Technische Universiteit Delft

Modeling the Spread of Epidemics

Visualizing the Spread of COVID-19 and Applying a Markov-Modulated Process Model to Mobility Processes

Brian Chang

TU Delft

NAS

Modeling the Spread of Epidemics

Visualizing the Spread of COVID-19 and Applying a Markov-Modulated Process Model to Mobility Processes

by

Brian Chang

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday June 28, 2021 at 1:30 PM.

Student number:	4913884	
Project duration:	August, 2020 – June, 2021	
Thesis committee:	Prof. dr. ir. Piet Van Mieghem,	TU Delft, supervisor
	Dr. Johan Dubbeldam,	TU Delft
	Dr. Mattia Sensi,	TU Delft, daily supervisor

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft

Abstract

This thesis consists of two parts which are connected by the central theme of epidemics. In the first part, a website is designed for forecasting the number of cases of COVID-19 in the Netherlands. The forecasting is performed using the Network-Inference-Based Prediction Algorithm (NIPA). The first part of the thesis documents the design process and some of the challenges faced along the way. An explanation of the NIPA algorithm is presented, and its implementation on the website is discussed.

The second part of this thesis shifts the focus to the modeling of mobility processes. Human-to-human transmission is often the dominant viral transmission vector, and therefore the spreading of the virus is inextricably linked to human mobility. The Markov-modulated process (MMP) model has recently been proposed as a novel method for modeling mobility processes. Yang [1] has developed a so-called "aggregated MMP model" which is able to capture several key dynamics of mobility processes with high accuracy. This research builds on Yang's results and proposes the "quantized MMP model" as an extension of Yang's aggregated MMP model. Compared to the aggregated MMP model, the quantized MMP model reduces the number of required states by a factor equal to the quantization step size q, improving the efficiency of the model.

The behavior of the quantized MMP model for different step sizes q is compared with the aggregated MMP model. The results show that the quantization error has zero mean, allowing the quantized MMP model to achieve similarly high model accuracy. The SIR dynamics of the quantized MMP model for different step sizes are tested extensively and the disease spreading performance is compared with the mobility process. The suitability of the MMP model for forecasting the evolution of epidemics is evaluated and discussed.

Preface

This thesis project has been carried out at the Network Architectures and Services (NAS) group of the Delft University of Technology as part of the graduation requirements for a Master of Science in Electrical Engineering. My thesis project consisted of two parts; in the first part, I worked on creating a website and in the second, I have researched the modeling of mobility processes. I would like to thank Professor Piet Van Mieghem for the opportunity to work on these two very different but equally interesting topics. It has been a pleasure working on my thesis project at the NAS group.

The first part of my thesis project was supervised by Long Ma and Massimo Achterberg. I would like to thank them both for their guidance and for being the beta testers for the website. The quality of the final product has been made possible by their helpful suggestions and bug reports, and I am very grateful for their contributions. I would like to thank Massimo again for providing Dutch translations for the website.

The second part of my thesis was supervised by Dr. Mattia Sensi, Fenghua Wang, and Massimo Achterberg. I would like to express my gratitude to them for taking the time to meet every week and for all of the invaluable feedback and support they have provided over the duration of the project. Working with them has been a wonderful experience.

I would further like to thank Dr. Johan Dubbeldam agreeing to join my thesis committee.

Brian Chang
Den Haag, June 2021

Contents

I

Visualizing the Spread of COVID-19

1 Introduction

1.1. Problem Statement

The Network Architectures and Services (NAS) group at the Delft University of Technology (TU Delft) conducts research in the area of complex networks and members of the group have recently published a paper *Network-inference-based prediction of the COVID-19 epidemic outbreak in the Chinese province Hubei* [2]. The paper proposes the Network-Inference-Based Prediction Algorithm (NIPA) to forecast the prevalence of COVID-19. The objective of the first part of this master thesis is to deploy a website on the server of the NAS group that displays the daily confirmed cases of COVID-19 in the Netherlands and applies NIPA to provide a forecast of the cases in the following days.

As part of a bachelor thesis, a group of Computer Science students at TU Delft have previously created a website which obtains the daily confirmed cases of COVID-19 from the National Institute for Public Health and the Environment (Dutch: *Rijksinstituut voor Volksgezondheid en Milieu* or *RIVM*) [3]. The dashboard of the website is visible in Figure 1.1.[1] The backend of the website processes the data and implements the NIPA algorithm to forecast the number of cases in the following days. The frontend retrieves the data from the backend and displays the data to the user. The students have generously agreed to allow us to adapt the website they have made for use by the NAS group. The objective of the first part of this master thesis project is to adapt the frontend and backend of the website so that it can be hosted on the NAS servers. Additionally, we will make changes to the frontend of the website to better suit the needs of the NAS group, such as providing an English translation.

1.2. Initial Planning and Approach

We divide our approach to this problem into three phases with three distinct goals in mind. In phase 1, the goal is to deploy a basic version of the website. To start with, we will need to set up and familiarize ourselves with the tools that were used to develop the original version of the website. We will then need to study the code of the frontend and see if any modifications need to be made to make it compatible with the NAS servers. Once the frontend is running, we will need to learn how to create and deploy a backend to supply our frontend with data. The backend should automatically update the website with the latest data from RIVM and provide forecasts based on the latest data.

With the frontend and backend both functional, we will then move onto phase 2, where we focus on making modifications to the frontend of the website and adapt it to better suit the needs of the NAS department. One of the main goals of this phase is to translate the website into English, but we would like also to preserve the Dutch version of the website and allow users to select their desired language. We also want to rework the About Us page of the website, because it currently introduces the project carried out by the original students and this is not relevant to the NAS department. We will create a new version of the About Us page to introduce the NAS department and its position here at TU Delft, but we will still include a section on this page that acknowledges the original creators of the website. Finally, to make our website fully independent, we will remove the link to the API of the original website and replace it with a new Downloads page where we make the data used on our webpage available for download.

[1] The website produced by the students is available at `http://networkdatascience.ewi.tudelft.nl/`. Note that due to a bug in the code of the backend, the website has not been updating data since January 7, 2021.

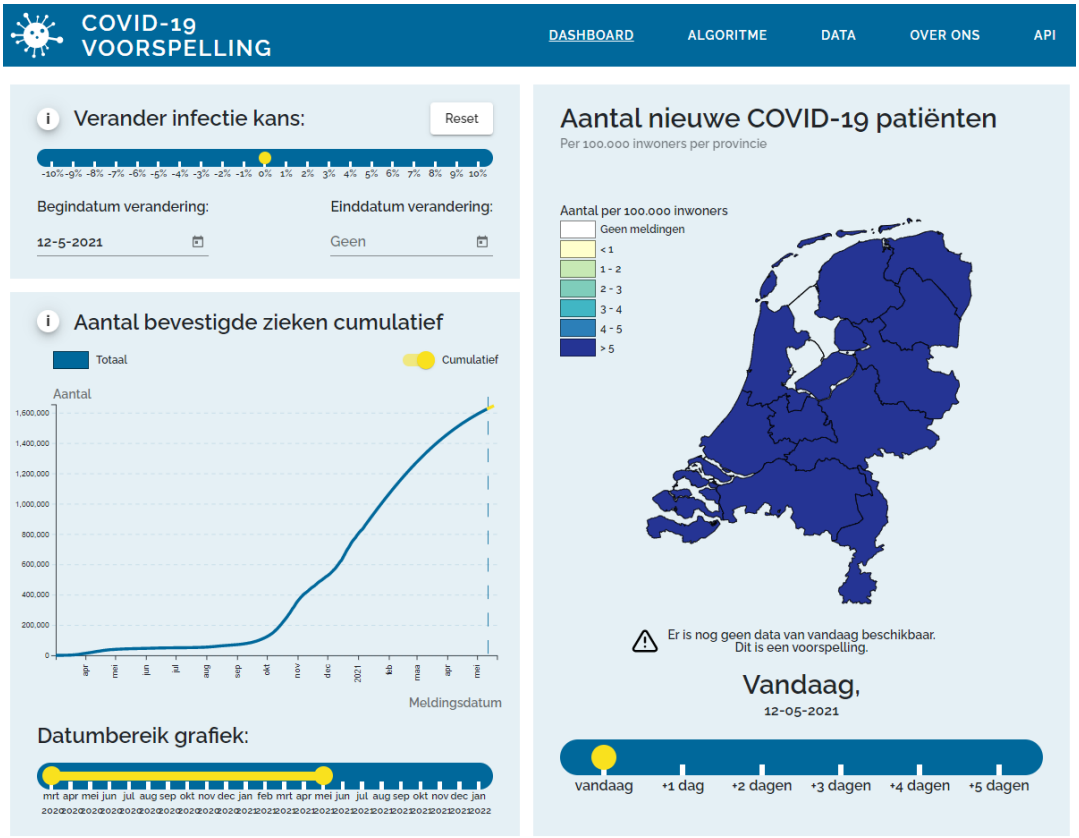


Figure 1.1: Dashboard of the original website produced by the group of Computer Science students.

Once the frontend changes are complete, we will begin phase 3 where we shift our focus to the backend. The goal of this phase is to validate the Python implementation of the NIPA algorithm that was created by the original group of Computer Science students. We want to make sure that the implementation of NIPA used by the website is correct. We choose to do this by studying the paper [2] ourselves to gain an understanding of the algorithm, and then writing our own implementation of the algorithm. We will then compare the implementation of the algorithm created by the Computer Science students with our own.

2

Phase 1: Deploying a Basic Version of the Website

2.1. Initial Setup

The original website created by the bachelor students consists of a frontend and backend components. The frontend of the website controls the appearance of the website and how the user interacts with the website. The backend is hidden from the user and is responsible for collecting the daily updated infection data from RIVM and running NIPA, an algorithm that forecasts the number of infected cases. The backend has an Application Programming Interface (API) which allows users to request the processed data and forecasts. The frontend of the website makes use of this API to send a request to the backend and retrieve the processed infection data and forecasts so that it can be displayed on the website.

Very limited documentation was provided for the source code of the original website. The original website was developed on Linux, and the documentation also assumes that the user is using Linux. Since we were not familiar with the tools that were used to develop the website and did not know whether there were Windows alternatives, we decided to use Linux as well. We started by installing a fresh copy of Ubuntu 20.04.1 LTS on a virtual machine in VirtualBox.

The frontend of the website was developed in Angular, which is a web application framework based on the TypeScript language. We consulted a tutorial for setting up Angular on Ubuntu [4]. Since we are on a fresh install of Ubuntu, we are missing some commonly used command line programs which we needed to install before we can do anything. We started by installing Git and cURL by entering the following commands into the terminal.

```
sudo apt install git
sudo apt autoremove
sudo apt install curl
```

With these programs installed, we can now start setting up Angular. The first step is to install Node.js, an open-source server environment that is required by Angular. The latest version at the time was v15.x.

```
curl -sL https://deb.nodesource.com/setup_15.x | sudo -E bash -
sudo apt-get install -y nodejs
```

The next step is to update npm to ensure that we are using the latest version. npm is the package manager for Node.js and is typically installed along with it, but it is still good practice to update it manually.

```
sudo npm install npm@latest -g
```

The final setup is to install the Angular CLI (command line interface).

```
sudo npm install -g @angular/cli
```

With the required tools installed, we can now set up the Angular project. We do not have to make a new project because the original code already includes all of the project files. We can continue working on this project, but we first need to install all of the required libraries on our machine. We do this by navigating to the project directory and running the following command in the terminal.

```
npm ci
```

2.2. First Website Deployment

The original website was deployed on the GitLab servers of the EEMCS faculty at TU Delft, and the code makes use of GitLab CI/CD (continuous integration and continuous delivery) in order to deploy the website. We do not have access to these servers, and furthermore we do not have any experience with setting up a CI/CD pipeline since we do not have a background in Computer Science. Furthermore, the original CI/CD pipeline is not suitable for our needs because we would like to deploy the website NAS servers and we do not have the option to run a backend there. Therefore, we choose to develop the frontend and backend of the website separately.

Having set up Angular, we can start a development server using the following command.

```
ng serve --watch
```

This will allow us to view a live preview of the website while developing as shown in Figure 2.1. After starting the development server, we verified that the code does indeed compile and the website is being displayed correctly on our local machine.

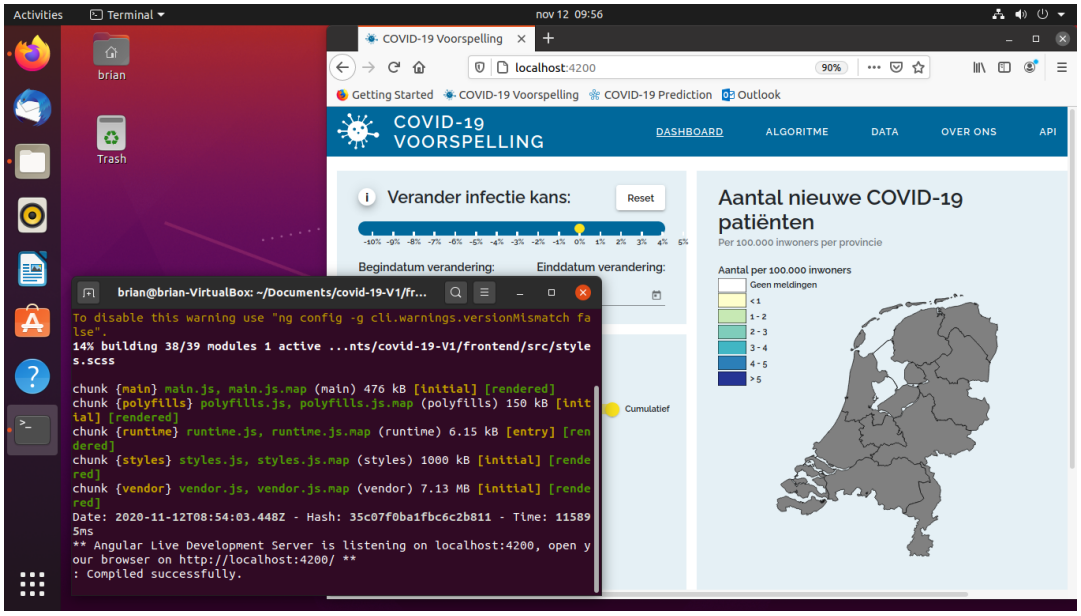


Figure 2.1: After starting the development server, we can view a live preview of the website so we can see immediately see the effects of any changes that we make.

Before we make any changes to the website, we first deploy the website to the NAS server as-is in order to familiarize ourselves with the procedure and identify any issues. We start by compiling the frontend in Angular using the following command.

```
ng build --prod
```

The `--prod` flag tells Angular to use the production environment which changes some variables inside the website. Defining a production environment and a development environment is useful because it allows us to easily change variables. For example, during development, we would like the frontend to retrieve the data from a local URL so that we can test the website. However, during production, the data needs to be retrieved from the real backend. We can therefore define a variable for the data URL and set it to different values in the production and development environments.

After compiling the website, we upload the compiled files to the NAS server. However, when navigating to our website, we are presented with a completely blank page. After some troubleshooting, we determined that the issue is due to the fact that our compiled application expects to be located at the root of the website `https://nas.ewi.tudelft.nl/`, but it is actually located at `https://nas.ewi.tudelft.nl/nipa/covid-prediction/`. To resolve this issue, we add an extra argument when compiling the website.

```
ng build --prod --base-href /nipa/covid-prediction/
```

After recompiling the frontend in Angular and uploading the files to the NAS server, we find that we are able to see the main page of the website, but we do not see any data on the website as shown in Figure 2.2. Furthermore, navigating to any of the other pages of the website results in a server error; the server cannot find the requested page as shown in Figure 2.3.

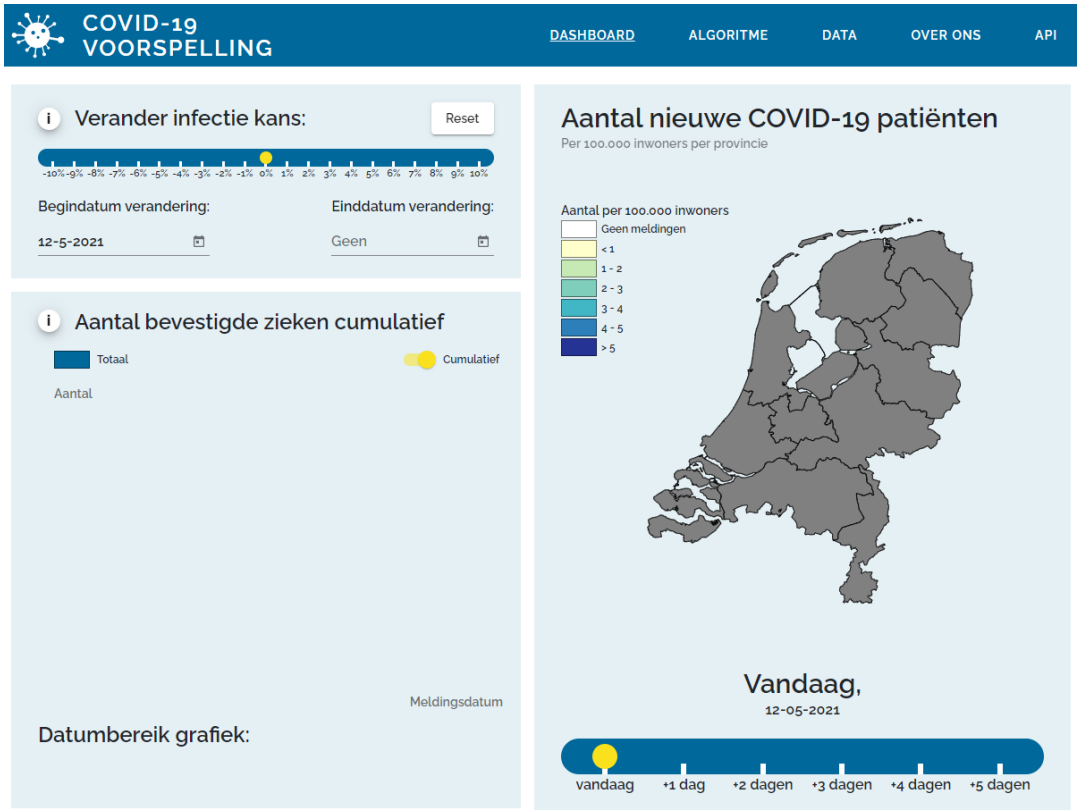


Figure 2.2: The website is initially unable to receive any data due to mixed content being blocked by default in most browsers.

The reason that no data is visible is due to mixed content. Mixed content occurs when the initial request to the webpage is sent over HTTPS, but the reply contains both HTTPS and HTTP content. Because HTTPS is secure while HTTP is not, most browsers will block the HTTP content if the initial request was sent over HTTPS. This problem occurs because we have not made any modifications to the frontend of the website yet, so it still tries to retrieve data from the original backend. The initial request to the NAS server is made over HTTPS. The webpage then tries to request data from the original backend, but the original backend does not support HTTPS so it sends the data over HTTP. This results in the data being blocked if mixed content is not explicitly enabled in the browser.

With regard the navigation issues, we discovered that the problem arises from the fact that Angular applications are designed to be single-page application, and routing is meant to be handled internally by the Angular application. The web server needs to be configured to redirect users to the index of the Angular application in order for it to function correctly. To illustrate with an example, suppose we have an Angular application located at http://www.xyz.com/mypage/; for brevity, we will refer to the location as /mypage/. When we send a request for /mypage/, the web server will serve us with the index of the Angular page, located at /mypage/index.html. Now suppose that we have a page titled “about” in our Angular application which we

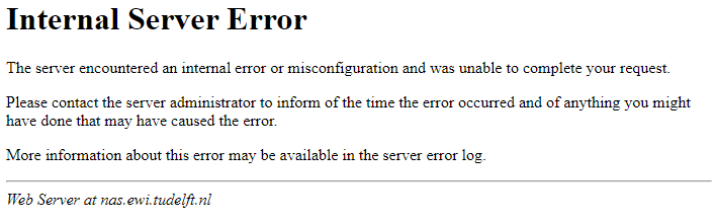


Figure 2.3: The NAS web server returns an error message when trying to navigate to the other pages of the website.

want to open, so we send a request to the web server for /mypage/about. Because Angular applications are designed to be single-page applications, so the "about" page does not physically exist on the web server. The intended behavior is as follows:

1. We request the page /mypage/about from the web server.
2. The web server discovers that the request page does not exist on the server.
3. The web server instead serves us with the index of the Angular application, /mypage/index.html.
4. The Angular application sees that we have requested the "about" page and will display the content of the page.

In order for this process to work, the critical points are in step 2 and step 3. The web server must be configured to redirect to the index of the Angular application whenever a request is made for a page which does not exist on the server. In the case of the NAS servers, the web server has not been configured in this way, so it shows us an error when we try to request a page which does not exist. We do not have permission to modify the configuration of the server, so we must find an alternative solution.

We address this issue by making copies of the Angular index for each page in our application and naming them accordingly. So if there is a page titled "about", we make a copy of index.html and name it "about.html". When we make a request to the web server for the "about" page, the web server will now return "about.html" which is an exact copy of the index. The Angular application will then display the about page. After making this change, we verified that we are now able to navigate to different pages when the website is deployed on the NAS servers. We can now move onto addressing the issue of data not being visible due to mixed content.

2.3. Creating a Temporary Backend

The issue of mixed content arises from the fact that we have not yet implemented our own backend, so the website is requesting data from the original backend which is served over HTTP. Once we have set up our own backend, this will no longer be an issue. One of the challenges faced in setting up a backend is finding an appropriate service on which to host the backend. The original backend was written in Python, and we would like to continue working in Python as well. We therefore searched for a service on which we could deploy a Python backend.

The service we found that showed the most promise was Heroku, which is a cloud application platform that can be used to host a variety of apps, including a Python-based web backend. The free tier of service offers 550 free hours of processing time per month, with an additional 450 free hours for accounts with a verified credit card for a total of 1,000 hours. This should easily be enough for us to host a web backend.

However, there are some challenges that need to be addressed if this service is to be used. The biggest issue is that if no web traffic is received for 30 minutes, the backend will be put to sleep. While there are third-party services that can be used to periodically ping the backend to prevent it from sleeping, this is not an ideal solution. If the backend has gone to sleep, this will incur a delay of about 5-10 seconds for the next request, which we would like to avoid.

The second issue is that Heroku has an "ephemeral" hard drive. This is typical for most cloud computing services, and it means that while files can be written to the disk, these files will be lost once the application is restarted. This could be due to a manual restart or the application crashing, and in any case, Heroku automatically restarts applications every 24 hours. We would like to maintain a record of all of the data that we process, so we will need some place to store it.

