemiology. *Proceedings of the Royal Soceity B. Biological Sciences*, 273:2541–2550, 2006.

[15] S. Gupta, N. Ferguson, and R. Anderson. Chaos, persistence, and evolution of strain structure in antigenically diverse infectious agents. *Science*, 280:912–915, 1998.

[16] W.K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[17] M. Hopping, A. Fonville, J.M. Russell, C.A. James, S. Smith, and D.J Smith. Influenza B vaccine lineage selection–an optimized trivalent vaccine. *Vaccine*, 34(33):1617–1622, 2016.

[18] S.C. Howard and C.A. Donnelly. Estimation of a time-varying force of infection and basic reproduction number with application to an outbreak of classical swine fever. *Journal of Epidemiology and Biostatistics*, 5:161–168, 2000.

[19] B.R. Hunt, E.J. Kostelichb, and I. Szunyoghc. Efficient data assimilation for spatiotemporal chaos: A local ensemble transform kalman filter. *Physica D: Nonlinear Phenomena*, 230(1-2):112–126, June 2007.

[20] K. Ito, M. Igarashi, Y. Miyazaki, T. Murakami, S. Iida, H. Kida, and A. Takada. Gnarled-trunk evolutionary model of influenza a virus hemagglutinin. *PLOS ONE*, 6:e25953, 2011.

[21] D. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, Dept. of Mathematics, M.I.T., Cambridge, MA, 1973.

[22] R.E. Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME- Journal of Basic Engineering, Series D*, 82:34–45, 1960.

[23] M. Kamo and A. Sasaki. The effect of cross-immunity and seasonal forcing in a multi-strain epidemic model. *Physica D-Nonlinear Phenomena*, 165:228–241, 2002.

[24] S. Karsten, G. Rave, and J. Krieter. Monte Carlo simulation of classical swine fever epidemics and control. i. general concepts and description of the model. *Veterinary Microbiology*, 108:187–198, 2005.

[25] S. Kasif. Optimal parallel algorithms for quadtree problems. *CVGIP: Image Understanding*, 59(3):281–285, 1994.

[26] A. Klinger. Patterns and search statistics. In *Optimizing Methods in Statistics Proceedings of a Symposium Held at the Center for Tomorrow, the Ohio State University,*, pages 303–337. Academic Press, 1971. icmcs18.

[27] R.E. Korf. A new algorithm for optimal bin packing. In *The 18th National Conference on Artificial Intelligence, pages 731-736*, 2002.

[28] S. Kryazhimskiy, U. Dieckmann, S.A. Levin, and J. Dushoff. On state-space reduction in multi-strain pathogen models, with an application to antigenic drift in influenza A. *PLOS Computational Biology.*, 2007.

[29] J.P. Lauzon, D.M. Mark, L. Kikuchi, and J.A. Guevara. Two-dimensional run-encoding for quadtree representation. *Computer Vision, Graphics, and Image Processing*, 30(1):56–69, 1985.

[30] M. Lenorm, F.Jabot, and G.Deffuant. Adaptive approximate Bayesian computation for complex models. *Computational Statistics*, 28(6):2777–2796, 2013.

[31] M. Luksza and M. Lassig. A predictive fitness model for influenza. *Nature*, 507:57–61, 2014.

[32] J. Mandel, J.D. Beezley, L. Cobb, and A. Krishnamurthy. Data driven computing by the morphing fast Fourier transform ensemble Kalman filter in epidemic spread simulations. *Procedia Computer Science*, 1(1):1221–1229, 2010.

[33] P. Marjoram. Approximation Bayesian computation. *OA Genetics*, 1(1):1–5, 2013.

[34] L. Martino and J. Miguez. A generalization of the adaptive rejection sampling algorithm. *Statistics and Computing*, 21(4):633–647, 2010.

[35] Meteorological Development Laboratory/Office of Science and Technology/National Weather Service/NOAA/U.S. Department of Commerce. 1987, updated half-yearly. TDL U.S. and Canada surface hourly observations, daily 1976Dec-cont. Research data archive at the National Center for Atmospheric Research, Computational and Information Systems Laboratory. [online]. Available: http://rda.ucar.edu/datasets/ds472.0/. [Accessed: July 28, 2014].

[36] P. Minayev and N. Ferguson. Improving the realism of deterministic multi-strain models: implications for modelling influenza. *Journal of the Royal Society Interface*, 2009.

[37] T. Miyoshi, K. Kondo, and K. Terasaki. Big ensemble data assimilation in numerical weather prediction. *Computer*, 48(11):15–21, May 2015.

[38] N.M. Molinari, I.R. Ortega-Sanchez, M.L. Messonnier, W.W. Thompson, P.M. Wortley, E. Weintraub, and C.B. Bridges. The annual impact of seasonal influenza in the US: measuring disease burden and costs. *Vaccine*, 25:5086–5096, 2007.

[39] M. Nielen, A.W. Jalvingh, M.P. Meuwissen, S.H. Horst, and A.A. Dijkhuizen. Spatial and stochastic simulation to evaluate the impact of events and control measures on the 1997-1998 classical swine fever epidemic in the Netherlands. ii. comparison of control strategies. *Preventive Veterinary Medicine*, 42:297–317, 1999.

[40] M. Nyirenda, H. Suleman, A. Maunder, and R. vanRooyen. X-Switch: An effi-

cient, multi-user, multi-language Web application server. *South African Journal of Computer Science*, 44:57–68, 2009.

[41] M.A. Oliver and N.E. Wiseman. Operations on quadtree leaves and related image areas. *Computer Journal*, 26(4):375–380, 1983.

[42] R. Omori and A. Sasaki. Timing of the emergence of new successful viral strains in seasonal influenza. *Journal of Theoretical Biology*, 329:32–38, 2013.