Amazon S3 is a cloud storage service offered by Amazon Web Services (AWS). The free tier of service is valid for one year and supports 5GB of storage, 20,000 get requests per month (retrieving data from S3), and 2,000 put requests per month (writing data to S3). This should be sufficient for our use case, and using S3 can actually solve both of our current issues with Heroku. Obviously, storing files to S3 will prevent them from being lost when our backend restarts. But since we are storing the data to S3 anyways, we can also modify our frontend to retrieve the data from S3. This way, our backend no longer needs to respond to requests from the frontend and the sleeping is no longer an issue. In fact, we could go one step further and simplify the backend so that it does not need to respond to requests at all: instead of having an API for users to request data from the backend, we can simply allow users to download the data from S3. The backend then only needs to process data and write the results to S3.

As a proof of concept, we start by writing a basic backend which does not do any data processing yet; instead, it downloads the processed data from the old backend and then uploads the data to S3. In S3, we create a storage object called a "bucket", and we then we generate a pair of access keys which we provide to Heroku so that it has the necessary credentials to write files to our bucket. We then deploy our backend to Heroku and set up a script to run our backend every 10 minutes as shown in Figure 2.4.

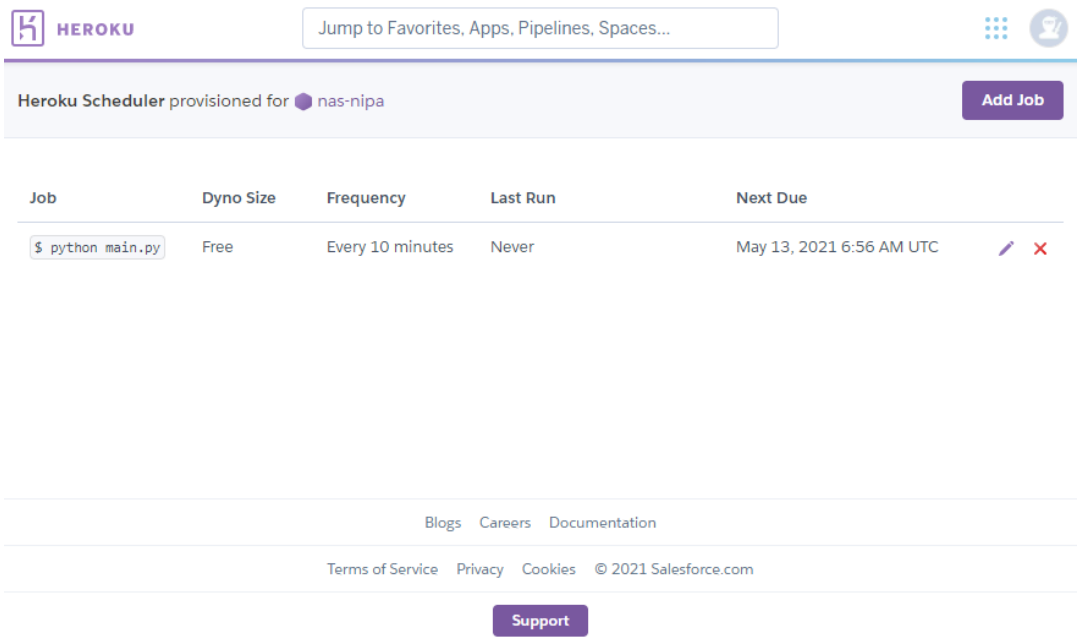


Figure 2.4: The Heroku Scheduler can be used to schedule our backend to run at set intervals. Illustrated is a job scheduled to run at 10 minute intervals.

After verifying that the backend is functioning and the data is being written to S3, we modify the frontend to retrieve the data from our S3 bucket. After compiling the new version of the frontend and deploying it to the NAS servers, however, we observed that we were still not see any data on the website. The issue is no longer due to mixed content, but rather cross-origin resource sharing (CORS). The frontend makes a request to S3 for the data, but because the request originated from a different domain, the request was blocked by S3. After reconfiguring the S3 bucket to allow CORS as shown in Figure 2.5, we are finally able to see data on our website.

2.4. Creating a Real Backend

Now that we have verified that our backend is able to write data to S3 and the data is accessible by the frontend, we can begin creating a real backend which performs data processing. The RIVM reports the cumulative number of confirmed cases of COVID-19 per province per day, and Esri Nederland makes the data accessible via ArcGIS, an online geographic information system maintained by Esri Nederland. Using the ArcGIS API for Python, we can query the service to obtain the raw data [5].

The raw data published by the RIVM reports the cumulative number of confirmed cases per day per municipality, but for the NIPA forecasting, we require the number of new confirmed cases per day per province.

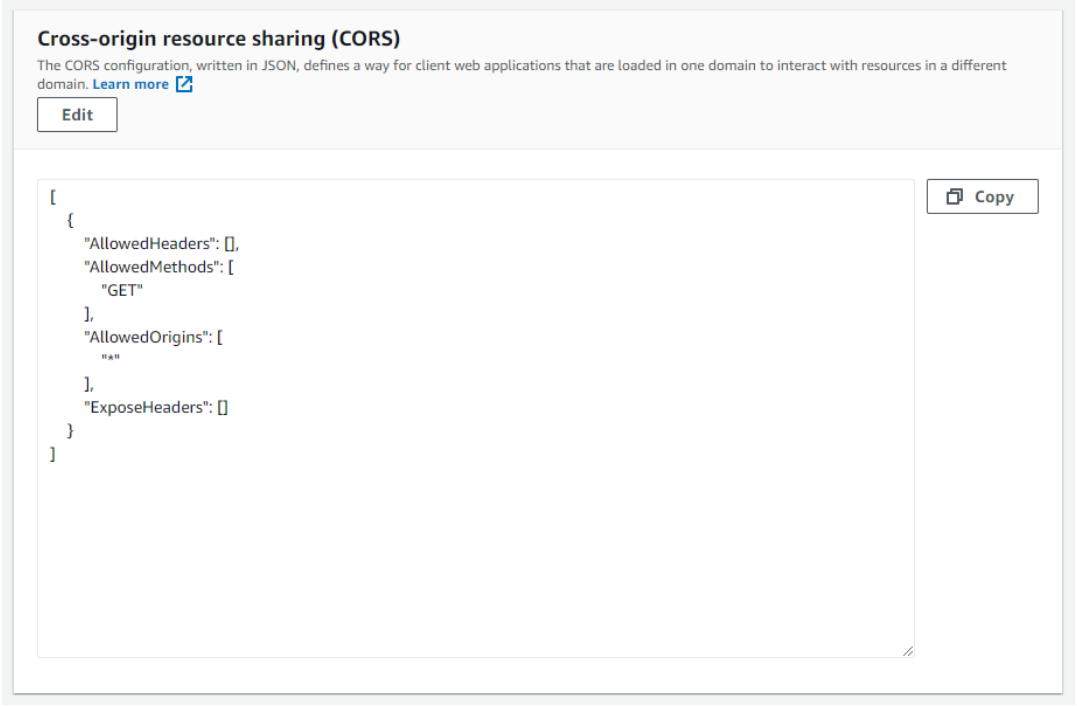


Figure 2.5: The following CORS configuration allows the "GET" method from any origin, allowing the data stored in the public directory to be accessible from anywhere on the internet.

One issue which arises is that there are some days in the data for which data is not available in certain municipalities. The first instance of missing data is March 2, 2020, and the last occurrence of missing data is on April 14, 2020. To address the issue of missing data, the number of confirmed cases is filled in on the missing days by means of linear interpolation. If a municipality does not have data on the first day (March 2, 2020), we set it to 0. The number of confirmed cases needs to be an integer, so when the linear interpolation results in a non-integer value, it is cast to an integer using a floor function (i.e. rounding down to the nearest integer).

After linearly interpolating to fill in number of confirmed cases for the missing days, we sum the number of cumulative confirmed per day per municipality over the province that each municipality belongs. We can then take the difference between the number of cumulative confirmed cases between each pair of days to find the number of new confirmed cases per day per province. On some days, we observe that the number of newly confirmed cases is negative, indicating that the cumulative number of confirmed cases has decreased. The most likely explanation for this is that the number of confirmed cases of the previous day was incorrectly entered and set too high. When these negative values are encountered, they are set to 0. After these preprocessing steps are complete, we can run the NIPA algorithm to forecast the number of confirmed cases for the following days. For the moment, we directly use the code of the NIPA implementation from the original backend.

The final issue that needs to be addressed is data versioning. The RIVM publishes the number of confirmed cases on a daily basis, and if new data is not yet available, we should not re-run the NIPA algorithm. We can solve this issue by making use of the timestamp on the data. Before we query ArcGIS for the infection data, we can first send a query for the metadata. The metadata is a JSON file which contains some information about the dataset, and it includes an entry for the last edit date of the data. If we save the last observed timestamp to S3, then we can use it next time to check if new data from RIVM is available yet. Our completed backend carries out the following steps each time it runs:

1. Send a request to ArcGIS for the metadata file and extract the timestamp.
2. Send a request to S3 to retrieve the value of the last observed timestamp. If the timestamp matches the timestamp on ArcGIS, new data is not available yet. In this case, do not proceed any further.
3. If the timestamps do not match, new data is available. Send a request to ArcGIS for the data.
4. Preprocess the data according to the previously described steps.

5. Run NIPA on the preprocessed data to forecast the number confirmed cases.
6. Upload a public copy of the processed and forecasts to S3 where it can be accessed by the frontend. Also upload a private copy of the data to S3 for archiving purposes (Figure 2.6).
7. Write the new timestamp to S3.

Requesting the metadata file is very fast compared to requesting the data and the amount of overhead added is minimal. The metadata file is only several kilobytes in size and it takes less than a second to receive a reply. For comparison, the data is several megabytes in size and it takes several minutes before we receive the data from ArcGIS.

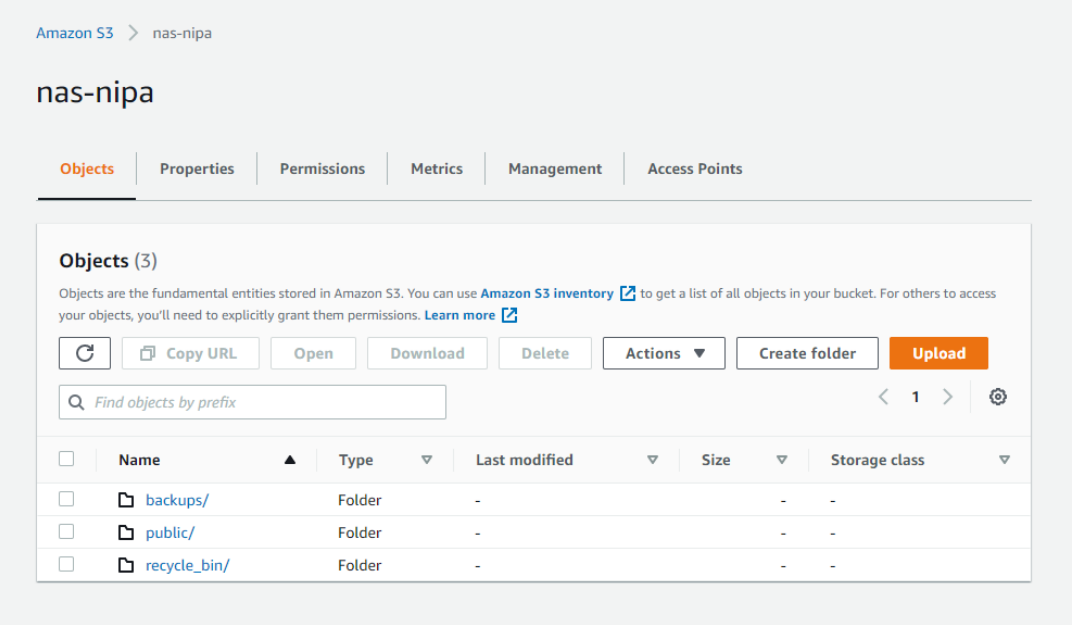


Figure 2.6: The S3 bucket is configured with three directories. The public directory contains the data which is accessible anywhere on the internet. The backups directory contains a record of all of the data we have uploaded to S3. The recycle bin directory is a temporary folder for files pending deletion.

The writing of the new timestamp to S3 is intentionally chosen to be the last step in the update process. In the event that something goes wrong and the backend terminates prematurely, the timestamp will not be saved to S3. There are several reasons why this may happen; for example, we observed occasionally that requests sent to the ArcGIS server would time out and we would not receive a reply, causing the backend to exit. Of course it is also possible that the backend crashes. By leaving the timestamp update until the end, we ensure that the timestamp is only written to S3 if all other actions have completely successfully. If something goes wrong, the timestamp will not be updated, and the next time the backend runs it will still see an outdated timestamp on S3; the backend will then try to update the data again.

We schedule a script on Heroku to run our backend on an hourly basis, so it checks hourly for new data on ArcGIS. After deploying the backend to Heroku, we perform two manual runs to ensure that it is functioning properly. On the first run, there is no timestamp stored on S3 yet, so it requests data from ArcGIS, runs NIPA, and writes the result and timestamp to S3. On the second run, the backend sees that the timestamp of ArcGIS matches the timestamp saved on S3, and the backend exits without performing any further actions. We have now set up a backend which will automatically update the data whenever RIVM publishes the daily number of infected individuals. We can now focus on making changes to the frontend of the website, which we will discuss in the next chapter.

3

Phase 2: Adapting the Website

3.1. English Translation

3.1.1. Setting up the Translation Framework

The most significant change made to the frontend of the website is the English translation. Although the actual translation is quite straightforward, we decided that we would like to preserve the Dutch version of the website as well and allow users the option to select their desired language. This required some significant changes to the structure of the frontend. To start with, we needed to install a framework for handling translations, and after some searching we found that the ngx-translate module for Angular should be suitable for our needs. The ngx-translate module also requires an HTTP loader for loading translations from files. Since the frontend was originally built in Angular version 9.1.3, we specifically installed ngx-translate version 12.1.2 because from the documentation, we know that this version is compatible. For the same reason, we chose to install version 5.0.0 of the HTTP loader.

```
npm i @ngx-translate/core@12.1.2
npm i @ngx-translate/http-loader@5.0.0
```

With the translation framework installed, the next step was to remove all of the text from each page of the website and replace it with a unique ID. We then created a JSON file called `nl.json` where we mapped each ID to the text that needs to be displayed. We then created a second JSON file called `en.json` where we map the same IDs to the English version of the text. Depending on the current language setting, the translation framework will load the corresponding JSON file and display the text in the selected language.

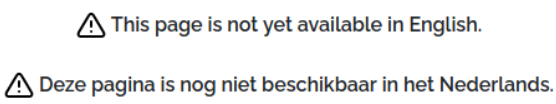


Figure 3.1: A page which is not available in the user's selected language will display one of the following warning messages. The bottom message is in Dutch and reads "This page is not yet available in Dutch."

In order to make things more convenient for the user, we implement a function that checks the language setting of the user's browser when then open our website. If their browser uses a language supported by our website (i.e. English or Dutch), the language of the website is automatically set to match their browser settings. In case the user's browser is set in another language, the website will default to English. In case there are pages on our website which are currently unavailable in the user's preferred language, the website will display the page in English. The user will be presented with a banner that informs them that the current page is not yet available in their language as shown in Figure 3.1.

3.1.2. Adding the Language Selection Option

We initially implement the language selection by adding a native HTML selection box in the header as shown in Figure 3.2. However, one issue that immediately arises is that if a user manually changes the language

setting, the chosen language does not persist when the user navigates to a different page on our website. Instead, the language reverts to the automatically set language (either matching their browser language or defaulting to English). We therefore need some mechanism to store the user's preferred language setting.

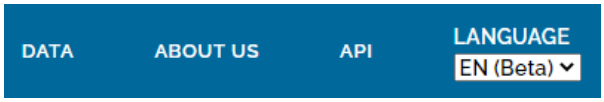


Figure 3.2: Shown on the far right is the native HTML selection box in the header for choosing the language.

In order to store the user's preferred language, we can use either `sessionStorage` or `localStorage` which are part of the web storage API. The difference between the two is that `sessionStorage` is cleared after the current window or browser tab is closed, while `localStorage` is persisted until it is manually cleared either by the user or our frontend. Additionally, `sessionStorage` applies only to the current browser tab, so opening the website in two tabs will create two different instances of `sessionStorage`; `localStorage` is shared between all tabs which are opened to our website. We ultimately decide to use `sessionStorage` for multiple reasons. Firstly, `sessionStorage` will be cleared automatically upon closing the tab so we will avoid storing unnecessary data on the user's device. Secondly, we find it unlikely that users will need to open our website in multiple tabs simultaneously so the lack of sharing in `sessionStorage` is not a major issue. Finally, we believe that the automatic language setting will already match the desired language of most users, so they will not need to change the language setting at all.

3.1.3. Improving the Language Selection Menu

The current language selection menu uses the native HTML selection box which is limited in functionality and we were unable to set the language on mobile devices. The selection box also does not really fit with the overall aesthetic of the website, and we would like to replace it with one that can display a flag for the selected language. The material select module provides a library for setting up more advanced drop-down menus that support images, and we installed the latest version (9.0.0 at the time).

```
npm i @material/select@9.0.0
```

After some tuning, we created a drop-down menu with a transparent background that displays the flag of the country next to the language as shown in Figure 3.3. The flag images that we used are in the public domain, so copyrights are not a concern. The new drop-down menu we created is also compatible with mobile devices, so mobile users now also have the option to select their preferred language.

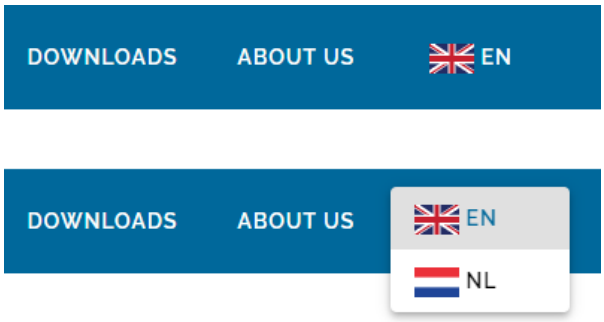


Figure 3.3: The updated language selection menu supports images and is compatible with mobile devices.

3.1.4. Translation of User-Interface Elements

While translating the text on the website is relatively straightforward with the ngx-translate library, translating some of the user-interface elements proves more difficult. This mainly applies to the graphs and sliders on the dashboard. It is not possible to translate the text on the slider after it has been created; and trying to do so causes the slider to break. We therefore modified the slider class to translate the tick labels before creating the slider.

Translating the dates with the ngx-translate module is not practical. Instead, we should localize the dates by setting the locale to use for date formats. We thus modify the code to match the locale to the users selected

language, using the date format of United Kingdom for English and the date format of the Netherlands for Dutch. Some of these user interface elements do not immediately update after setting the new language, so we resolve this by automatically reloading the website after the language is switched.

3.2. Modifications for the NAS Department

The original website was created by a group of bachelors students for their final project, and the "About Us" page of the website describes their project. However, this is not relevant to the NAS department, so we have fully reworked this page. The original content of the page has been condensed to a single section that acknowledges the original creators of the website. We have added a section that briefly introduces the NAS department its role at TU Delft. The links to the TU Delft website have been set up such that they will link to the English or Dutch version of the TU Delft page depending on the user's currently selected language. We have also added a section to credit those who were involved in the translation of the website. In the future, this can be easily extended if someone contributes to translate the website into another language. In addition to these changes, we also designed a stylized NAS logo to replace the original icon in the website header as shown in Figure 3.4.

Figure 3.4: The header has been updated with a stylized NAS logo that matches the design of the web page.

3.3. Replacing API with Downloads

The original backend had an API that users could query for data and the original website had a link in the header to the documentation of the API. Since the backend that we have created does not have an API, we have removed the link to the API and instead created a new "Downloads" page where we make our data available for direct download. On this page, we give a detailed explanation explanation of all of the preprocessing steps that have been performed on the data as we discussed in Section 2.4. We clearly explain where and how our data differs from the officially published numbers so that users are aware that the data has been preprocessed.

We provide download links to the data in .CSV format. The data is stored in our S3 bucket and is updated automatically when our backend processes the newly published figures from RIVM each day. As discussed in Section 2.4, we store the timestamp of the data to our S3 bucket as well so that we can check whether there is any new data. In the frontend, we retrieve this timestamp and display it on the downloads page so that users know when the data was last updated.

4

Phase 3: Validating the NIPA Implementation

4.1. Preliminaries

The Network-Inference-Based Prediction Algorithm (NIPA), proposed in 2020 by Prasse et al. [2], is an algorithm for forecasting epidemics. The original backend of the website created by the group of Computer Science students included a Python implementation of NIPA which was used to provide a prediction of the number of confirmed cases in the following days. We choose to create our own implementation of NIPA to gain a deeper understanding of the algorithm and also ensure that there are no mistakes in the original implementation. This chapter only gives a brief overview of the algorithm as the focus is on our implementation; for more details on NIPA, we refer the reader to [2].

4.1.1. The SIR Epidemic Model

In the SIR model, an individual exists in one of three states: susceptible, infectious, or removed. Susceptible individuals are healthy, but can become infected by coming into contact with an infectious individual. Infectious individuals can spontaneously become removed, after which they can no longer infect others or become infected again; there are different physical interpretations of this, for example, the individual could have been quarantined, the individual could have died, or the individual could have recovered and become immune to the virus.

We use the SIR model to model the spread of a virus between provinces, and we follow the theory of [2, pp. 3-6]. We have N provinces, and for every province i, the viral state vector at time k is given by

$$v_i[k] = \begin{pmatrix} S_i[k] \\ I_i[k] \\ R_i[k] \end{pmatrix} \tag{4.1}$$

The components $S_i[k]$, $I_i[k]$, and $R_i[k]$ denote the *fraction* of susceptible, infectious, and removed individuals in the province respectively; it thus follows that the sum of the components is 1. For each province, the fraction of infected individuals evolves according to

$$I_i[k+1] = (1-\delta_i) I_i[k] + S_i[k] \sum_{j=1}^{N} \beta_{i,j} I_j[k]. \tag{4.2}$$

The first term on the right-hand side of (4.2) shows the effect of infectious individuals recovering. The curing rate δ_i of the province i is the rate at which infectious individuals recover and become removed. The fraction of individuals at time k therefore becomes scaled by a factor $(1-\delta_i)$ at time $k+1$. The second term shows the effect of susceptible individuals becoming infected. The infection rate $\beta_{i,j}$ is the rate at which the infectious individuals of province j infect the susceptible individuals of province i. Each province $j \in N$ can infect the susceptible individuals of province i with the corresponding infection rate $\beta_{i,j}$. Note that the sum is taken over all cities j, including $j = i$; that is to say the infectious individuals of province i can also infect the susceptible individuals of province i.

For each province, the fraction of recovered individuals evolves according to

$$R_i[k+1] = R_i[k] + \delta_i I_i[k]. \tag{4.3}$$

An infectious individual which has recovered can no longer become infected again, so all of the recovered individuals from time k remain recovered at time $k+1$ as seen in the first term of (4.3). Additionally, the infectious individuals of time k can recover with rate δ_i, so at time $k+1$, the fraction of recovered individuals grows by $\delta_i I_i[k]$. From the condition that the sum of the components is 1, the fraction of susceptible individuals follows as

$$S_i[k+1] = 1 - I_i[k+1] - R_i[k+1]. \tag{4.4}$$

In the SIR model, it is assumed that the curing rate δ_i and the infection rate $\beta_{i,j}$ are constant over time.

4.1.2. The Least Absolute Shrinkage and Selection Operator (LASSO)

Given n observations of the viral state in a province i from time $1,...,n$, we can use the SIR equations (4.2) and (4.3) to construct a linear system

$$V_i = F_i \begin{pmatrix} \beta_{i,1} \\ \vdots \\ \beta_{i,N} \end{pmatrix}, \tag{4.5}$$

where V_i is given by

$$V_i = \begin{pmatrix} I_i[2] - (1-\delta_i) I_i[1] \\ \vdots \\ I_i[n] - (1-\delta_i) I_i[n-1] \end{pmatrix}, \tag{4.6}$$

and F_i is given by

$$F_i = \begin{pmatrix} S_i[1] I_1[i] & \cdots & S_i[1] I_N[i] \\ \vdots & \ddots & \vdots \\ S_i[n-1] I_1[n-1] & \cdots & S_i[n-1] I_N[n-1] \end{pmatrix}. \tag{4.7}$$

The linear system will only be satisfied if the viral state evolves exactly according to the SIR model, but for real data, this will not be the case. Therefore, the linear system only holds approximately, and the infection rates $\beta_{i,j}$ are instead estimated by minimizing the deviation of the left and right side of (4.5) using the least absolute shrinkage and selection operator (LASSO):

$$\min_{\beta_{i,1},...,\beta_{i,N}} \left\| V_i - F_i \begin{pmatrix} \beta_{i,1} \\ \vdots \\ \beta_{i,N} \end{pmatrix} \right\|_2^2 + \rho_i \sum_{j=1, j\neq i}^{N} \beta_{i,j} \tag{4.8}$$

such that the all of the infection rates $\beta_{i,j}$ satisfy the condition $0 \leq \beta_{i,j} \leq 1$ for all provinces $i, j \in N$ [2, pp. 6]. The value ρ_i is a regularization parameter to prevent overfitting, but as seen in the second term of (4.8), $\beta_{i,i}$ is excluded from the sum. In NIPA, it is intentionally chosen not to penalize $\beta_{i,i}$ because it is expected that the infection rate between individuals of the same province will be dominant. In the following sections, we will discuss how the curing rates δ_i and the regularization parameters ρ_i are set, and how the calculation of the infection rates $\beta_{i,j}$ is implemented.

4.2. Constructing the Linear System

As discussed in Section 2.4, after the preprocessing performed by our backend, we have the number of new confirmed cases per province per day. According to [2, pp. 5], the number of confirmed cases is first smoothed using MATLAB's `smoothdata` function. After looking at MATLAB's source code for the function, we determined that the default settings performs a moving average with a window of size 3. Because [2] calls the `smoothdata` function with th default settings, we also smooth the number of confirmed cases using a moving average with a window size of 3 in our implementation. We use the `rolling` function from the pandas

library; we call the function with arguments `center=True` and `min_periods=1`, which will center the window and ignore the part of the window that falls outside of the data range. We choose to implement it in this way because this is the same way that the MATLAB function `smoothdata` operates.

The number of new confirmed cases in a province i at time k is divided by the population of the province i, and the result is used as an estimate of the fraction of infected individuals in the province on that day $I_i[k]$ [2, pp. 3]. To construct the linear system in (4.5), we also need the fraction of susceptible individuals. Using the time series of infected individuals, we first obtain the fraction of recovered individuals by setting the initial fraction of recovered individuals to 0 and then iterating using (4.3). The fraction of susceptible individuals then follows from (4.4).

We additionally need the curing rate δ_i to build the linear system. However, we do not know what the true curing rate is, and in NIPA, we instead consider many candidate values for the curing rate and choose the one that produces the smallest error in the LASSO (4.8). Note that for different values of δ_i, we will obtain a different time series for the fraction of susceptible and recovered individuals. In the next section, we will discuss how we set the value of the curing rate δ_i and the regularization parameter ρ_i for each province i.

4.3. Setting the Curing Rate and Regularization Parameter

Before we can solve the LASSO (4.8) for $\beta_{i,j}$, we need to set the value of the curing rate δ_i and the regularization parameter ρ_i for each city i. Since we do not know the true value of δ_i, NIPA considers 50 candidate values for δ_i linearly spaced in the range $[0.01, 1]$. For the regularization parameter ρ_i, 100 candidate values logarithmically spaced in the range $[\rho_{\min,i}, \rho_{\max_i}]$ where $\rho_{\max_i}$ and $\rho_{\min,i}$ are given by

$$\begin{aligned} \rho_{\max,i} &= 2 \left\| F_i^T V_i \right\|_\infty, \\ \rho_{\min,i} &= 10^{-4} \rho_{\max,i}. \end{aligned} \tag{4.9}$$

To set the value of δ_i and ρ_i, k-fold cross-validation is performed; specifically, 3-fold cross-validation. The rows of the linear system in (4.5) are divided into 3 partitions. We then solve the LASSO on two of the partitions (the training dataset) and calculate the error on the third partition (the validation dataset). The validation error of is calculated using

$$\left\| V_{i,\text{val}} - F_{i,\text{val}} \begin{pmatrix} \beta_{i,1} \\ \vdots \\ \beta_{i,N} \end{pmatrix} \right\|_2^2, \tag{4.10}$$

where $V_{i,\text{val}}$ and $F_{i,\text{val}}$ are the partitions of the original matrices V_i and F_i that are reserved for validation in the current fold. This is repeated three times, each time using a different partition as the validation dataset.

For each combination of δ_i and ρ_i, we perform 3-fold cross-validation and calculate the average validation error over the three folds. We then choose the value of δ_i and ρ_i that produces the lowest error. We perform this process for each province i.

4.4. Solving the LASSO

After setting δ_i and ρ_i for every province, we solve the LASSO to determine the infection rates $\beta_{i,j}$ for all provinces i, j. We use the qpsolvers library in Python to do this. The solver is called with the function

```
solve_qp(P, q, G, h, A, b, lb, ub)
```

and it solves a convex quadratic program in the standard form given by

$$\min_x \frac{1}{2} x^T P x + q^T x, \tag{4.11}$$

subject to the conditions $Gx \leq h$, $Ax = b$, and $lb \leq x \leq ub$. In order to use this solver, we need to express our LASSO using the standard form. To start with, we can rewrite (4.5) as

$$\min_x \| V_i - F_i x \|_2^2 + \rho_i^T x \tag{4.12}$$

where x is the vector of infection rates we are trying to solve and ρ_i^T is a row vector with ρ_i at every position in the vector except for position i where the value is 0 (because we choose not to penalize $\beta_{i,i}$). We can expand the term $\| V_i - F_i x \|_2^2$ as follows:

$$\begin{aligned}\|V_i - F_i x\|_2^2 &= (V_i - F_i x)^T (V_i - F_i x) \\ &= V_i^T V_i - V_i^T (F_i x) - (F_i x)^T V_i + (F_i x)^T (F_i x) \\ &= V_i^T V_i - 2V_i^T (F_i x) + (x^T F_i^T)(F_i x) \\ &= V_i^T V_i - (2V_i^T F_i)x + x^T (F_i^T F_i)x.\end{aligned} \tag{4.13}$$

In standard form, we therefore have

$$\begin{aligned}P &= 2F_i^T F_i, \\ q &= \rho_i^T - 2V_i^T F_i.\end{aligned} \tag{4.14}$$

Our infection rates are subject to the condition $0 \leq \beta_{i,j} \leq 1$, so we set `lb` to a zeros vector and `ub` to a ones vector. For each province, we call `solve_qp` to solve the LASSO and find the infection probabilities $\beta_{i,j}$.

4.5. Forecasting

Having now solved δ_i and $\beta_{i,j}$ for all provinces, we can now use the SIR model to forecast the fraction of infectious individuals in the following days. As discussed in Section 4.2, we use the number of new confirmed cases in a province i divided by the population of the province as an approximation of the fraction of infected individuals. From (4.3) and (4.4), we calculate the fraction of recovered and susceptible individuals up until the current day for each province. We can then predict the fraction of infected individuals on the following day by iterating the SIR model using (4.2).

4.6. Comparison with the Original Implementation

After having completed our own implementation of NIPA, we compare our implementation with the one created by the Computer Science students. We determined that students have implemented the NIPA algorithm correctly, but some of the parameters they used are different from the ones specified in [2]. The implementation of the students uses 20 linearly spaced candidate values for the value of δ_i, compared to 50 in the paper; they use 20 logarithmically spaced candidate values for ρ_i compared to 100 candidate values in the paper. In addition, the original backend does not perform the initial data smoothing on the number of confirmed cases. Apart from these differences, the NIPA implementation created by the students is the same as described in [2].

5

Closing Remarks

The beginning of this project was challenging because I did not have any background in web design. I was not familiar with the tools of the trade; I had never heard of the Angular framework or the TypeScript language. Over the course of this project, I slowly became more familiar with Angular, starting by making small changes to the website and working my way up to adding language selection, and eventually I was able to add completely new pages. This project has exposed me to many new tools and concepts that I would not have known about otherwise. The problems I encountered when trying to deploy the website forced me to come up with creative solutions. Adapting the COVID-19 prediction website has been a valuable learning experience, and I hope to be able to apply the knowledge and skills I acquired during this project in the future.

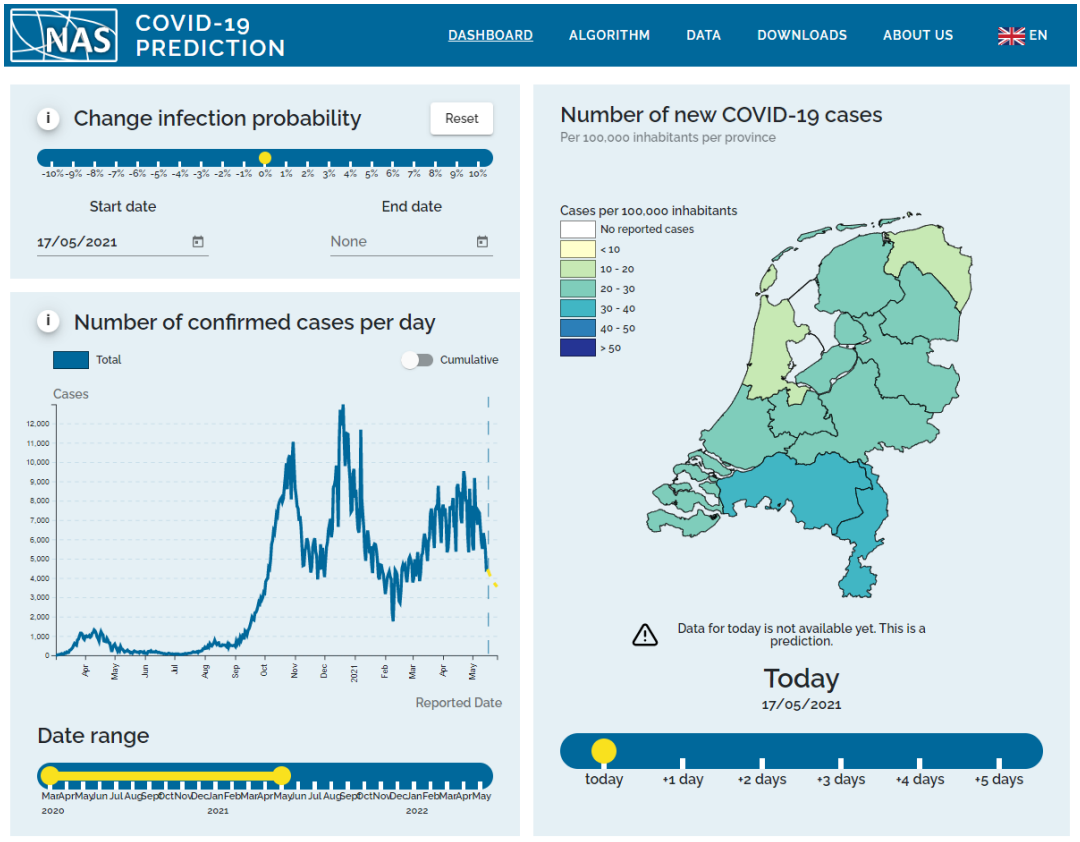


Figure 5.1: The final version of the website.

II

Applying a Markov-Modulated Process Model to Mobility Processes

1 Introduction

For many viruses, human-to-human transmission is a major transmission vector, and therefore, the spread of the virus is inextricably linked to human mobility. A clear understanding of the human mobility process is therefore crucial if one hopes to understand the spread of the virus. With the ongoing COVID-19 pandemic, this topic is more relevant than ever, and many studies into the connection between the pandemic and human mobility have been conducted [6] [7] [8].

A novel idea for modeling mobility processes has recently been proposed, namely the Markov-modulated process model. Using a simulated mobility process represented as a series of adjacency matrices, Yang [1] has developed a Markov-modulated process (MMP) model that is capable of capturing some of the key properties of the mobility process with a very high accuracy. On average, Yang's model produces a graph where the average number of links at any time matches that of the mobility process. Furthermore, the average number of links which are added and removed between two time steps also matches that of the mobility process, so the average rate at which the graph evolves between two time steps is the same as in the mobility process.

The second part of this thesis builds on Yang's research and applies the MMP model to a simulated mobility process with a larger number of nodes. The structure of the second part of the thesis is as follows. In Chapter 2, the Markov-modulated process model is introduced, specifically focusing on the aggregated MMP model developed by Yang. The quantized MMP model is proposed as a modification of the aggregated MMP model. In Chapter 3, the simulated mobility process that will be modeled is described. In Chapter 4, the quantized MMP model is applied to model the mobility process and the results are compared with the aggregated MMP model. In Chapter 5, SIR simulations are performed on the quantized MMP model, aggregated MMP model, as well as the mobility process to investigate differences in the spreading behavior. Finally, Chapter 6 concludes the thesis by presenting the main findings and suggesting some directions for future research.

2

The Markov Modulated Process Model

2.1. Introduction

The Markov-modulated process (MMP) model has recently been proposed as a tool for modeling mobility processes. In a recent master's thesis, Yang [1] has taken the first step in investigating how the MMP model can be applied to model a mobility process. This chapter provides an introduction to the MMP model, the current state of the research, and the research goals of this thesis. The outline of this section is as follows. Section 2.2 begins by providing a definition of the MMP model. Section 2.3 summarizes the key findings of Yang's thesis and discusses the two MMP models developed by Yang: the unaggregated and aggregated MMP model. Section 2.4 describes the quantized MMP model, which is a modification of the aggregated MMP model that will be investigated in this thesis. Finally, Section 2.5 discusses the main research goals of this thesis.

2.2. Definition of the MMP Model

Consider a temporal graph G where the links in the graph can change over time. This temporal graph could, for example, represent the contacts in a network of people that are moving around (in other words, a mobility process). Each node in the graph represents a person, and a link in the graph represents a contact between two people. Each person in the mobility process is able to move, and when two people come into contact with each other, the corresponding link in the graph G is created. When they move away from each other and are no longer in contact, the corresponding link is removed.

The Markov-modulated process (MMP) model is proposed as a tool for modeling mobility processes by modulating a temporal graph. Consider again a temporal graph G which is now modulated by a Markov process. Each state of the Markov process has an associated action, and when the Markov process enters a state, the corresponding action is applied to modulate the graph. When constructing an MMP model, there are three main challenges involved. Firstly, the number of states needs to be chosen. Secondly, an action needs to be chosen for each state (and there are an infinite number of possible actions to choose from). And finally, the transition probability between each pair of states needs to be set.

In the MMP model developed by Yang, actions are defined as jointly adding and removing a number of links from the temporal graph. The transition probabilities are derived from the observed transitions in the simulated mobility process, and the number of states is related to the number of nodes in the graph. The next section discusses the model developed by Yang and highlights the key results of her research.

2.3. The Link-Aware MMP Model

In this section, the MMP model developed by Yang [1] is introduced and the key results of the model will be discussed. The MMP models developed by Yang are referred to as "link-aware" MMP models because the states of the MMP model additionally encode information about the number of links that exist in the graph. Yang has researched two variations of the link-aware MMP model, which are referred to as the "unaggregated" and "aggregated" MMP model. In the unaggregated MMP model, each state of the MMP model encodes the number of links which currently exist in the graph, as well as an action to apply to modulate the graph. Therefore, there are multiple states which encode the same number of links, but have different actions associated

with them. The main drawback of the unaggregated MMP model is the explosion in the number of states as the number of nodes N increases: the required number of states in the unaggregated MMP model increases with $O(N^6)$. Yang therefore introduces the concept of state aggregation to reduce the number of states in the Markov process, resulting in the aggregated MMP model. The number of states in the aggregated MMP model increases with $O(N^2)$, a significant improvement compared to the unaggregated MMP model. Yang demonstrated that when applied to the simulated mobility process, the aggregated MMP model produces results which are indistinguishable from the unaggregated MMP model.

The structure of this section is as follows. In Section 2.3.1, the data produced by the simulated mobility process used by Yang is described, and the goal of the modeling process is defined. Section 2.3.2 describes the actions that are defined in Yang's model and how the performance of the MMP model is evaluated. Section 2.3.3 discusses the first version of the link-aware MMP model created by Yang: the unaggregated MMP model. Section 2.3.4 discusses the aggregated MMP model that Yang developed to address the issue of the state explosion in the unaggregated MMP model. Finally, Section 2.3.5 discusses the modeling accuracy of the unaggregated and aggregated MMP models.

2.3.1. The Simulated Mobility Process

In Yang's research, an MMP model is developed for a simulated mobility process [1, Ch. 2]. This section gives a description of the data produced by the simulated mobility process that is used for creating the MMP model; the details of how the simulation is conducted will be discussed in Chapter 3. The simulated mobility process consists of N nodes which are confined to move within a given region so that the number of nodes does not change over the course of the simulation. Each trial of the simulated mobility process produces a contact network G which is represented by a time-varying adjacency matrix $A(t)$ changing at each discrete time t. Each element of the matrix $a_{i,j}(t)$ describes if there is a contact between node i and node j at time t. A contact between node i and node j at time t is indicated by $a_{i,j}(t) = 1$, and no contact is indicated by $a_{i,j}(t) = 0$.

As described in [1, Sec. 3.2], the goal of the modeling process is to construct an MMP model which, when applied to modulate a graph of N nodes, produces a series of adjacency matrices which "resemble" the series of adjacency matrices from the simulated mobility process. That is, the adjacency matrices of the MMP model should have properties which match those of the mobility processes. The next section describes how Yang defines the actions of the MMP model and how the similarity between the MMP model and mobility process is evaluated.

2.3.2. Defining Actions

As mentioned previously, one of the main challenges in creating an MMP model is choosing the number of states and choosing an action for each state. There are an infinite number of possibilities for the actions that can be chosen. Some possible actions are:

1. Add 1 link to the graph.
2. Remove 1 link from the graph.
3. Add 5 links and remove 3 links from the graph.
4. Add all possible links to the graph (i.e. add links until the complete graph is reached).
5. Remove each link that currently exists with probability $p = 0.1$.
6. Generate an Erdős-Rényi graph with link probability $p = 0.5$.
7. Remove half of the nodes from the graph.
8. Do nothing.

In Yang's research, every action of the MMP model is defined as jointly adding and removing a number of links from the graph (like in the third example of the list). The number of nodes in the simulated mobility process is fixed, so adding and removing nodes has not been considered.

The simulated mobility process used by Yang is described by a series of adjacency matrices in discrete time, and therefore Yang also uses an MMP model with a discrete Markov process. The MMP model is designed to follow the same time scale as the simulated mobility process, so one time step in the Markov process

corresponds to one time step in the simulated mobility process. At each time step in the mobility process, links can be added as well as removed in the mobility process, which is why Yang considers actions as a joint combination of adding and removing a certain number of links. Yang mainly evaluates the similarity between the adjacency matrices of the graph modulated by the MMP model and the adjacency matrices of the simulated mobility process through three link metrics:

- Average number of links at each time step.
- Average number of links added between two time steps.
- Average number of links removed between two time steps.

The average number of links at each time step is used to evaluate the long-term behavior of the MMP model compared to the mobility process; in the long-term, the average number of links in the MMP-modulated graph should be the same as the mobility process. The average number of links added and removed between two time steps is used to evaluate the short-term behavior of the MMP model; this captures how fast the graph is evolving at each time step.

Yang's initial investigation considered an MMP model where the states did *not* encode the number of links in the graph, so each state simply corresponded to the action of adding and removing a certain number of links from the graph.[1] Although this first model produced a modulated graph where the average number of links added and removed between two time steps matched the mobility process, the average number of links at each time step did not match the mobility process. This initial MMP model led to the key observation that, when considering actions as adding and removing a number of links from the graph, it is necessary for the MMP model to encode information about the number of links which currently exist in the graph. Without this information, there can be no form of feedback, and it is not possible for the MMP model to maintain the same average number of links as the mobility process. This led Yang to develop link-aware MMP models, where the states of the Markov process encode the number of links in the graph. The first version of the link-aware MMP model developed by Yang is referred to as the unaggregated MMP model and will be discussed in the next section.

2.3.3. The Unaggregated MMP Model

In the unaggregated MMP model, each state encodes the number of links in the graph as well as an action. Given a graph with N nodes, the maximum number of links that can exist in the graph is $L_{\max} = N(N-1)/2$. The number of links in the graph can thus range from 0 to $L_{\max}$. For each possible number of links, all possible actions need to be modeled. Yang defines actions as jointly adding and removing a certain number of links from the graph. The action of adding x links and removing y links is denoted as $(+x,-y)$. Assuming L links currently exist in the graph, the maximum number of links that can be removed is bounded by L because it is not possible to remove more links than exist. The maximum number of links that can be added is $L_{\max} - L$, because it is not possible to add more links once the complete graph is reached. Considering all possible combinations of added and removed links results in $(L+1)(L_{\max} - L + 1)$ actions; the +1 terms are due to the fact that it is also possible to add or remove 0 links. The number of states $\#X$ in the unaggregated MMP model can be found by summing the number of possible actions over all possible numbers of links:

$$\#X = \sum_{L=0}^{L_{\max}} (L+1)(L_{\max} - L + 1). \tag{2.1}$$

As proven in [1, App. A], the number of states can be expressed precisely in terms of N:

$$\#X = \frac{N^6}{48} - \frac{N^5}{16} + \frac{5N^4}{16} - \frac{25N^3}{48} + \frac{7N^2}{6} - \frac{11N}{12} + 1. \tag{2.2}$$

The number of states in the unaggregated MMP model therefore increases with $O(N^6)$.