[43] R. Rasmussen and G. Hamilton. An approximate Bayesian computation approach for estimating parameters of complex environmental processes in a cellular automata. *Environmental Modelling & Software*, 29(1):1–10, 2012.

[44] P. Rohani, D.J. Earn, B. Finkenstadt, and B.T. Grenfell. Population dynamic interference among childhood diseases. *Proceedings of the Royal Society B-Biological Sciences*, 265:20332041, 1998.

[45] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addision-Wesly, Reading, MA, 1990.

[46] S.C.Howard and C.A. Donnelly. Estimation of a time-varying force of infection and basic reproduction number with application to an outbreak of classical swine fever. *Journal of Epidemiology and Biostatistics*, 5:161–168, 2000.

[47] I.B. Schwartz and H.L. Smith. Infinite subharmonic bifurcation in an SEIR epidemic model. *Journal of Mathematical Biology*, 18:233–253, 1983.

[48] J. Shaman and M. Kohn. Absolute humidity modulates influenza survival, transmission, and seasonality. *Proceedings of the National Academy of Sciences of the United States of America*, 106:3243–3248, 2009.

[49] J. Shaman, V.E. Pitzer, C. Viboud, B.T. Grenfell, and M. Lipsitch. Absolute humidity and the seasonal onset of influenza in the continental United States. *PLOS Biology*, 8:e1000316, 2010.

[50] G. Shrack. Finding neighbors of equal size in linear quadtrees and octrees in constant time. *CVGIP: Image Understanding*, 55:221–230, 1992.

[51] D.M. Skowronski, T.S. Hottes, M. Chong, G. De Serres, D.W. Scheifele, B.J. Ward, S.A. Halperin, N.Z. Janjua, T. Chan, S. Sabaiduc, and M. Petric. Randomized controlled trial of dose response to influenza vaccine in children aged 6 to 23 months. *Pediatrics*, 128:276–289, 2011.

[52] L. Stone, R. Olinky, and A. Huppert. Seasonal dynamics of recurrent epidemics. *Nature*, 446:533–536, 2007.

[53] M. Sunnaker, A.G. Busetto, E. Numminen, J. Corander, M. Foll, and C. Dessimoz. Approximate Bayesian computation. *PLOS Computational Biology*, 9:e1002803, 2013.

[54] M.M. Tanaka, A.R. Francis, F. Luciani, and S.A. Sisson. Using approximate Bayesian computation to estimate tuberculosis transmission parameters from genotype data. *genetics*, 173:1511–1520, 2006.

[55] W.W. Thompson, D.K. Shay, E. Weintraub, L. Brammer, N. Cox, L.J. Anderson, and K. Fukuda. Mortality associated with influenza and respiratory syncytial virus in the United States. *JAMA*, 289:179–186, 2003.

[56] T.Lee and H.Shin. Combining syndromic surveillance and ILI data using particle

filter for epidemic state estimation. *Flexible Services and Manufacturing Journal*, 28(1):233–253, 2016.

[57] T. Toni and M.P.H. Stumpf. Simulationbased model selection for dynamical systems in systems and population biology. *Bioinformatic*, 26(1):104–110, 2010.

[58] F. Tria, M. Lassig, L. Peliti, and S. Franz. A minimal stochastic model for influenza evolution. *Journal of Statistical Mechanics-Theory and Experiment*, 2005.

[59] B.M. Turnera and T. Van Zandtb. A tutorial on approximate Bayesian computation. *Journal of Mathematical Psychology*, 56(2):69–85, 2012.

[60] R.R. Vatsavai and B. Bhaduri. Geospatial analytics for big spatiotemporal data: Algorithms, applications and challenges. [Online]. Available: http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/raju-bigspatial.pdf. [Accessed: March 27, 2015], 2013.

[61] D. Vijaykrishna, E.C. Holmes, U. Joseph U, M. Fourment, Y.C.F. Su, R. Halpin, R.T.C. Lee, Y. Deng, V. Gunalan, X. Lin, T. Stockwell, N.B. Fedorova B. Zhou, N. Spirason, D.K. Khnert, V. Boskova, T. Stadler, A. Costa, D.E. Dwyer, Q.S. Huang, L.C. Jennings, W. Rawlinson, S.G. Sullivan, A.C. Hurt, S. Maurer-Stroh, D. Wentworth, G.J.D Smith GJD, and I.G. Barr. The contrasting phylodynamics of human influenza B viruses. *eLife*, 4:e05055, 2015.

[62] T. Wai-Yuan and Y. Zhengzheng. Estimation of HIV infection and incubation via state space models. *Mathematical Biosciences*, 167(1):31–50, 2000.

[63] E. Wolfgang, R. Muller-Pfefferkorn, M. Kluge, and D. Hackenberg. Execution environments for big data: Challenges

for storage architectures and software. [Online]. Available: http://www.exascale.org/bdec/sites/www.exascale.org.bdec/files/whitepapers/raju-bigspatial.pdf [Accessed: March 27, 2015], 2013.

[64] J.R. Woodwark. The explicit quadtree as a structure for computer graphics. *Computer Journal*, 25(2):235–238, 1982.

[65] C. Xu, K. Chan, T.K. Tsang, V.J. Fang, R.O.P. Fung, D.K.M. Ip, S. Cauchemez, G.M. Leung, J.S.M. Peiris, and B.J. Cowling. Comparative epidemiology of influenza B Yamagata- and Victoria-lineage viruses in households. *American Journal of Epidemiology*, 182:705–713, 2015.

[66] S. Yadav and H. Weng. Estimating the scale of adverse animal welfare consequences of movement restriction and mitigation strategies in a classical swine fever outbreak. *BMC Veterinary Research*, 13:83, 2017.