The procedure for constructing the probability transition matrix of the unaggregated MMP model is as follows. The mobility process generates a time-varying series of adjacency matrices. At time t, the current number of links in the graph can be found from the adjacency matrix $A(t)$. The adjacency matrix $A(t)$ is then compared with the adjacency matrix at the next time step $A(t+1)$ to find the number of links which are added and removed (the action). Given the number of links at time t and the action, the Markov state at time

[1] Details on how the actions and transition probabilities of Yang's first MMP were derived can be found in [1, Sec. 3.4].

t follows immediately. Repeating this process for all t, the series of adjacency matrices can be transformed into a series of Markov states.

The series of Markov states can then be used to calculate the probability transition matrix. To find the transition probability from state i to state j, the observed number of transitions from state i to state j is divided the observed number of transitions from state i to any other k:

$$p_{i,j} = \frac{n_{i,j}}{\sum_{k \in X} n_{i,k}}, \tag{2.3}$$

where $p_{i,j}$ is the transition probability from state i to state j, $n_{i,j}$ is the number of observed transitions from state i to state j, and X is the set of all states in the Markov process.

The major drawback of the unaggregated MMP model is the explosion in the number of states as N increases. Therefore, Yang introduces the concept of state aggregation to reduce the number of states, resulting in the aggregated MMP model. The derivation of the aggregated MMP model is discussed in the next section.

2.3.4. The Aggregated MMP Model

The aggregated MMP model was proposed by Yang as a method to address the explosion in the number of states in the unaggregated MMP model. In the aggregated MMP model, actions are decoupled from the states so that the states encode only the number of links in the graph. For a graph of N nodes, the maximum number of links that can exist is $L_{\max} = N(N-1)/2$. The number of states in the unaggregated MMP model is thus $L_{\max} + 1$, where the $+1$ is due to the fact that one additional state is required to model 0 links. The number of states in the aggregated MMP model therefore increases with $O(N^2)$.

In the aggregated MMP model, actions are no longer associated with states and are instead tied to state transitions. For each state transition from a state i to state j, an action set $J_{i,j}$ is defined. The action set $J_{i,j}$ is the set of all possible actions, given that i links currently exist in the graph, that will result in a transition to j links. Each action set has its own probability distribution; when the Markov process transitions from state i to state j, an action f is randomly chosen from the corresponding action set $J_{i,j}$ based on the probability distribution of the action set.

The procedure for constructing the probability transition matrix of the aggregated MMP model is similar to that of the unaggregated MMP model. The mobility process generates a time-varying series of adjacency matrices. In the aggregated MMP model, the Markov state corresponds directly to the number of links in the graph. The series of adjacency matrices can be directly transformed into a series of Markov states by finding the number of links in each adjacency matrix. From the series of Markov states, the transition probabilities are calculated in the same way as before using eq. (2.3).

For each state transition from state i to state j, the probability distribution of the action set $J_{i,j}$ needs to be determined. For each action $f \in J_{i,j}$, the probability of choosing action f is given by

$$p(f)_{i,j} = \frac{n(f)_{i,j}}{n_{i,j}}, \tag{2.4}$$

where $p(f)_{i,j}$ is the probability of choosing action f from the action set $J_{i,j}$, $n(f)_{i,j}$ is the number of times that action f was observed when transitioning from state i to state j, and $n_{i,j}$ is the number of times that a transition from state i to state j was observed.

Yang showed that the aggregated MMP model is directly related to the unaggregated MMP model and could be derived from the unaggregated MMP model using a state aggregation method based on the steady state probabilities of the Markov process [1, Sec. 4.3.3]. Yang proved that the aggregated MMP model will produce a modulated graph that has the same number of links on average as the unaggregated MMP model. Furthermore, the long-term rate at which each action is applied in the aggregated MMP model matches the long-term rate at which the action is applied in the unaggregated MMP model. However, the state aggregation results in the loss of the dependence between consecutive actions. In the aggregated MMP model, actions have been decoupled from the states so the actions are only dependent on the current number of links in the graph. In the unaggregated MMP model, each state encodes an action as well as the current number of links in the graph, and therefore the next action is dependent on both the current number of links in the graph as well as the previous action.

2.3.5. Modeling Accuracy of the Unaggregated and Aggregated MMP Model

Yang's investigation compared the performance of the unaggregated and aggregated MMP model when applied to a simulated mobility process with 9 nodes. For both MMP models, the average number of links at

each time step as well as the average number of links added and removed between two time steps differed from the mobility process by less than 1%. Furthermore, the unaggregated MMP model produced results which were indistinguishable from the aggregated MMP model despite requiring far fewer states. The main deficit of the aggregated MMP model compared to the unaggregated MMP model is that the aggregated model does not take into account dependence between consecutive actions. Yang quantified the dependence between consecutive actions in the mobility process and found there was almost no dependence, which explains why the aggregated MMP model could produce results which are just as good as the unaggregated MMP model.

2.4. The Quantized MMP Model

As a method to further reduce the number of states, this thesis explores the idea of a quantized MMP model as a modification of the aggregated MMP model developed by Yang. In the aggregated MMP model, each state encodes the number of links in the graph and to cover all possible numbers of links, $L_{\max}+1$ states are needed. In the quantized MMP model, the number of links is rounded to the nearest multiple of q, where $q \in \mathbb{N}_{>0}$ is the quantization step size. This reduces the number of states by a factor of q.

As in the aggregated MMP model, the actions of the quantized MMP model are not tied to states, and instead an action set $J_{i,j}$ is defined for each transition from a state i to state j. However, an important issue arises: after rounding the number of links in the graph, the original action that was applied in the mobility process may no longer be valid. For example, consider the situation where the number of links at time t is $L_t = 5$, the number of links at time $t+1$ is $L_{t+1} = 8$, and the quantization step size is $q = 5$. Suppose that the original action when going from time t to time $t+1$ is $f = (+5, -2)$; the change in the number of links between time t and $t+1$ is $\Delta\hat{L} = +3$ and it is clear that after applying the action f, the number of links in the graph will increase by 3.

However, after quantization, the number of links at time t is $\hat{L}_t = 5$, and the number of links at time $t+1$ is $\hat{L}_{t+1} = 10$. The change in the number of links between t and $t+1$ now becomes $\Delta\hat{L} = +5$: the number of links in the graph increases by 5 between these two time steps. The original action f is no longer valid because it only increases the number of links by 3 (which would cause the number of links in the graph to leave the defined quantization levels). Therefore, the action needs to be modified so that it produces the required change in the number of links.

Compared to the aggregated MMP model, constructing the quantized MMP model requires two additional steps. The first is *link quantization*, where the number of links in the mobility process needs to be restricted to multiples of the quantization step size q. The second is *action remapping*, where the original actions of the mobility process need to be modified so that they produce the required net change in the number of links. The details of how these two steps are implemented will be discussed in Section 2.4.1 and Section 2.4.2 respectively.

2.4.1. Link Quantization

The quantization step size is defined as in integer q, and the number of links in the MMP model is restricted to multiples of q. The number of links at each time step in the mobility process can take any integer in the range $[0, L_{\max}]$ and needs to be quantized. Given that there are L links in the graph, the quantized number of links $\hat{L}$ is calculated using

$$\hat{L} = q \left\lfloor \frac{L}{q} \right\rceil, \tag{2.5}$$

where the operator $\lfloor x \rceil$ denotes the *round to even* function. Decimals which are smaller than 0.5 are rounded down, and decimals which are larger than 0.5 are rounded up. In the event that the decimal is *exactly* equal to 0.5, x is rounded such that the result is an even number (so 1.5 and 2.5 would be rounded to 2, while 3.5 and 4.5 would be rounded to 4).[2]

Link quantization reduces the number of states by a factor of approximately q. The number of states in the quantized MMP model is given by

$$\#X = \left\lfloor \frac{L_{\max}}{q} \right\rceil + 1. \tag{2.6}$$

[2]This also happens to be the behavior of the native rounding function in Python, which is the programming language used in this research.

Due to the +1 term and the rounding, the factor by which the number of states is reduced will not be exactly equal to q, but for $L_{\max} \gg 1$, the difference is negligible.

2.4.2. Action Remapping

After link quantization, the actions need to be remapped such that they produce the required change in the number of links. To avoid bias in the remapping of actions as much as possible, the following strategy is used to remap actions in a fair way:

1. Whenever possible, offset the number of added and removed links equally. For example, if the link delta after quantization is $\Delta\hat{L} = +5$ while the original link delta is $\Delta L = +3$, increase the number of added links by 1 and decrease the number of removed links by 1. Conversely, if $\Delta\hat{L} = +5$ and $\Delta L = +7$, decrease the number of added links by 1 and increase the number of removed links by 1.
2. If it is not possible to offset the number of added and removed links equally and $\Delta\hat{L} > \Delta L$, favor increasing the number of added links whenever possible.
3. If it is not possible to offset the number of added and removed links equally and $\Delta\hat{L} < \Delta L$, favor increasing the number of removed links whenever possible.

Note that it is not always possible to remap actions according to this strategy, because the remapped actions must still respect the fundamental limits of the graph: links cannot be removed when the graph is empty, and links cannot be added if the complete graph is reached. Three examples of action remapping are described below. It is assumed that the quantization step size is $q = 5$ in all of the examples.

Example 1: The number of added and removed links can be offset equally.
The number of links at time t is $L_t = 5$, and the number of links at time $t+1$ is $L_{t+1} = 8$. The original action when going from time t to $t+1$ is $f = (+5, -2)$. After quantization, the number of links at time t and time $t+1$ is $\hat{L}_t = 5$ and $\hat{L}_{t+1} = 10$ respectively. The original action is no longer valid because $\Delta L = +3$ while $\Delta\hat{L} = +5$. It is possible to offset the number of added and removed links equally, so the number of added links is increased by 1 and the number of removed links is decreased by 1. The new action is $\hat{f} = (+6, -1)$.

Example 2: The number of added and removed links cannot be offset equally.
The number of links at time t is $L_t = 6$, and the number of links at time $t+1$ is $L_{t+1} = 8$. The original action when going from time t to $t+1$ is $f = (+3, -1)$. After quantization, the number of links at time t and time $t+1$ is $\hat{L}_t = 5$ and $\hat{L}_{t+1} = 10$ respectively. The original action is no longer valid because $\Delta L = +2$ while $\Delta\hat{L} = +5$. It is not possible to offset the number of added and removed links equally because the difference between $\Delta\hat{L}$ and ΔL is not even. Since $\Delta\hat{L} > \Delta L$, increasing the number of added links will be favored, so the number of added links is increased by 2 and the number of removed links is reduced by 1. The new action is $\hat{f} = (+5, -0)$.

Example 3: The number of added and removed links cannot be offset equally because it would result in an invalid action.
The number of links at time t is $L_t = 2$, and the number of links at time $t+1$ is $L_{t+1} = 5$. The original action when going from time t to $t+1$ is $f = (+5, -2)$. After quantization, the number of links at time t and time $t+1$ is $\hat{L}_t = 0$ and $\hat{L}_{t+1} = 5$ respectively. The original action is no longer valid because $\Delta L = +3$ while $\Delta\hat{L} = +5$. Trying to offset the number of added and removed links equally would result in the action $\hat{f} = (+6, -1)$. However, this action is also not valid because $\hat{L}_t = 0$ and it is not possible to remove links from the empty graph. Therefore, the new action can only be $\hat{f} = (+5, -0)$.

2.4.3. Calculating Transition Probabilities and Action Set Probability Distributions

The procedure for constructing the probability transition matrix of the aggregated MMP model is almost the same as that of the aggregated MMP model. The mobility process generates a time-varying series of adjacency matrices. In the quantized MMP model, each state encodes a number of links that is a multiple of q. The series of adjacency matrices is transformed into a series of Markov states by finding the number of links in each adjacency matrix and then rounding to the nearest multiple of q. From the series of Markov states, the transition probabilities are calculated in the same way using eq. (2.3).

After the action remapping step, all of the actions are now valid, and the procedure for calculating the probability distribution for each action set is identical to that of the aggregated MMP model and is calculated using eq. (2.4).

2.5. Research Goals

There are three main research goals in the second part of this thesis. The first is to investigate the performance of the MMP model when applied to a larger number of nodes. Yang's investigation primarily focused on developing the link-aware MMP model and investigating differences between unaggregated and aggregated MMP models. Due to the fact that the number of states in the unaggregated MMP model grows with $O(N^6)$, Yang chose a small number of nodes $N = 9$. In this thesis, the number of nodes will be increased to 1,024.

The maximum number of links which can exist in a graph of 1,024 nodes is $L_{\max} = 523,776$, and in principle, the aggregated MMP model requires $L_{\max} + 1$ states to model a mobility process of $1,024$ nodes. In practice, many of these states will be empty because they will not be observed in the mobility process, but nevertheless, this raises an interesting point about the scalability of the MMP model: for a large mobility process with many links, is it necessary to have states which correspond to every integer number of links? Or can a high model accuracy still be achieved while using a smaller number of states? This forms the basis of the second research goal, where performance of the quantized MMP model will be compared with that of the unaggregated MMP model.

The third area that will be investigated is the comparison of the SIR spreading dynamics on the modulated graphs produced by the MMP model with the graphs of the mobility process. Some preliminary SIR results were presented by Yang for the aggregated MMP model of 25 nodes, and the goal is to perform a more in-depth investigation with a larger number of nodes to investigate the differences between the SIR dynamics of the MMP models with the mobility process and whether the quantization has any impact on the SIR dynamics.

The first step in this research is to generate the simulated mobility process to be modeled, and this will be discussed in the next chapter.

3

The Simulated Mobility Process

3.1. Introduction

This chapter describes the simulated mobility process that will be modeled. The number of nodes in the simulated mobility process will be increased to 1,024 to investigate how the MMP model performs with a larger number of nodes. To allow for a fair comparison of the results, the simulation will be carried out using the same methodology used by Yang [1, Ch. 2]. Section 3.2 describes the logic that controls the simulation and Section 3.3 defines the simulation parameters. Section 3.4 presents the results of the simulated mobility process.

3.2. Simulation Methodology

3.2.1. Starting Positions

The simulated mobility process consists of N nodes which are constrained to move in a square-shaped region with dimensions $W \times W$ units, referred to as the simulation region. The number of nodes N is chosen to be a square number, so there is some n for which $n^2 = N$. Initially, the nodes are spaced evenly in a square lattice with n rows and n columns. An example illustrating the initial positions for $N = 25$ nodes is shown in Figure 3.1. Nodes are represented by blue circles and the simulation region is represented by the green square.

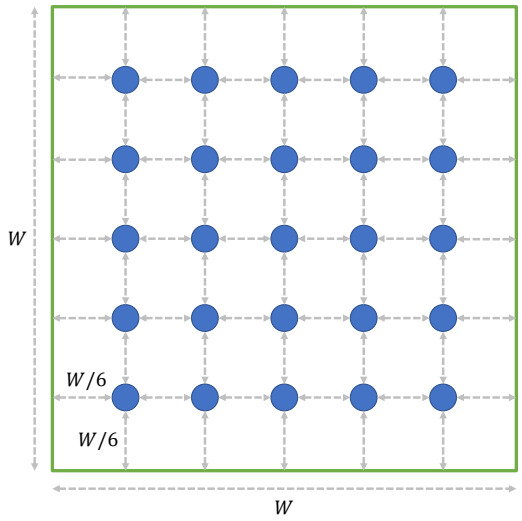


Figure 3.1: Initial positions of the nodes in the simulated mobility process. Nodes are represented by blue circles and the simulation region is represented by the green square [1, Fig. 2.1].

Note that the illustration is not to scale because nodes in the simulation are treated as dimensionless points. The separation between the rows and columns of the starting lattice is $W/(n+1)$, and the distance between the edges of the lattice and the borders of the region is also $W/(n+1)$.

3.2.2. Node Movement

During the simulation, the position of each node is represented by a pair of Cartesian coordinates, where the bottom left corner of the simulation region is defined as having coordinates $(0,0)$. The coordinates use the same units as the dimension of the region, so the x and y coordinates both lie in the range $[0,W]$. The coordinates are rounded to three decimal places. Each node also has a direction which is represented by an integer in the range $[0,359]$. The direction is interpreted as an angle in degrees and follows the same convention as the unit circle, where East is 0 degrees and the angle increases counterclockwise.

The mobility process is simulated in discrete time. At each time step, a node can change its current direction by a random amount, but limited to $\pm\theta$ degrees. The direction change is also restricted to be an integer and is uniformly distributed in the range $[-\theta,\theta]$. After changing the direction, each node can move a random distance along its current direction. The move distance is a real number uniformly distributed in the range $[0,2v]$. The move distance is not constrained to be an integer, but the coordinates of the node after moving are rounded to three decimal places. Nodes move independently of one another and are treated as dimensionless points, so they do not collide with other nodes.

3.2.3. Out of Bounds Handling

During the simulation, nodes are constrained to move within the simulation region. If a node attempts to move out of bounds, a series of steps are applied to bring the node back into the region. An illustration of this process is shown in Figure 3.2.

After a node has selected the move distance, the new position of the node after moving is calculated and a check is performed to see if both the x and y coordinates are in the range $[0,W]$. If either coordinate fails this check (Figure 3.2a), a series of steps are applied:

1. To keep the node inside the region, the coordinates are clamped to the range $[0,W]$ (Figure 3.2b). Coordinates less than 0 are set to 0, and coordinates greater than W are set to W. Coordinates which lies in the range $[0,W]$ are not changed.

2. The node is now on the border of the region, but the direction is still pointing "outwards" away from the region. Therefore, the direction of the node is reset to avoid the node trying to move out of bounds again in the next time step. The node is reset to face the center of the region; the new direction of the node is a vector from the node's current position to the point $(W/2,W/2)$ (Figure 3.2c). The direction is rounded to the nearest integer.

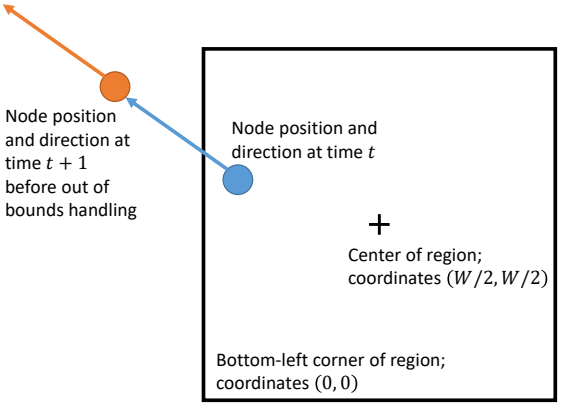
3. To add some randomness, an offset of up to $\pm\phi$ degrees is applied to the new direction. The offset is an integer uniformly distributed in the range $[-\phi,\phi]$ (Figure 3.2d).

The simulated mobility process also applies the out of bounds handling in the event that a node happens to land *exactly* on the border. In this case, the new position of the node stays the same, but the direction will be reset according to steps 2 and 3.

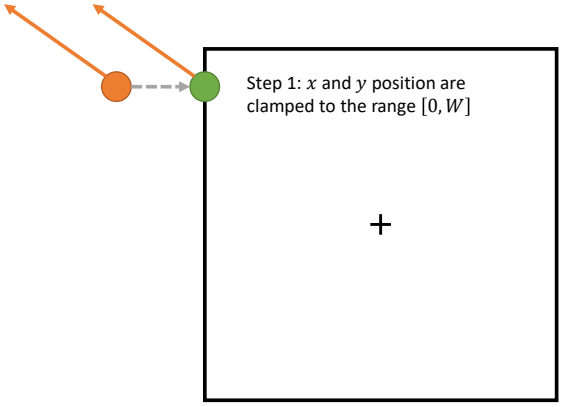
3.2.4. Simulation Procedure

The mobility process simulation is in discrete time and the simulation starts at time $t=0$ where the nodes are initialized in an evenly distributed square lattice as described in Section 3.2.1. Each node is also initialized with a random direction uniformly distributed in the range $[0,359]$. At each time step t, the nodes move according to the procedure described in Section 3.2.2 and the out of bounds handling described in Section 3.2.3 is applied to keep the nodes inside the simulation region.

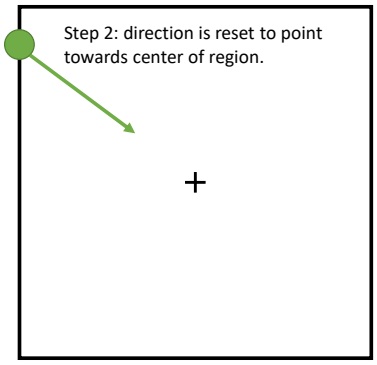
After moving the nodes, the contact matrix $A(t)$ is constructed. Two nodes are considered to be in contact when the distance between them is less than the distance threshold d. Each element of the matrix $a_{i,j}(t)$ describes if there is a contact between node i and node j at time t. A contact between node i and node j at time t is indicated by $a_{i,j}(t)=1$, and no contact is indicated by $a_{i,j}(t)=0$. The simulation is carried out for T time steps, which produces a series of $T+1$ contact matrices from $A(0)$ to $A(T)$.



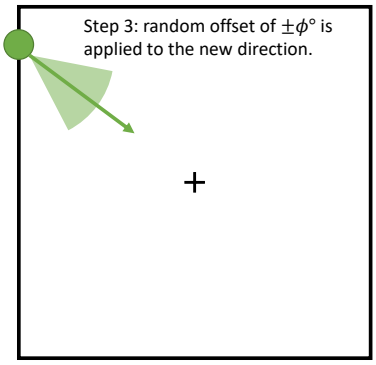
(a) A node's new position will be out of bounds.



(b) The coordinates of the node's new position are clamped to the range $[0, W]$.



(c) The node's direction is reset to point to the center of the simulation region.



(d) A random offset of up to $\pm\phi$ degrees is applied to the new direction of the node.

Figure 3.2: Illustration of the steps for preventing nodes from going out of bounds [1, Fig. 2.2].

3.3. Simulation Parameters

The maximum number of nodes considered in Yang's investigation was 25 [1, Sec. 2.5]. One of the goals of this thesis is to investigate how the MMP model performs with a larger number of nodes; to this end, the number of nodes in the mobility process is increased to $N = 1{,}024$. To accommodate the larger number of nodes, the size of the simulation region is scaled accordingly. Yang's investigation used a simulation region of size $W = 10$ units; the nodes are initially spaced in a 5-by-5 square lattice and the separation between the rows and columns of the lattice is $10/6 \approx 1.667$ units. With 1,024 nodes, the square lattice will have 32 rows and columns; to maintain the same spacing, the size of the region is therefore increased to $W = 55$ units.

To allow for a fair comparison of the results, the remaining simulation parameters are kept the same as those used by Yang [1, Sec. 2.5]. The maximum angle change in a time step is $\theta = 20$ degrees, and the maximum angle offset for the out of bounds handling is $\phi = 30$ degrees. The distance threshold is chosen to be $d = 1.5$ units, and the simulation duration is $T = 1{,}000$ time steps.

3.4. Simulation Results

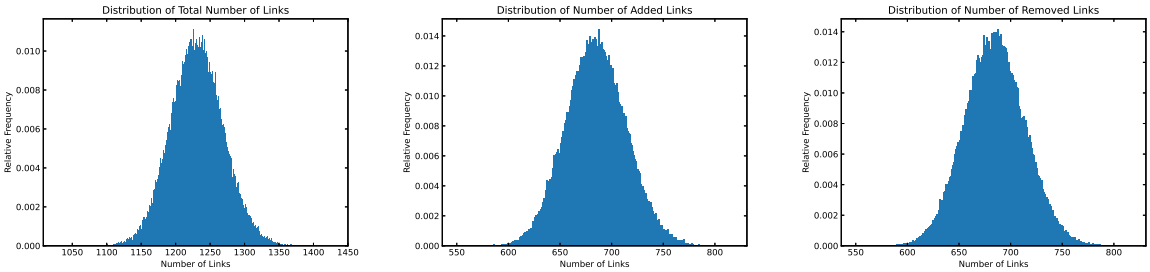
In total, 100 trials of the mobility simulation are performed. At each time step of each trial, the number of links in the graph is calculated, and between each pair of consecutive time steps, the number of links added and number of links removed is calculated. The distributions are plotted in Figure 3.3[1] and the averages values are as follows:

- Average number of links at each time step: 1,229.07.

[1] Larger versions of the plots of the link distributions of the mobility process can be found in Appendix A.1.

- Average number of links added between two time steps: 684.66.
- Average number of links removed between two time steps: 683.42.

The number of links at each time step for one trial of the mobility process is shown in Figure 3.4.



(a) Distribution of the total number of links at each time step.

(b) Distribution of the number of links added between two time steps.

(c) Distribution of the number of links removed between two time steps.

Figure 3.3: Histograms of link statistics for the simulated mobility process across 100 trials of 1,000 time steps.

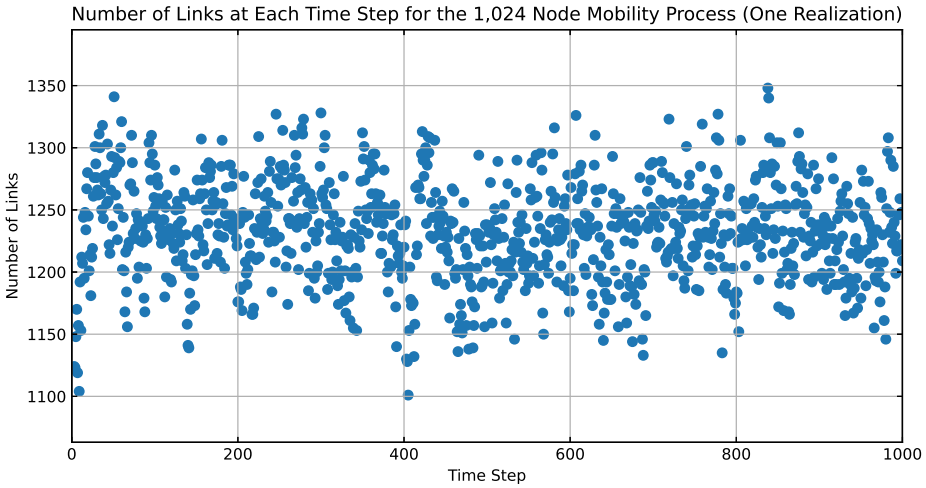


Figure 3.4: Number of links at each time step for one trial of the simulated mobility process.

4

Modeling the Simulated Mobility Process

4.1. Introduction

In this chapter, different MMP models will be used to model the simulated mobility process of 1,024 nodes. Each mobility process simulation lasts for 1,000 time steps and 100 trials were performed. Therefore, each MMP model will also be simulated for 1,000 time steps and 100 trials will be performed. In order to establish a baseline, Section 4.2 begins by considering the aggregated MMP model developed by Yang and applies it to model the simulated mobility process.[1] In Section 4.3, a quantized MMP model with a step size of 5 will be used to model the same simulated mobility process. Larger quantization step sizes will be considered in Section 4.4, namely step sizes of 10, 20, 30, 50, and 100. The different quantization step sizes will be compared in Section 4.5, and the chapter is concluded with a discussion in Section 4.6.

4.2. Modeling Using the Aggregated MMP Model

The mobility process being modeled has 1,024 nodes and the maximum number of links that can exist in a graph of 1,024 nodes is $L_{\max} = 523,776$. Each state of the aggregated MMP model encodes the number of links in the modulated graph and in principle, $L_{\max} + 1 = 523,777$ states are required. In practice, only numbers of links which are actually observed during the simulation need to be modeled. The transition probabilities of the MMP model are derived from the observed transitions in the mobility process, so if the mobility process never reaches a certain number of links L, then it will never be possible for the MMP model to enter the state that models L links, and therefore that state can be omitted from the model. Over each time step of the 100 trials of the mobility process simulations, 360 unique numbers of links were observed, and so in practice the aggregated MMP model has 360 states.

Using the data of all 100 trials of the mobility process, the probability transition matrix and action sets of the aggregated MMP model are constructed according to the procedure described in Section 2.3.4. After constructing the aggregated MMP model, it is used to modulate a graph of 1,024 nodes. The graph is initialized as an empty graph containing no links, and the Markov process is also initialized in the state with 0 links. This is chosen because the mobility process simulations also always start with 0 links, due to the fact that the node positions are initialized in a square lattice where the separation between the rows and columns of the lattice exceeds distance threshold for a link to exist.

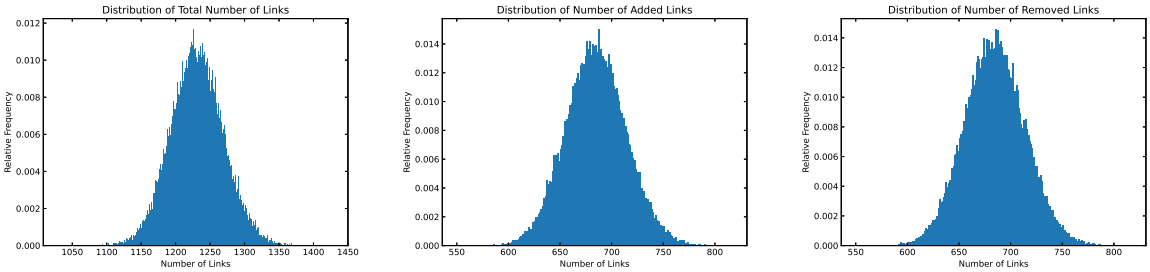
After initializing the MMP model, it is simulated for 1,000 time steps. At each time step, the Markov process transitions to the next state. Suppose that the Markov process is in state i at time t and it transitions to state j at time $t+1$. The action set corresponding to this state transition is $J_{i,j}$. One of the actions $f \in J_{i,j}$ will then be randomly chosen based on the probability distribution of $J_{i,j}$. The action f is applied to modulate the graph; the links to be added are randomly chosen from the set of all links which do not exist at time t, while the links to be removed are randomly chosen from the set of all links that exist at time t. After applying the action, the simulation continues to the next time step.

The MMP model is simulated for 1,000 time steps and 100 trials are performed. The adjacency matrix at each time step is saved, and each simulation produces a series of 1,001 adjacency matrices because there is one additional matrix for the initial graph at time $t = 0$. The number of links in each adjacency matrix is

[1] The aggregated MMP model can also be viewed as a quantized MMP model with a quantization step size of 1.

calculated and the distribution of the number of links has been plotted in Figure 4.1a. Between each pair of adjacency matrices at consecutive time steps, the number of added links and number of removed links is calculated, and the distributions are plotted in Figure 4.1b and Figure 4.1c respectively.[2] The average values are as follows:

- Average number of links at each time step: 1,228.96.
- Average number of links added between two time steps: 684.55.
- Average number of links removed between two time steps: 683.32.



(a) Distribution of the total number of links at each time step.

(b) Distribution of the number of links added between two time steps.

(c) Distribution of the number of links removed between two time steps.

Figure 4.1: Histograms of link statistics for the aggregated MMP model across 100 trials of 1,000 time steps.

The average link metrics of the aggregated MMP model match the mobility process with a very high accuracy: the average number of links differs by approximately 0.01%, while the average number number of links added and removed between two time steps both differ from the mobility process by approximately 0.016%. The distributions of these link metrics also closely match the real mobility process: the standard deviation of the number of links differs by approximately 0.06%, while the standard deviations of the number of links added and removed differ by 0.4% and 0.04% respectively.[3] A plot of the number of links at each time step for one realization of the aggregated MMP model is shown in Figure 4.2.

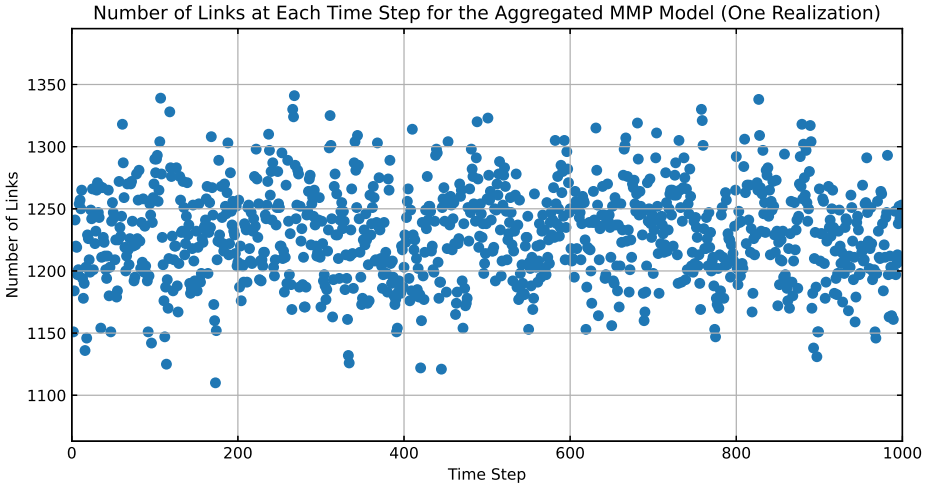


Figure 4.2: Number of links at each time step for one trial of the aggregated MMP model.

[2] Larger versions of the plots of the link distributions of the aggregated MMP model can be found in Appendix A.2.
[3] A table of the mean and standard deviation of the link distributions can be found in Appendix B.

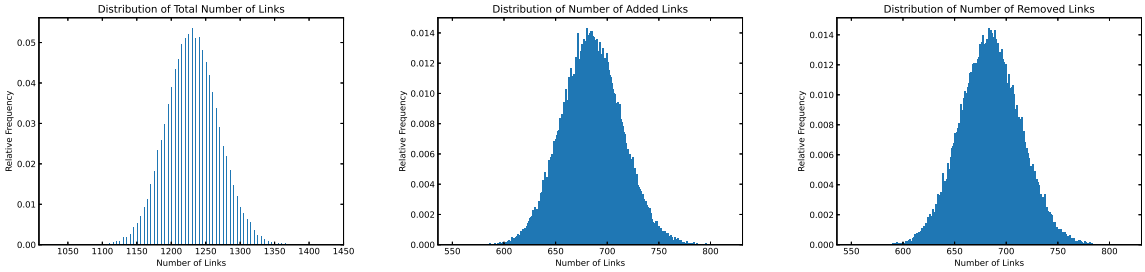
4.3. Modeling Using the Quantized MMP Model with Step Size 5

The quantized MMP model is proposed as a modification of the aggregated MMP model to further reduce the number of states, and a quantization step size of $q = 5$ is investigated first. As described in Section 2.4, there are two additional steps involved in constructing the quantized MMP model, namely link quantization and action remapping. At each time step of the mobility process, the number of links is calculated and quantized according to the procedure described in Section 2.4.1. Between each pair of consecutive time steps, the action is calculated and remapped (if necessary) to ensure that it matches the link delta after quantization $\Delta\hat{L}$, as described in Section 2.4.2. After these two steps have been performed, the probability transition matrix and the action sets are constructed using the exact same procedure used by the aggregated MMP model, as described in Section 2.4.3.

Compared to the aggregated MMP model, the quantized MMP model reduces the number of states by a factor equal to the quantization step size q. The required number of states in the quantized MMP model can be calculated from eq. (2.6); given a graph $N = 1,024$ nodes and using a quantization step size of $q = 5$, the number of states is calculated to be 104,756. However, as was the case with the aggregated MMP model, it is only necessary to model numbers of links which are actually observed in the mobility process. The quantized MMP model for $q = 5$ therefore only consists of 91 states.

The simulation procedure for the quantized MMP model is identical to that of the aggregated MMP model described in the previous section. Each simulation is performed for 1,000 time steps and 100 trials are performed. After performing the simulations, the number of links in each adjacency matrix is calculated and the distribution of the number of links has been plotted in Figure 4.3a. Between each pair of adjacency matrices at consecutive time steps, the number of added links and number of removed links is calculated, and the distributions are plotted in Figure 4.3b and Figure 4.3c respectively.[4] The average values are as follows:

- Average number of links at each time step: 1,228.76.
- Average number of links added between two time steps: 684.81.
- Average number of links removed between two time steps: 683.58.



(a) Distribution of the total number of links at each time step.

(b) Distribution of the number of links added between two time steps.

(c) Distribution of the number of links removed between two time steps.

Figure 4.3: Histograms of link statistics for the quantized MMP model with step size 5 across 100 trials of 1,000 time steps.

The average link metrics of the quantized MMP model also match the mobility process with a very high accuracy: the average number of links differs by approximately 0.026%, while the average number number of links added and removed between two time steps both differ from the mobility process by approximately 0.022%. The error is higher than the aggregated MMP model, which is to be expected due to the loss of information in the quantization step.

As seen in Figure 4.3a, there are gaps in the distribution of the number of links; the distribution now exists only at numbers which are multiples of the quantization step size $q = 5$. The shape still matches the distribution that was observed in the mobility process, and the standard deviation differs from that of the mobility process by about 0.33%.[5]

For the number of links added and the number of links removed between two time steps, there are no gaps in the distributions as seen in Figure 4.3b and Figure 4.3c respectively. This is because the quantized MMP

[4] Larger versions of the plots of the link distributions of the aggregated MMP model can be found in Appendix A.3.
[5] A table of the mean and standard deviation of the link distributions can be found in Appendix B.

model only imposes a restriction on the net effect of each action: for each action, the number of links added and the number of links removed must result in a net change which is a multiple of q, but the actual amount of links added or removed need not be multiples of q. Despite the action remapping, the distributions of number of links added and removed still match the distributions observed in the mobility process as seen in the figures. The standard deviation of the number of links added in the quantized MMP model differs from that of the mobility process by 0.4%, while the standard deviation of the number of links removed differs by 0.004%. Note that comparing the standard deviation can only serve as a rough indicator of whether there is a significant difference between the distributions. A large difference in the standard deviation indicates that the distribution has changed significantly, but a small difference does not necessarily mean the distributions are the same. The result of a 0.004% difference in the standard deviation should therefore not be interpreted as an almost identical distribution; quantization will inevitably cause changes in the distribution. The quantization error will be further discussed in Section 4.5.1.

A plot of the number of links at each time step for one realization of the quantized MMP model is shown in Figure 4.4. The number of links is now restricted to multiples of quantization step size, but this is not very apparent yet due to the scale of the plot and the small step size.

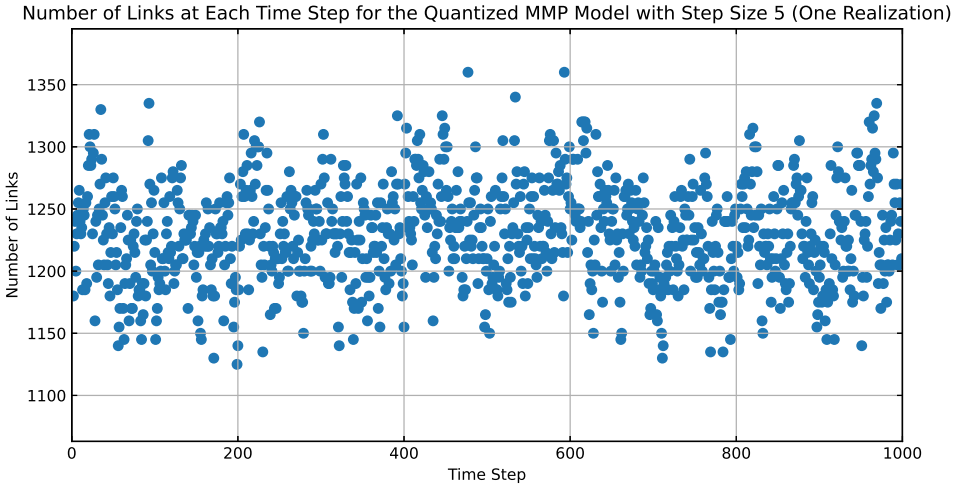


Figure 4.4: Number of links at each time step for one trial of the quantized MMP model with step size 5.

4.4. Modeling with Larger Quantization Step Sizes

The quantized MMP model with a step size of $q = 5$ required only 91 states compared to the 360 states of the unaggregated MMP model. At step size $q = 5$, the quantized MMP model was still able to model the mobility process with very high accuracy and the accuracy was only marginally lower than the aggregated MMP model. Larger quantization step sizes will be now be considered, namely $q = 10$, 20, 30, 50, and 100. After constructing the quantized MMP model for each of these step sizes and performing 100 trials of 1,000 time steps, the same link metrics are calculated. For each of the MMP models, the number of states, relative error in the mean of the link metrics (compared to the mobility process), and relative error of the standard deviation of the link metrics (compared to the mobility process) has been calculated and is summarized in Table 4.1.[6]

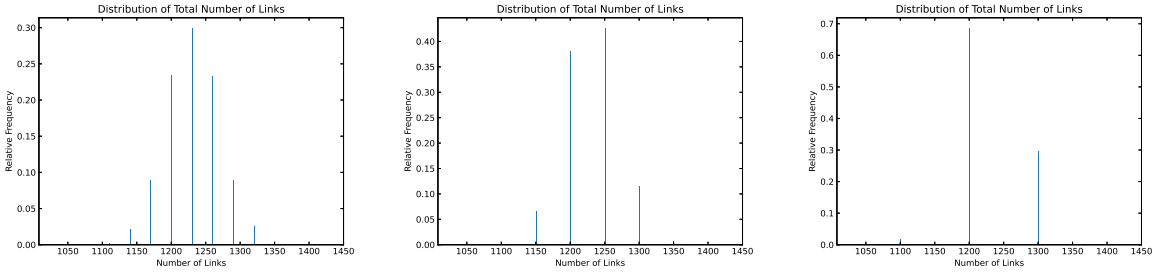
[6]A table of the mean and standard deviation of the link distributions can be found in Appendix B.

Table 4.1: Comparison of link metrics for the different MMP models.

Model	Number of States		Relative Error of Mean			Relative Error of Std. Dev.		
	Theoretical	Actual	Links	Added	Removed	Links	Added	Removed
Aggregated	523,777	360	0.010%	0.016%	0.016%	0.060%	0.389%	0.039%
Quant. 5	104,756	91	0.026%	0.022%	0.022%	0.327%	0.400%	0.004%
Quant. 10	52,379	49	0.014%	0.036%	0.036%	0.302%	0.485%	0.359%
Quant. 20	26,190	26	0.026%	0.022%	0.022%	0.785%	1.002%	0.617%
Quant. 30	17,460	19	0.004%	0.032%	0.033%	1.202%	2.106%	1.548%
Quant. 50	10,477	12	0.016%	0.045%	0.047%	3.215%	5.056%	3.519%
Quant. 100	5,239	7	0.184%	0.044%	0.046%	13.791%	21.285%	14.385%

The mean of the link metrics of the quantized MMP models have a slightly larger error than the aggregated MMP model, but the accuracy is still very high: the largest error is still only 0.184%. The quantization step rounds the number of links which preserves the mean number of links; the fair strategy used for remapping actions is also able to preserve the mean number of links added and removed between two time steps. However, the quantization and action remapping causes the distributions to change, and as the quantization step size increases, the change in the distribution is reflected by the increasing error in the standard deviation of the link metrics.

The effect of quantization on the shape of the link metric distributions becomes more apparent for larger quantization step sizes. In Figure 4.5, the distribution of the number of links for the quantized MMP model of step size 30, 50, and 100 are plotted.[7] The gaps in the distribution become wider as the quantization step size increases and the shape of the original distribution is eventually lost.



(a) Distribution of the total number of links at each time step for quantization step size 30.

(b) Distribution of the total number of links at each time step for quantization step size 50.

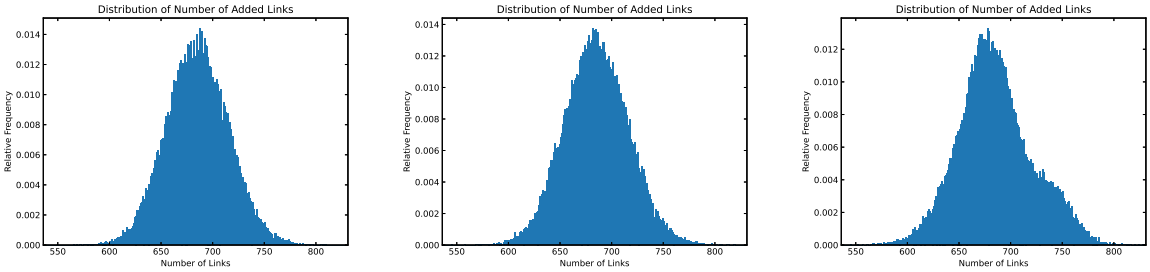
(c) Distribution of the total number of links at each time step for quantization step size 100.

Figure 4.5: Comparison of the distribution of the total number of links at each time step for different quantization step sizes.

In Figure 4.6, the distribution of the number of added links for the quantized MMP model of step size 30, 50, and 100 are plotted. The number of added and removed links is not directly quantized so there are no gaps in the distribution, but the action remapping causes the distributions to change. While the distributions for step size 30 and 50 are still quite close to the original distribution of the mobility process, the extreme step size of 100 shows a pronounced change in the shape of the distribution.

A plot of the number of links at each time step for on realization of the quantized MMP model with step size 30 is shown in Figure 4.7. The quantization steps are now clearly visible and the number of links in the graph jumps between the quantization levels. The number of links at each time step for the quantized MMP model with step size 100 is shown in Figure 4.8, and the number of links is almost always jumping between two levels.

[7] The plots for all of the link distributions can be found in Appendix A.



(a) Distribution of the number of links added between two time steps for quantization step size 30.

(b) Distribution of the number of links added between two time steps for quantization step size 50.

(c) Distribution of the number of links added between two time steps for quantization step size 100.

Figure 4.6: Comparison of the distribution of the number of links added between two time steps for different quantization step sizes.

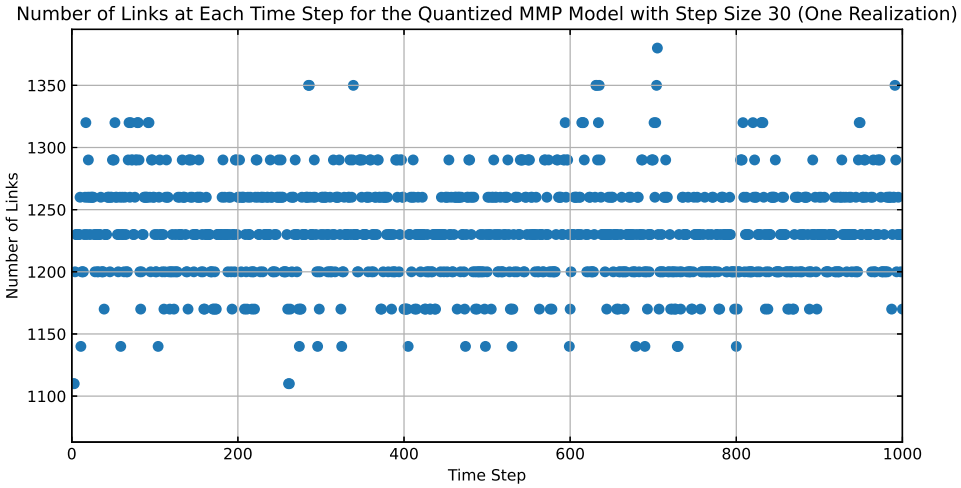


Figure 4.7: Number of links at each time step for one trial of the quantized MMP model with step size 30.

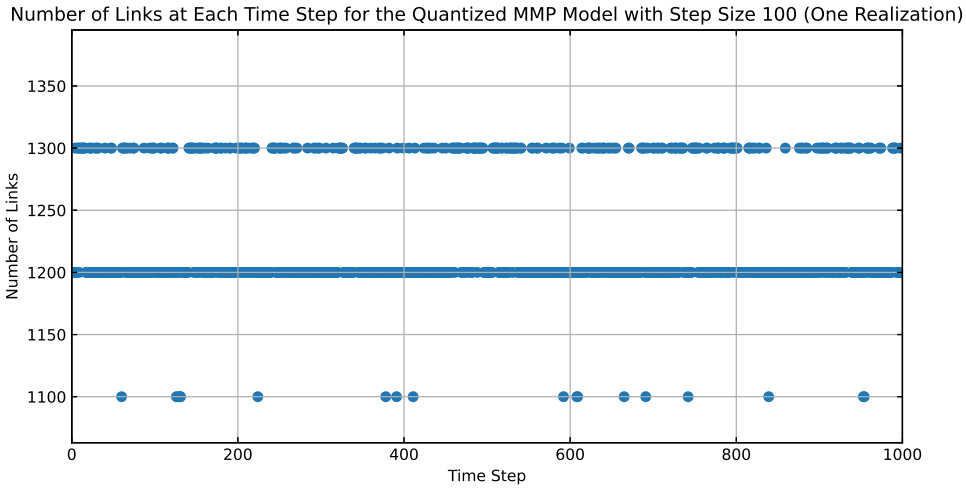


Figure 4.8: Number of links at each time step for one trial of the quantized MMP model with step size 100.

4.5. Further Comparison of the Quantization Step Sizes

4.5.1. Quantization Error

The error incurred when constructing the quantized the MMP model is analyzed in this section. During quantization, the number of links in the mobility process is rounded. At any given time step, the error e_{link} incurred when quantizing the number of links is calculated as

$$e_{\text{link}} = \hat{L} - L, \tag{4.1}$$

where L is the number of links in the graph of the mobility process $\hat{L}$ is the new number of links after quantization. The error sequence for the number of links is denoted as $e_{\text{link}}(k)$ with an arbitrary index k which simply corresponds to the order in which the links were quantized when building the model. Assuming the mobility process trials are processed starting from trial 1 and the time steps are processed in numerical order, then $e_{\text{link}}(0)$ corresponds to the error of time step 0 of trial 1, $e_{\text{link}}(1)$ corresponds to time step 1 of trial 1, up until $e_{\text{link}}(1,000)$ which corresponds to time step 1,000 of trial 1. Assuming the trials are also processed in numerical order, then the next trial is trial 2 and $e_{\text{link}}(1,001)$ will correspond to time step 0 of trial 2. The order in which the quantization is unimportant.

The action remapping will also incur an error in the number of added and the number of removed links. The error sequences of the added and removed number links are denoted as $e_{\text{add}}(k)$ and $e_{\text{rem}}(k)$ respectively. Given an action f and the corresponding remapped action $\hat{f}$, the error in the number of added and removed links are respectively calculated as

$$e_{\text{add}} = \hat{f}_{\text{add}} - f_{\text{add}} \tag{4.2}$$

and

$$e_{\text{rem}} = \hat{f}_{\text{rem}} - f_{\text{rem}}, \tag{4.3}$$

where f_{add} denotes the number of links added by action f and f_{rem} denotes the number of links removed.

The error sequence $e(k)$ can be interpreted as a random signal. The mean of the error sequence is $\mathbb{E}[e(k)]$ and the power (i.e. variance) in the error sequence is given by

$$\sigma^2 = \mathbb{E}[e^2(k)]. \tag{4.4}$$

The RMS (root mean square) error σ is equal to the square root of the power. The error sequence of the number of links, the number of added links, and the number of removed links for each of the quantized MMP models has been calculated. The mean and RMS of each of the error sequences has been summarized in Table 4.2.

Table 4.2: The mean and RMS of each of the error sequences for different quantization step sizes.

Step Size	**Mean Error**			**RMS Error**		
	Links	Added	Removed	Links	Added	Removed
5	-0.0001	0.2414	0.2415	1.411	1.059	1.058
10	0.0030	0.2512	0.2515	2.914	2.091	2.088
20	-0.0122	0.2514	0.2511	5.775	4.116	4.111
30	-0.0065	0.2511	0.2516	8.669	6.167	6.162
50	-0.0815	0.2519	0.2504	14.426	10.203	10.197
100	-2.3243	0.2536	0.2601	29.073	20.170	20.145

It is known from literature that, assuming a uniformly distributed input distribution, quantizing with rounding results in an error sequence with a mean of 0 and an RMS value proportional to the quantization step size q. The proportionality is given by

$$\sigma = \frac{q}{\sqrt{12}}, \tag{4.5}$$

as shown in [9, p. 462]. As seen in Table 4.2, the mean error of the number of links is indeed very close to 0 for all of the quantization step sizes, with the exception for $q = 100$ where a somewhat larger error is observed.

This is due to the fact that quantization step size is very large and the assumption of a uniformly distributed input over the quantization levels is no longer valid.

The mean error of the number of added and removed links are very close to each other for all of the quantization step sizes, but there is a slight positive bias. This is a consequence of the chosen action remapping strategy: the action remapping strategy tries to equally offset the number of added and removed links, but if this is not possible, it either increases the number of added or removed links. Because it always increases the number of links in this case, this results in a slight positive bias. However, the link remapping strategy is successful in fairly spreading the quantization error over the added and removed number of links, evidenced by the fact that the mean error of the added and removed links is almost equal, and the RMS error is also almost equal.

The RMS error in the number of links, added links, and removed links has been plotted against the quantization step size in Figure 4.9. The RMS error increases linearly with the quantization step size. From eq. (4.5), the RMS error of the number of links is expected to be proportional to the quantization step size by $1/\sqrt{12} \approx 0.289$. Linearly fitting the RMS error of the number of links results in a gradient of 0.291 and an R^2 value of 0.99998, matching the theoretical result almost exactly.

Figure 4.9: RMS for different quantization step sizes. Note that the RMS error of the added and removed links conicide almost exactly and the green line is on top of the orange line.

The RMS error of the number of added and removed links are almost exactly equal, and the lines on the plot are on top of each other. Although the error also increases linearly with the quantization step size, the proportionality with $1/\sqrt{12}$ is not observed. This is to be expected because the added and removed links are not directly quantized with the step size q. The error incurred by the action remapping is related to the quantization step size, and in Appendix C, it has been derived that the relationship between the RMS error and the quantization step size for the added and removed links is given by

$$\sigma = \frac{q}{\sqrt{24}}. \tag{4.6}$$

Therefore, the RMS error of the added and removed links is expected to be proportional to the quantization step size by $1/\sqrt{24} \approx 0.204$. Fitting the added links with a linear trend line results in a gradient of 0.201 and an R^2 value of 0.99997, which is very close to the expected result. Fitting the removed links with a linear trend line also results in a gradient of 0.201 and an R^2 value of 0.99997. Both of these values are very close to the theoretical result.

4.5.2. K-Step Link Retention Probability

One important aspect of the mobility process that cannot be captured by the MMP model is the *selection* of which links to remove. In the simulated mobility process, the adding and removing of links is based on the movement of the nodes. However, the spatial aspect of the mobility process is lost in the MMP model, and therefore, links can only be added and removed randomly. To investigate how randomly selecting links to

remove affects the modulated graph, Yang defines a metric called the *K-step link retention probability*. Yang defines the K-step link retention probability as "the probability that a link which exists at time t will still exist at each subsequent time step up to and including $t + K$" [1, p. 31].

The link retention probability up to $K = 20$ has been calculated for the mobility process and each of the MMP models and is plotted in Figure 4.10. The probability is plotted on a logarithmic scale. The link retention probability of all of the MMP models decrease exponentially and up to 14 time steps, the values are almost exactly the same. After 14 steps, some variations are observed in the probabilities but it appears to be simply due to stochasticity in the models; the link retention probability after 14 steps is already less than 10^{-5} and there are no trends in the variations.

The link retention probability of the MMP models matches the mobility process up to around 4 time steps, but afterwards they diverge and the mobility process has a consistently higher link retention probability. This is because in the mobility process, there is the possibility for nodes to move in a similar direction. If two nodes are connected and happen to be facing a similar direction, there is a larger probability that they will stay connected at the next time step. In the MMP model, the spatial aspect is lost and every link is removed with equal probability.

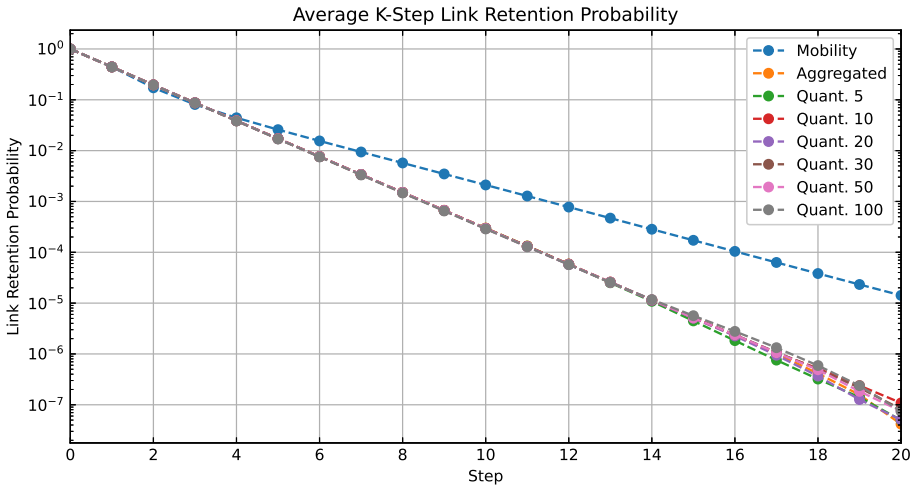


Figure 4.10: K-step link retention probability for the different models.

4.5.3. Unique Links Observed

Due to the fact that the spatial aspect of the mobility process is lost, the selection of which links to add can also only be performed randomly. Yang considers the cumulative number of unique links observed as a metric to compare how this affects the modulated graph [1, p. 33]. The cumulative number of unique links observed at each time step has been averaged over all of the realizations of the mobility process and MMP models and has been plotted in Figure 4.11. The results for the MMP models lie on top of each other and are indistinguishable unless zooming in very far, as shown in the inset in Figure 4.11. The largest difference between the maximum unique links and minimum unique links across the MMP models is about 222 which occurs at time step 612; the inset shows time steps 610-614 and is enlarged 80 times.

The unique links observed in the MMP model is consistently higher than in the mobility process, which is the opposite of the behavior observed in Yang's research [1, p. 34]. Since the simulation region is now larger, nodes have more freedom in their movement and it is harder for a previously unobserved link to be created. The MMP does not capture this spatial aspect and chooses a random link to create, so the number of unique links observed grows faster.

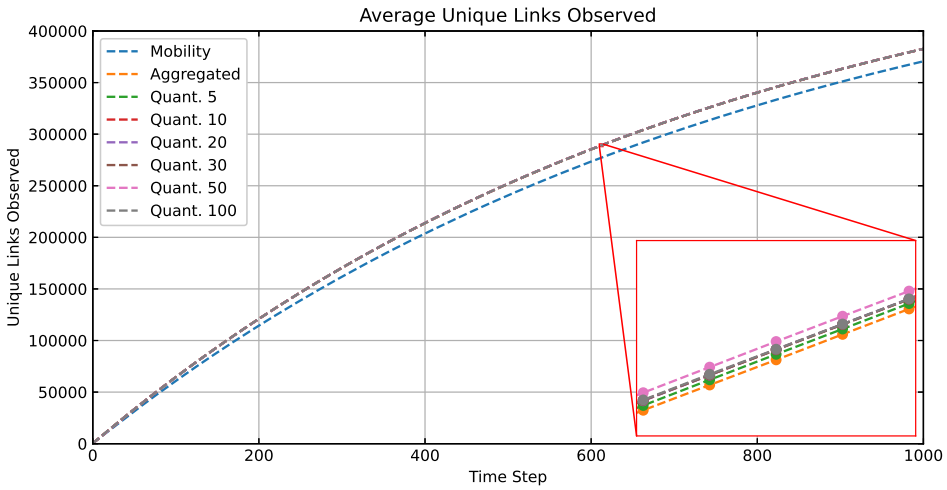


Figure 4.11: Average cumulative number of unique links observed at each time step. The inset shows time steps 610-614 and is enlarged 80 times.

4.6. Discussion

When the number of nodes is increased to $N = 1,024$, the aggregated MMP model is still able to accurately capture the average number of links in the graph as well as the average number of links added and removed between two time steps with a very high accuracy. In theory, the aggregated MMP model for 1,024 nodes requires 523,777 states, but in practice, only 360 states were used.

The quantized MMP models are also able to capture the average number of links and the average number of links added and removed with high accuracy, but the accuracy is slightly lower than the mobility process. The advantage of the quantized MMP models is the reduction in the number of states: for a quantization step size of 5, the number of states was reduced to 91. As a result of the quantization, the distribution of links in the graph changes because the number of links is now forced to a multiple of the quantization step size q. However, the mean number of links in the graph remains the same.

The mean value of the number of added and removed links in the quantized MMP model matches the mobility process very closely, but there is a slight positive bias. The positive bias is due to the fact that the action remapping strategy always increases the number of added or removed links when equally offsetting the number of added and removed links is not possible. However, the action remapping strategy is successful in distributing the quantization error in the number of links equally over the added and removed links.

When comparing the K-step link retention probability, the behavior of all of the MMP models is identical and the retention probability decreases exponentially. The retention probability of the MMP models matches the mobility process up to about 4 steps, after which they diverge and the mobility process has a higher retention probability. When comparing the number of unique links observed, all of the MMP models again exhibit the same behavior. Compared to the mobility process, however, the unique links in the MMP model tends to increase faster. In the next chapter, SIR processes running on the different MMP models will be investigated to see whether the different quantization levels result in any differences in the dynamics of the SIR process.

5

SIR Simulations

5.1. Introduction

Yang [1] briefly investigated the spreading behavior of an SIR process running on top of the mobility process and compared it with the same process running on top of an aggregated MMP model. Yang used a graph of 25 nodes and found that the SIR process tended to spread faster on the aggregated MMP model and the peak fraction of infectious nodes tended to be higher compared to the mobility process [1, Sec. 6.3]. In this chapter, the same SIR process will be considered, but the graph will now consist of 1,024 nodes. The objective is to investigate how the SIR process spreads with a larger number of nodes, and also to investigate any differences in behavior between the different MMP models.

5.2. Simulation Procedure

To allow for a fair comparison, the SIR simulation will follow the same procedure used by Yang [1, Sec. 6.3.3]. In the SIR process, nodes exist in one of three states: susceptible, infectious, or recovered. At each time step, susceptible nodes can be infected by an infectious neighbor with probability β. Since temporal graphs are being considered, a node's neighbors will change over time. Each infectious neighbor independently tries to infect the susceptible node with probability β, and the node becomes infected if at least one of the probabilities succeed. Infectious nodes can become cured by the curing probability δ, after which the enter the recovered state. Nodes in the recovered state can no longer infect other nodes nor become infected again. This models an infectious disease characterized by permanent immunity following the recovery of an infected individual, and is one of the most well-known compartmental models.

SIR simulations will be performed for the quantized MMP models for each of the quantization step sizes $q \in \{5, 10, 20, 30, 50, 100\}$, as well as the aggregated MMP model and the mobility process. The infection probabilities are chosen to be $\beta \in \{0.1, 0.3, 0.5, 0.7\}$ and the curing probabilities are chosen to be $\delta \in \{0.1, 0.3, 0.5, 0.7\}$. For every model, each combination of β and δ will be simulated.

For each MMP model (and also for the mobility process), 100 realizations of the temporal graph are available and 10 SIR trials will be performed on each of the realizations for a total of 1,000 trials. The simulation procedure for each trial is as follows. At time $t = 0$, one node is randomly chosen to be infectious while the remaining 1,023 nodes are susceptible. At each time step t, susceptible nodes can be infected by each of their infectious neighbors with probability β. The neighbors are determined by the links in the adjacency matrix of the current realization of the MMP model (or mobility process) at time t. If at least one of the probabilities succeed, the node becomes infectious at time $t+1$. Each of the infectious nodes can recover with probability δ. If the probability succeeds, the node becomes recovered at time $t+1$. After evaluating all of the infection and recovery probabilities, the simulation continues to the next time step. The simulation ends either when there are no more infectious nodes remaining or when the maximum time step $t = 1,000$ is reached (although the latter condition is never observed during the simulations).

5.3. Results

5.3.1. Metrics

The SIR simulations are performed for the 8 different models: the quantized MMP model for steps $q \in \{5, 10, 20, 30, 50, 100\}$, the aggregated MMP model, and the mobility process. There are 16 possible combinations of the parameters β and δ resulting in a total of 128 SIR simulations. To compare the results of the simulations, several metrics are considered. The first set of metrics consider the duration of the simulation, which is the number of time steps until there are no more infectious nodes and the simulation ends. The average, standard deviation, and the maximum simulation duration are calculated across the 1,000 trials of each SIR simulation. The second set of metrics consider the fraction of infectious nodes averaged across the 1,000 trials. The peak fraction of infectious nodes, the time step at which the peak occurs, and the cumulative fraction of infected nodes are calculated.[1] These values are summarized in the table in Appendix D.

One of the most interesting observation is that there is no meaningful difference between the behavior of all of the MMP models: the aggregated MMP model and the quantized MMP model for all of the different quantization step sizes show the exact same behavior. While there are some small variations between the metrics of the different MMP models, the variations appear to simply be due to the stochasticity of the simulation and there are no trends observed. The averaged SIR curves of the quantized MMP model and the aggregated MMP model for $\beta = 0.1$ and $\delta = 0.1$ are plotted in Figure 5.1. The aggregated MMP model has 360 states while the quantized MMP model with step size 100 has 7 states (and in practice, the quantized MMP model with step size 100 is mostly staying in only 2 of the states). Despite this, it shows exactly the same SIR dynamics. In Appendix E.1, the SIR curves of the aggregated MMP model and the quantized MMP model of step size 100 are compared for all of the values of β and δ, and the same dynamics are observed.

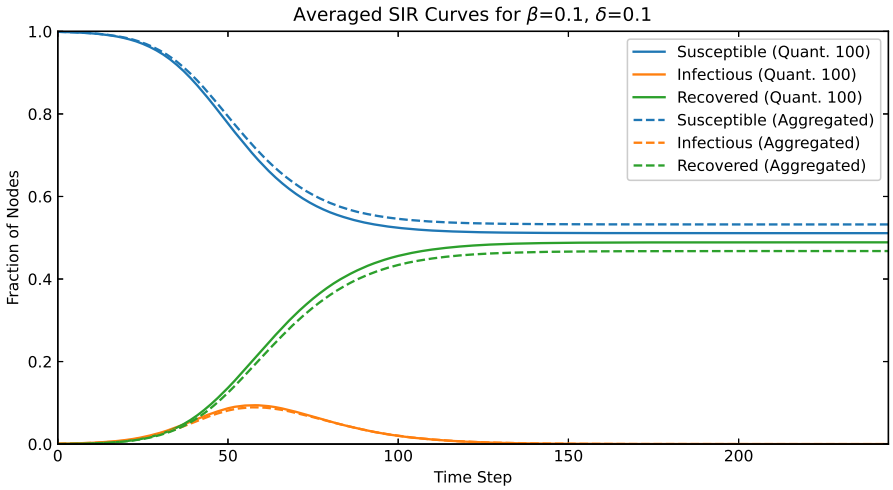


Figure 5.1: SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.1$ and $\delta = 0.1$.

Yang observed that for the SIR process running on the MMP model, the peak in the fraction of infectious nodes tends to occur earlier than in the mobility process and the peak value tends to be higher. The same behavior is observed in these simulations. The following subsections compare the results of the MMP model with the mobility process. Since all of the MMP models exhibit the same behavior, only the results of the quantized MMP model for step size 100 will be shown.

5.3.2. SIR Results for $\beta = 0.1$

The SIR curves of the MMP model and the mobility process for $\delta = 0.1$ are plotted in Figure 5.2. The fraction of infectious nodes peaks about 21% earlier in the MMP model at time step 58, compared to time step 73 for the mobility process. The fraction of infectious nodes at the peak in the MMP model is about 63% higher than the mobility process. The cumulative fraction of infected nodes in the MMP model is about 11% higher at 0.489 compared to 0.439 for the mobility process (this can be read from the graph by comparing the final value of the recovered curves).

[1] The cumulative fraction of infectious nodes is also equal to the final value of the fraction of recovered nodes.

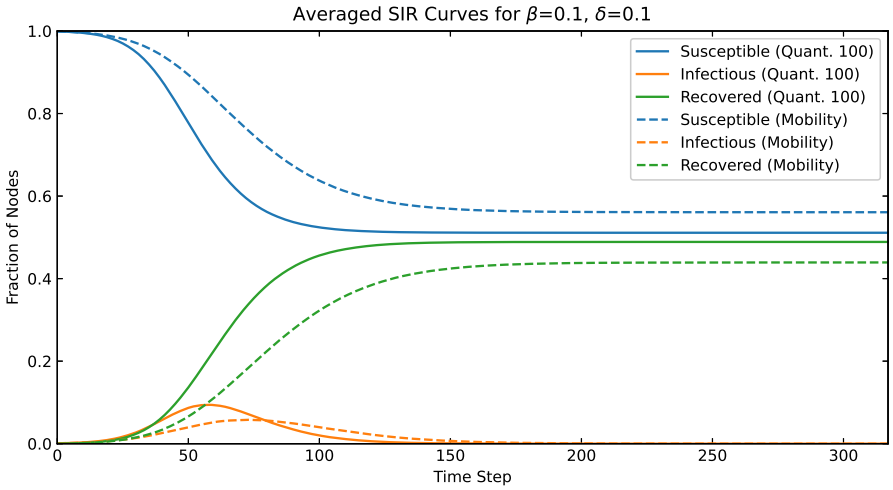


Figure 5.2: SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.1$ and $\delta = 0.1$.

For $\delta = 0.3, 0.5, 0.7$, the SIR process dies out immediately. The peak fraction of infectious nodes is observed at time $t = 0$ and the value is equal to $1/1024$, because the graph is initialized with 1 infectious node. The plots can be found in Appendix E.2.

5.3.3. SIR Results for $\beta = 0.3$

The SIR results for each value of δ has been plotted in Figure 5.3.

For $\delta = 0.1$ (Figure 5.3a), the MMP model again peaks earlier and higher than the mobility process. The difference in the peaks of the infectious curve is now much more significant: the peak occurs about 55% earlier for the MMP model, and the peak value in the is 2.78 times higher than the mobility process. The cumulative fraction of infectious nodes is quite close, however, with the MMP model being only 3% higher than the mobility process.

For $\delta = 0.3$ (Figure 5.3b), the infectious curve for the mobility process is much flatter and the cumulative fraction of infectious nodes is only 0.166. For the MMP model, a small peak is still observed and the cumulative fraction of infectious nodes is about 0.517, a little over 3 times higher.

For $\delta = 0.5$ (Figure 5.3c), the SIR process barely spreads in the mobility process with the cumulative fraction of infected nodes being 0.009. The infectious curve for the MMP model is also very flat, but the cumulative fraction of infected nodes is considerably higher at 0.129. For $\delta = 0.7$ (Figure 5.3d), the SIR process dies out immediately for both the MMP model and the mobility process.

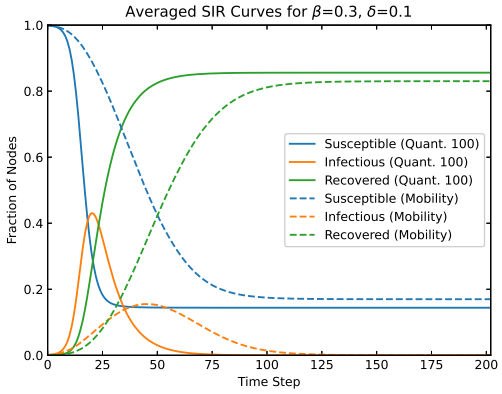
5.3.4. SIR Results for $\beta = 0.5$

The SIR plots of the results of $\beta = 0.5$ do not show any new behavior so they have been omitted from this section; they can be found in Appendix E.2. For $\delta = 0.1$, the difference in the peak fraction of infectious nodes again becomes larger. The peak of the MMP occurs 66% earlier and is about 3.16 times higher than the mobility process. The cumulative fraction of infectious nodes remains quite close with the MMP being only 2.5% higher than the mobility process.

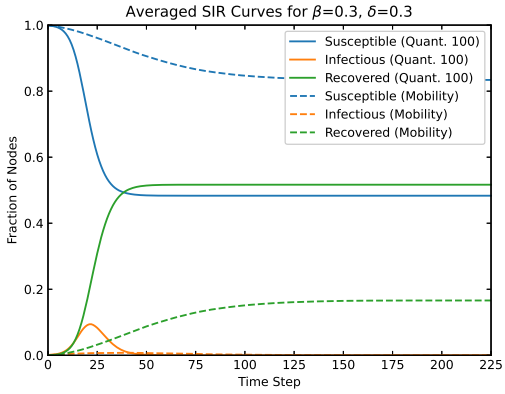
For $\delta = 0.3, 0.5, 0.7$, the infectious curve of the mobility process is very flat, while in the MMP model a peak is still observed. The cumulative fraction of infectious individuals in the MMP model is much higher than in the mobility and the difference becomes greater for the larger values of δ.

5.3.5. SIR Results for $\beta = 0.7$

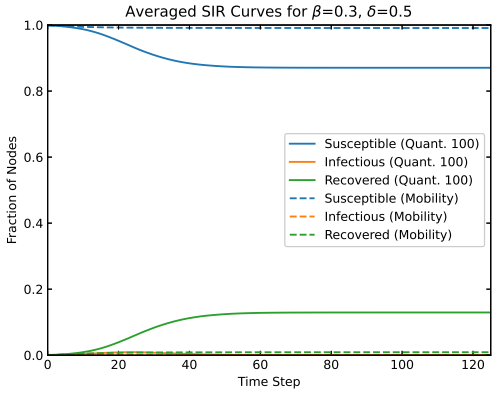
The SIR plots of the results of $\beta = 0.5$ do not show any new behavior so they have been omitted from this section; they can be found in Appendix E.2. For $\delta = 0.1$, the difference in the peak fraction of infectious nodes again becomes larger. The peak of the MMP occurs 72% earlier and is about 3.38 times higher than the mobility process. The cumulative fraction of infectious nodes remains quite close with the MMP being only 1.8% higher than the mobility process.



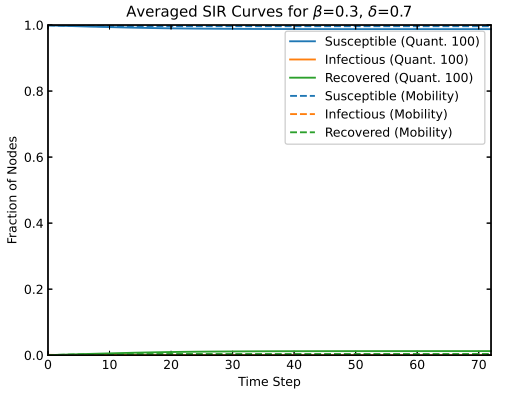
(a) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.3$.



(c) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.5$.



(d) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.7$.

Figure 5.3: SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and various values of δ.

For $\delta = 0.3, 0.5, 0.7$, the infectious curve of the mobility process is very flat, while in the MMP model a peak is still observed. The cumulative fraction of infectious individuals in the MMP model is much higher than in the mobility and the difference becomes greater for the larger values of δ.

5.4. Discussion

One of the most interesting results of the SIR simulations is that the SIR dynamics on all of the MMP models were the same. In the aggregated MMP model, all possible numbers of links are modeled, and the number of links in the modulated graph can take any number of links that was observed in the mobility process used to construct the model. In the quantized MMP, the number of links is restricted to a multiple of the quantization step size q, and therefore the number of links in the modulated graph will jump between values which are multiples of q. Considering the most extreme example of $q = 100$, the number of links in the quantized MMP model is almost always jumping between 1,200 and 1,300 links while rarely jumping to 1,100. This behavior is completely different from the aggregated MMP model, but despite this, the SIR spreading process for every value of β and δ showed the same dynamics and the calculated metrics were very close for both models.

When comparing the MMP models with the mobility process, it is considerably easier for the infection to spread on the MMP model and it spreads much faster. When considering larger values of the curing probability δ, the infectious curve for the mobility process is very flat and relatively few nodes are infected over course of the simulation. But for the MMP models, there is still a noticeable peak in the infectious curve and significantly more nodes are infected.

When considering the case where the curing probability is fixed at $\delta = 0.1$, the MMP model tends to spread the infection faster and have a higher peak fraction of infectious nodes than the mobility process. As the

infection probability increases, the peak becomes much higher than the mobility process. Interestingly, the cumulative fraction of nodes that are infected over the course of the simulation is quite similar for both the MMP model and the mobility process in this case.

The faster spreading of the SIR process on the MMP model is most likely due to the fact that the MMP model lacks the spatial dimension of the mobility process and adds links at random. Therefore, it is possible to create links between many different parts of the graphs, giving the infection many more opportunities to spread. However, in the mobility process, links are created when nodes come into contact with each other and nodes can only move a fixed distance each time step. It is therefore much harder for the infection to spread to nodes that are further away, which bottlenecks the spreading process.

6

Conclusion and Future Directions

6.1. Conclusion

In this research, the several MMP models have been used to model a simulated mobility process of 1,024 nodes. The results of the investigation show that the aggregated MMP model is able to capture the average number of links at each time step as well as average number of links added and removed between two time steps with very high accuracy. In principle, for a graph of 1,024 nodes, the aggregated MMP model requires 523,777 states to model each possible number of links in the graph. However, in practice, only numbers of links which are observed in the mobility process need to be modeled, and the aggregated MMP model only used 360 states.

The quantized MMP model is proposed as a modification of the aggregated MMP model that reduces the number of states by quantizing the number of links in the mobility process. The quantization is performed by rounding the number of links to a multiple of the quantization step size q, reducing the number of required states by a factor of q. Quantization with rounding results in a quantization error with a mean of 0, and all of the quantized MMP models were still able to model the average number of links in the graph with high accuracy. The quantization does, however, change the distribution of the number of links in the graph, because the number of links can now only take values which are multiples of q. The number of links in the modulated graph will also only be able to take values which are multiples of q.

Quantizing the number of links in the mobility process requires the original actions to be remapped so that the number of links in the graph stays at the quantization levels. The proposed action remapping strategy attempts to offset the number of added links and the number of removed links equally wherever to distribute the quantization error fairly between the number of added and removed links. The link remapping strategy is successful in distributing the error fairly, evidenced by the fact that the RMS error of the added and removed links is the same. However, due to the fact that the link remapping strategy always opts to either increase the number of added links or increase the number of removed links when a equal distribution is not possible, this results in a slight positive bias in the mean error of the added and removed links. As a result, the average number of links added and removed between two time steps in the quantized MMP model is slightly higher than in the mobility process. Nevertheless, the quantized MMP model is still able to capture the average number of links added and removed with very high accuracy and the relative error for all of the quantized MMP models was less than 0.05%.

A remarkable result from the SIR simulations was that all of the MMP models showed the same behavior. The quantized MMP models behaved exactly the same as the aggregated MMP model and no significant differences were observed for larger quantization step sizes. This is very interesting because the number of links in the quantized MMP model of step size 100 behaves completely differently from the aggregated MMP model, but the SIR dynamics were exactly the same. The MMP models in their current form, however, are not suitable for modeling the spread of a SIR process. Compared to the mobility process, the SIR process spreads much faster on the MMP models. The MMP model lacks the spatial dimension of the mobility process and adds links randomly, which enables many links between many different parts of the graph to be created. This creates many more opportunities for the infection to spread compared to the mobility process. This result shows that in the context of an SIR simulation, matching the average number of added and removed links is not sufficient, and the choice of which links to add or remove has a large impact on the dynamics. If the MMP

model is to be used for SIR simulations, alternative strategies for selecting which links to add and remove need to be investigated.

6.2. Future Directions

The Markov-modulated process model for modeling mobility processes is a novel idea and there are many different approaches to developing and applying this model that have not yet been researched. The MMP models considered in this thesis all modulate the graph by jointly adding and removing a number of links at each time step. In this section, several avenues for future research are suggested.

Link Selection

The MMP model is able to accurately model the average number of links in the graph as well as the average number of links added and removed between two time steps of the mobility process. However, the selection of which links to add and remove is done completely randomly, which is not representative of how links behave in a real mobility process. As an improvement, the MMP model could be extended by having a secondary process that selects the links to be added and removed in a more realistic way.

Of course such an extension would increase the complexity of the model, whereas one of the motivations for using the MMP model is to have a simplified description of a complex phenomenon. It may therefore also be interesting to investigate whether there are applications for which being able to model the average number of links added and removed is already a sufficient description of the mobility process.

Alternative Quantization Strategies

The quantized MMP model model researched in this thesis uses a fixed quantization step size of q and rounds the number of links to a multiple of q. As q increases, the distribution of the number of links in the graph in the quantized MMP model starts losing the shape of the distribution of the number of links in the original mobility process. At the largest step size $q = 100$ considered in this thesis, the distribution of the number of links in the graph is almost entirely concentrated at 1,200 and 1,300 links, and the modulated graph jumps between these two quantization levels. A simple first improvement would would be to offset all of the quantization levels so that there is a quantization level which coincides with the mean number of links in the graph, making the distribution symmetric. A varying quantization step size could also be considered, so that there are more quantization levels near the center of the distribution of the number of links where the probability is higher.

The link remapping strategy discussed in Section 2.4.2 could also be modified to remove the positive bias. When the number of added and removed links cannot be offset equally, the current strategy favors increasing the number of added links when $\Delta\hat{L} > \Delta L$ and favors increasing the number of removed links when $\Delta\hat{L} < \Delta L$. Although this fairly spreads the error between the number of added and removed links, in both cases it increases the remapping increases the number of links to add/remove, result in a positive bias. This could be reduced by *increasing* the number of links to add in case $\Delta\hat{L} > \Delta L$ and the current number of links being added is even, while *decreasing* the number of links to remove when it is odd. Similarly, when $\Delta\hat{L} < \Delta L$, the number of links being removed should be *increased* if the current number of links being removed is even, and *decreased* in case it is odd.

Scalability of the MMP Model in Practice

In theory, the aggregated MMP model requires $L_{\max} + 1$ states to model a graph of N nodes, where $L_{\max} = N(N-1)/2$ is the maximum number of links that can exist in the graph; for a graph of 1,024 nodes, 523,777 states would be required. However, in practice, the number of states needed is much fewer, due to the fact that many numbers of links are not realistic for a mobility process. To give an extreme example, consider the state that models 523,776 links, i.e. the complete graph for 1,024 nodes. To enter this state, all 1,024 nodes would need to be within 1.5 units of one another, and the probability of this occurring in the simulated mobility process is negligibly small. If a real-world mobility process is considered, this state is actually physically impossible because it is not possible to have 1,024 people all within a very short distance of each other.

In practice, the MMP model only needs to model numbers of links which are observed during the mobility process, and there were 360 unique numbers of links observed in the simulated mobility process of 1,024 nodes. Hence, only 360 states were used in the aggregated MMP model, which is less than 0.07% of the upper bound of 523,777 states. This is a very extreme difference, and it is therefore interesting to investigate how many states are needed when applying the MMP model in practical situations.

Bibliography

[1] L. Yang, "Developing a Markov-modulated process model for mobility processes," Master's thesis, Delft University of Technology, 2021.

[2] B. Prasse, M. A. Achterberg, L. Ma, and P. Van Mieghem, "Network-inference-based prediction of the COVID-19 epidemic outbreak in the Chinese province Hubei," *Applied Network Science*, vol. 5, no. 1, Jul 2020.

[3] M. Meester, I. Görkey, O. Braakman, T. Rood, and N. Bauman, "COVID-19 prediction and mitigation," Bachelor's thesis, Delft University of Technology, 2020.

[4] T. MV, "How to setup Angular on Ubuntu," Feb 2020. [Online]. Available: https://mvthanoshan.medium.com/how-to-setup-angular-on-ubuntu-14633ee93a57

[5] Esri Nederland, "Layer: Coronavirus_RIVM_vlakken_historie (ID:0)." [Online]. Available: https://services.arcgis.com/nSZVuSZjHpEZZbRo/ArcGIS/rest/services/Coronavirus_RIVM_vlakken_historie/FeatureServer/0

[6] C. Zhang, L. Qian, and J. Hu, "COVID-19 pandemic with human mobility across countries," *Journal of the Operations Research Society of China*, vol. 9, pp. 1–16, Aug 2020.

[7] M. Kraemer, C.-H. Yang, B. Gutierrez, C.-H. Wu, B. Klein, D. Pigott, L. Plessis, N. Faria, R. Li, W. Hanage, J. Brownstein, M. Layan, A. Vespignani, H. Tian, C. Dye, S. Cauchemez, O. Pybus, and S. Scarpino, "The effect of human mobility and control measures on the COVID-19 epidemic in China," *medRxiv*, Mar 2020.

[8] Z. Zheng, Z. Xie, Y. Qin, K. Wang, Y. Yu, and P. Fu, "Exploring the influence of human mobility factors and spread prediction on early COVID-19 in the USA," *BMC Public Health*, vol. 21, Mar 2021.

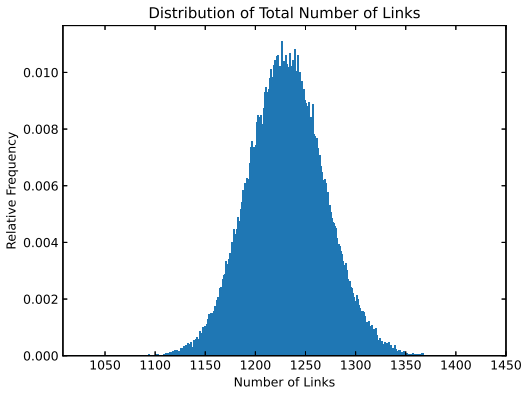
[9] A. Ambardar, *Analog and digital signal processing*, 2nd ed. Brooks/Cole, 1999.

Appendices

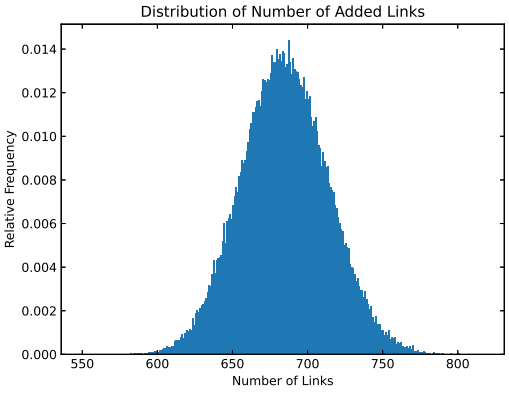
A

Link Distribution Plots

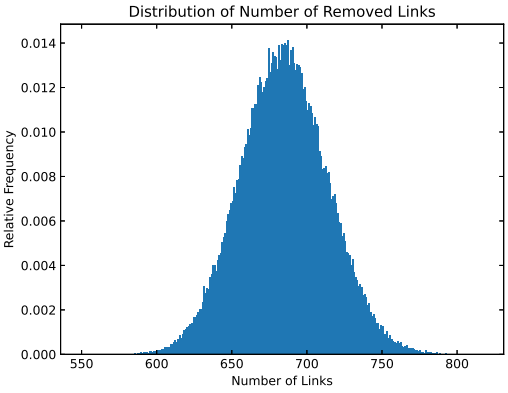
A.1. Simulated Mobility Process



(a) Distribution of the total number of links at each time step.



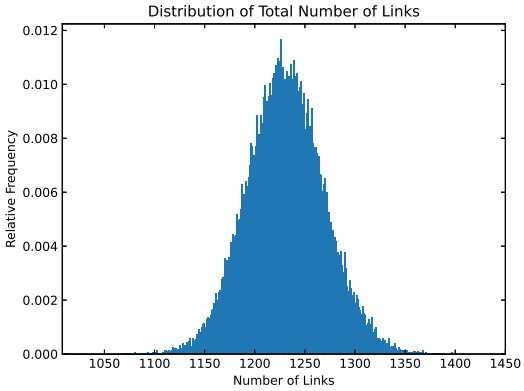
(b) Distribution of the number of links added between two time steps.



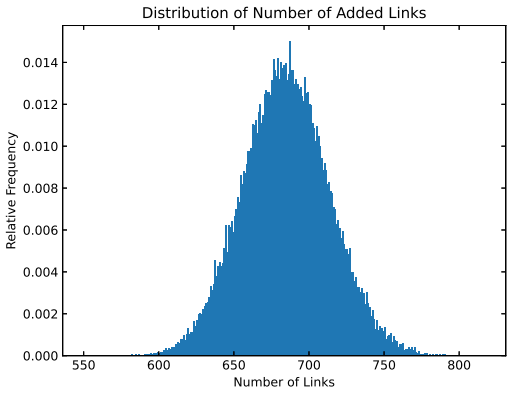
(c) Distribution of the number of links removed between two time steps.

Figure A.1: Histograms of link statistics for the simulated mobility process across 100 trials of 1,000 time steps.

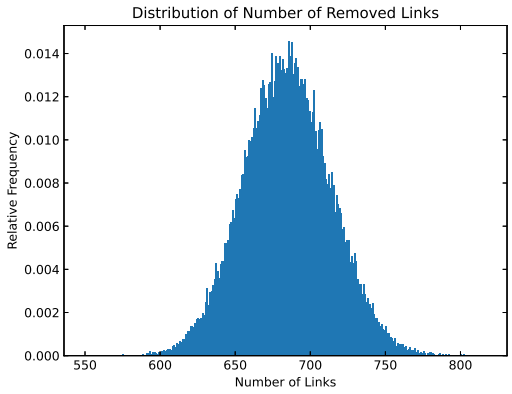
A.2. Aggregated MMP Model



(a) Distribution of the total number of links at each time step.



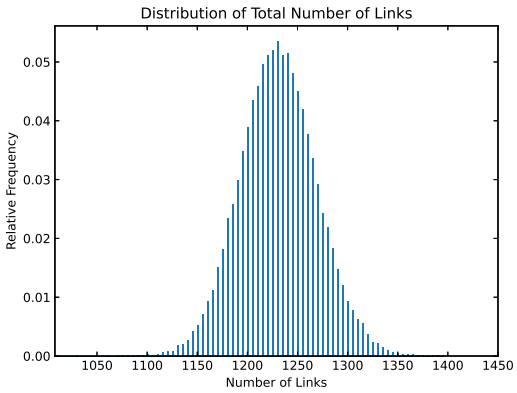
(b) Distribution of the number of links added between two time steps.



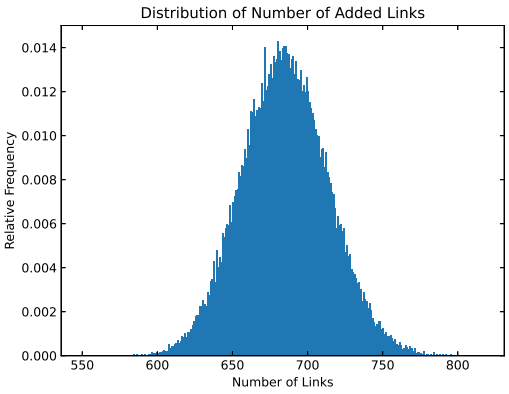
(c) Distribution of the number of links removed between two time steps.

Figure A.2: Histograms of link statistics for the aggregated MMP model across 100 trials of 1,000 time steps.

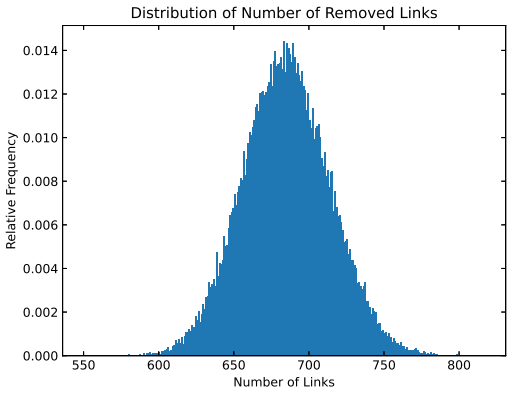
A.3. Quantized MMP Model, Step Size 5



(a) Distribution of the total number of links at each time step.



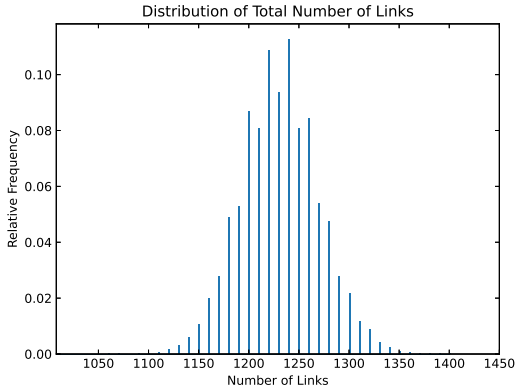
(b) Distribution of the number of links added between two time steps.



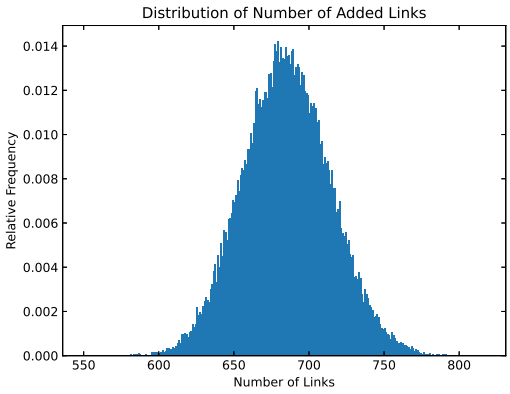
(c) Distribution of the number of links removed between two time steps.

Figure A.3: Histograms of link statistics for the quantized MMP model with step size 5 across 100 trials of 1,000 time steps.

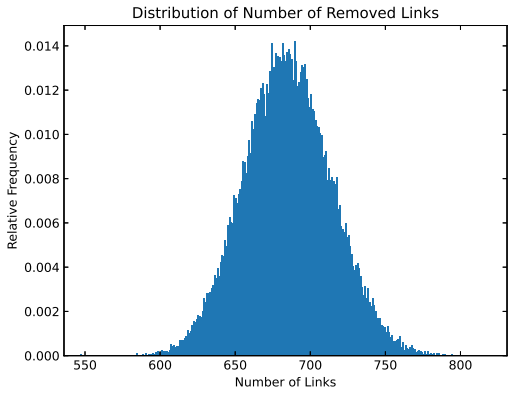
A.4. Quantized MMP Model, Step Size 10



(a) Distribution of the total number of links at each time step.



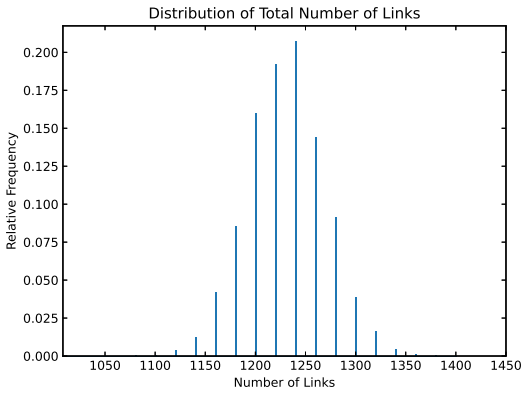
(b) Distribution of the number of links added between two time steps.



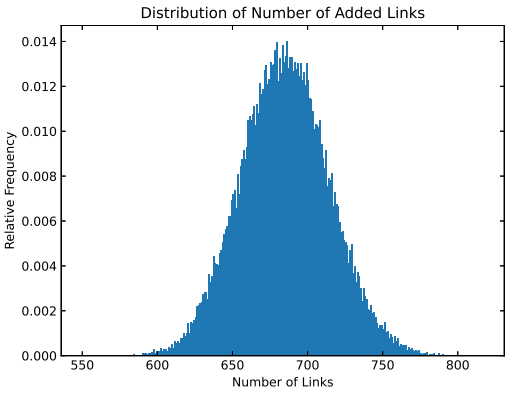
(c) Distribution of the number of links removed between two time steps.

Figure A.4: Histograms of link statistics for the quantized MMP model with step size 10 across 100 trials of 1,000 time steps.

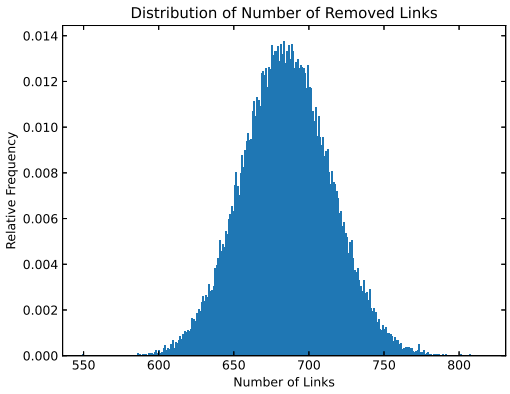
A.5. Quantized MMP Model, Step Size 20



(a) Distribution of the total number of links at each time step.



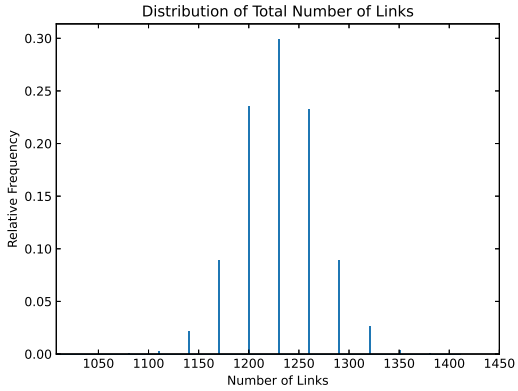
(b) Distribution of the number of links added between two time steps.



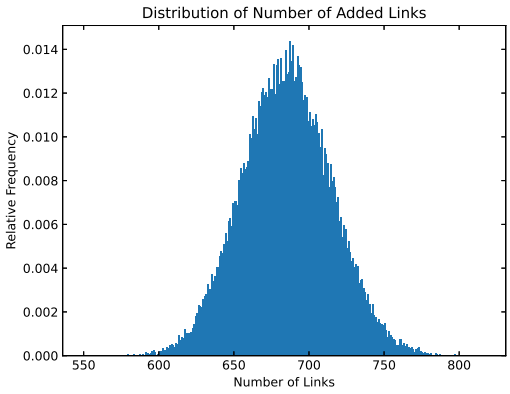
(c) Distribution of the number of links removed between two time steps.

Figure A.5: Histograms of link statistics for the quantized MMP model with step size 20 across 100 trials of 1,000 time steps.

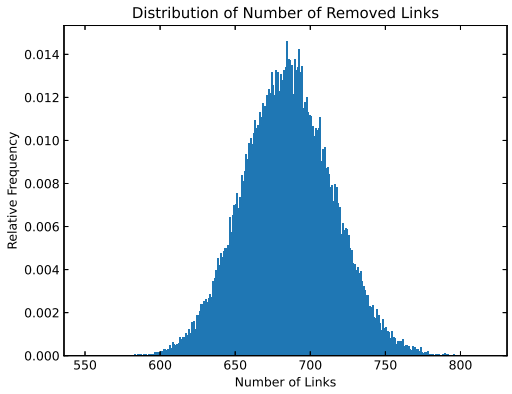
A.6. Quantized MMP Model, Step Size 30



(a) Distribution of the total number of links at each time step.



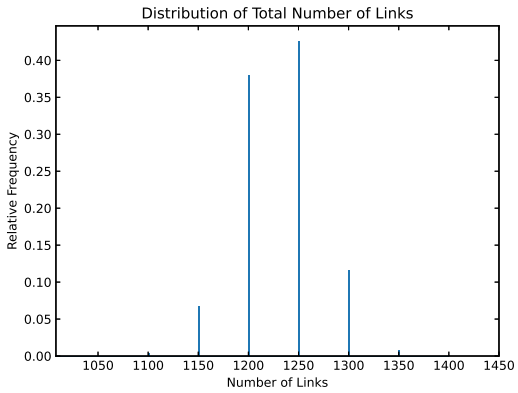
(b) Distribution of the number of links added between two time steps.



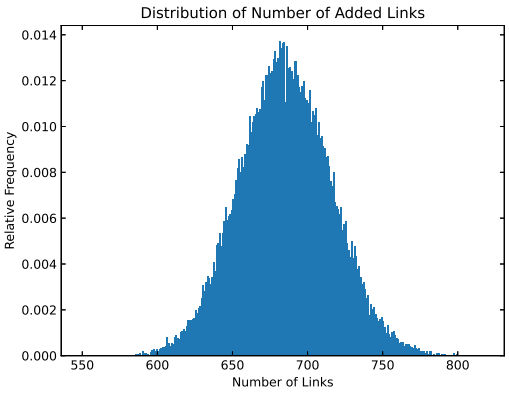
(c) Distribution of the number of links removed between two time steps.

Figure A.6: Histograms of link statistics for the quantized MMP model with step size 30 across 100 trials of 1,000 time steps.

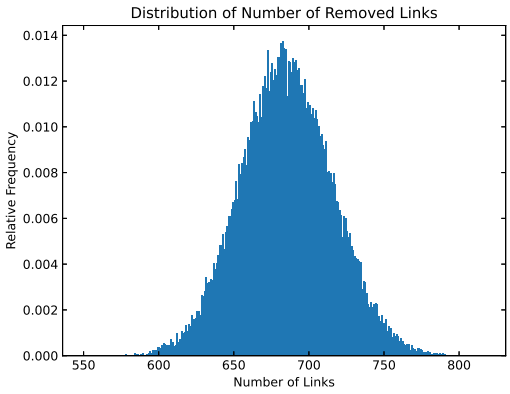
A.7. Quantized MMP Model, Step Size 50



(a) Distribution of the total number of links at each time step.



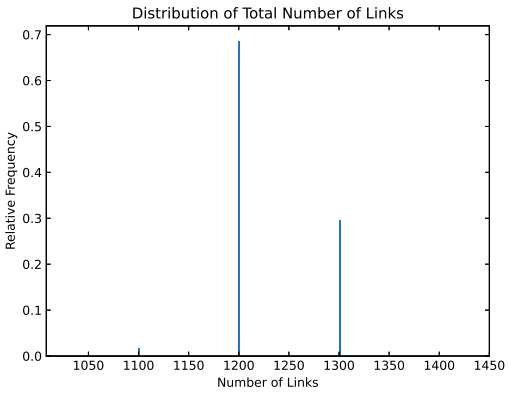
(b) Distribution of the number of links added between two time steps.



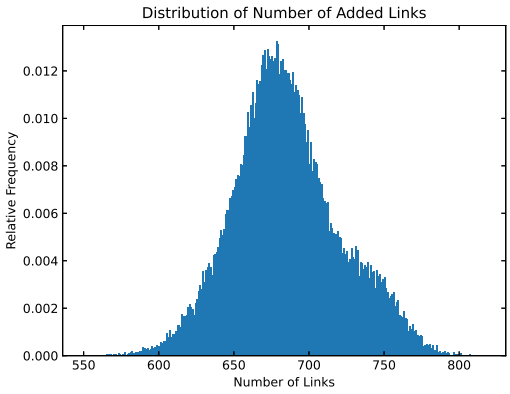
(c) Distribution of the number of links removed between two time steps.

Figure A.7: Histograms of link statistics for the quantized MMP model with step size 50 across 100 trials of 1,000 time steps.

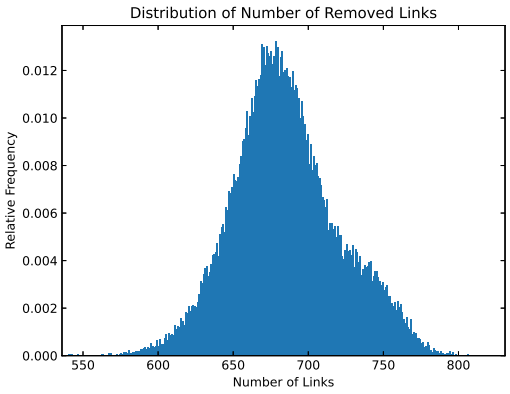
A.8. Quantized MMP Model, Step Size 100



(a) Distribution of the total number of links at each time step.



(b) Distribution of the number of links added between two time steps.



(c) Distribution of the number of links removed between two time steps.

Figure A.8: Histograms of link statistics for the quantized MMP model with step size 100 across 100 trials of 1,000 time steps.

B

Link Distribution Statistics

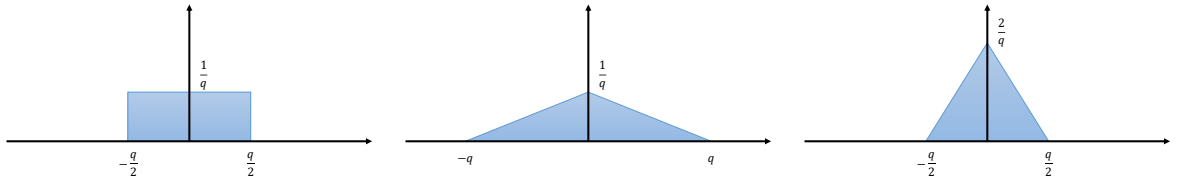
Table B.1: Mean and standard deviation of the the link distributions of the mobility process and MMP models.

Model	**Mean**			**Std. Dev.**		
	Links	Added	Removed	Links	Added	Removed
Mobility	1229.07	684.66	683.42	55.218	30.571	36.791
Aggregated	1228.96	684.55	683.32	55.185	30.452	36.777
Quant. 5	1228.76	684.81	683.58	55.037	30.449	36.790
Quant. 10	1229.24	684.90	683.67	55.384	30.720	36.924
Quant. 20	1228.75	684.81	683.57	55.651	30.878	37.019
Quant. 30	1229.03	684.88	683.65	55.881	31.215	37.361
Quant. 50	1228.88	684.97	683.74	56.993	32.117	38.086
Quant. 100	1226.81	684.96	683.74	62.833	37.079	42.084

C

Quantization Error of Actions

From [9, p. 462], the probability density function (PDF) of the quantization error of a signal quantized by rounding is known to be uniform and follows Figure C.1a. In the quantized MMP model, the number of links at each time step is quantized using rounding. The actions of the MMP model are evaluated by taking the difference between two time steps, and therefore, the quantization error in the action is equal to the difference between the quantization error of the two time steps. The PDF of the error in the action therefore follows from the convolution the PDF of the error in the links with itself, resulting in Figure C.1b. Because the action remapping strategy spreads the error equally over the number of added and removed links, the PDF of the error in the added/removed links will be half as wide, as shown in C.1c.



(a) PDF of quantization error in the number of links.

(b) PDF of quantization error in the actions.

(c) PDF of quantization error in the number of added/removed links.

Figure C.1: Probability distributions functions of the quantization errors.

The PDF of the quantization error for the added and removed links can be expressed as

$$p(\epsilon) = \begin{cases} \frac{4}{q^2}\epsilon + \frac{2}{q} & -\frac{q}{2} \leq \epsilon \leq 0, \\ -\frac{4}{q^2}\epsilon + \frac{2}{q} & 0 \leq \epsilon \leq \frac{q}{2}, \\ 0 & \text{otherwise.} \end{cases} \tag{C.1}$$

The power in the quantization error follows from solving the integral

$$\sigma^2 = \int_{-q/2}^{q/2} \epsilon^2 p(\epsilon) d\epsilon = \frac{q^2}{24}. \tag{C.2}$$

The power of the quantization error for the added and removed links is equal to half of the power of the quantization error in the number of links in the graph. The RMS error is the square root of the power, and therefore the RMS error for the added and removed links is a factor of $\sqrt{2}$ less than the RMS error of the number of links.

D

Summary of SIR Simulation Results

Table D.1 summarizes the results of the SIR simulations; an explanation of the table follows.

Column 1: Model
In total, 8 models are considered: the mobility process (denoted as "Mobility"), the aggregated MMP model (denoted as "Aggregated"), and the the quantized MMP model for quantization step sizes $q \in \{5, 10, 20, 30, 50, 100\}$ (denoted as "Quant. q").

Column 2-3: Simulation Parameters
The parameters of the SIR simulation are the infection probability $\beta \in \{0.1, 0.3, 0.5, 0.7\}$ and the curing probability $\delta \in \{0.1, 0.3, 0.5, 0.7\}$. For every combination of β and δ, an SIR simulation is performed for each of the models.

Column 4-6: Simulation Duration Results
Each SIR simulation consists of 100 trials and a trial ends when there are no infectious nodes remaining. The mean duration, maximum duration, and standard deviation of the duration across 100 trials is calculated (column 4, column 5, and column 6 respectively). The duration is measured in time steps.

Column 7-9: Infection Results
The number of infectious node at each time step is calculated and averaged over the 100 trials. The average number of infectious nodes is divided by the total number of nodes $N = 1024$ to find the average fraction of infectious nodes at each time step. The peak value of the fraction of infectious nodes and the time step at which it occurs has been calculated (column 8 and column 7 respectively). The cumulative number of nodes which are infected during the simulation, averaged over 100 trials, has also been calculated and is expressed as a fraction of the total number of nodes (column 9).

Table D.1: Summary of SIR simulation results.

Model	**Params.**		**Simulation Duration**			**Infections**		
	β	δ	Mean	Max.	Std. Dev.	Peak Time	Peak Value	Total Infected
Mobility	0.1	0.1	109.7	318	93.2	73	0.05784	0.43916
Aggregated	0.1	0.1	87.3	223	75.0	58	0.08906	0.46775
Quant. 5	0.1	0.1	93.9	246	73.6	56	0.09420	0.50514
Quant. 10	0.1	0.1	88.8	267	74.3	58	0.09331	0.47549
Quant. 20	0.1	0.1	89.4	244	74.8	58	0.09152	0.47770
Quant. 30	0.1	0.1	91.6	286	74.6	57	0.09595	0.49337
Quant. 50	0.1	0.1	92.6	220	73.4	57	0.09704	0.50195
Quant. 100	0.1	0.1	90.1	245	73.7	58	0.09413	0.48888

continued on next page ...

... continued

Model	Params.		Simulation Duration			Infections		
	β	δ	Mean	Max.	Std. Dev.	Peak Time	Peak Value	Total Infected
Mobility	0.1	0.3	6.8	63	6.9	0	0.00098	0.00277
Aggregated	0.1	0.3	7.8	114	9.4	0	0.00098	0.00410
Quant. 5	0.1	0.3	7.3	92	8.6	0	0.00098	0.00372
Quant. 10	0.1	0.3	7.4	62	8.1	0	0.00098	0.00376
Quant. 20	0.1	0.3	7.7	76	8.5	0	0.00098	0.00393
Quant. 30	0.1	0.3	7.4	72	8.4	0	0.00098	0.00364
Quant. 50	0.1	0.3	7.6	75	8.8	0	0.00098	0.00372
Quant. 100	0.1	0.3	7.4	75	8.3	0	0.00098	0.00361
Mobility	0.1	0.5	3.8	36	2.6	0	0.00098	0.00161
Aggregated	0.1	0.5	3.9	20	2.6	0	0.00098	0.00172
Quant. 5	0.1	0.5	3.7	25	2.6	0	0.00098	0.00168
Quant. 10	0.1	0.5	3.7	19	2.4	0	0.00098	0.00159
Quant. 20	0.1	0.5	3.7	24	2.4	0	0.00098	0.00162
Quant. 30	0.1	0.5	3.7	19	2.5	0	0.00098	0.00165
Quant. 50	0.1	0.5	3.8	23	2.6	0	0.00098	0.00172
Quant. 100	0.1	0.5	3.8	20	2.6	0	0.00098	0.00177
Mobility	0.1	0.7	2.9	12	1.4	0	0.00098	0.00139
Aggregated	0.1	0.7	2.8	14	1.4	0	0.00098	0.00138
Quant. 5	0.1	0.7	2.8	13	1.4	0	0.00098	0.00137
Quant. 10	0.1	0.7	2.8	11	1.3	0	0.00098	0.00136
Quant. 20	0.1	0.7	2.9	14	1.4	0	0.00098	0.00140
Quant. 30	0.1	0.7	2.8	16	1.3	0	0.00098	0.00136
Quant. 50	0.1	0.7	2.9	10	1.3	0	0.00098	0.00137
Quant. 100	0.1	0.7	2.8	17	1.4	0	0.00098	0.00141
Mobility	0.3	0.1	118.1	203	49.3	44	0.15520	0.83019
Aggregated	0.3	0.1	79.1	157	32.8	20	0.43257	0.85797
Quant. 5	0.3	0.1	78.8	169	32.8	20	0.42763	0.85705
Quant. 10	0.3	0.1	79.6	143	31.3	20	0.43587	0.87103
Quant. 20	0.3	0.1	78.4	159	33.5	20	0.42820	0.84907
Quant. 30	0.3	0.1	79.3	148	32.4	20	0.43032	0.86187
Quant. 50	0.3	0.1	80.0	172	30.9	20	0.43623	0.87491
Quant. 100	0.3	0.1	78.2	145	32.6	20	0.43080	0.85599
Mobility	0.3	0.3	48.4	226	54.1	39	0.00777	0.16600
Aggregated	0.3	0.3	32.4	81	25.4	22	0.08574	0.47294
Quant. 5	0.3	0.3	34.2	80	25.0	22	0.09159	0.50199
Quant. 10	0.3	0.3	33.9	87	25.2	22	0.09186	0.49758
Quant. 20	0.3	0.3	32.4	82	25.2	21	0.08792	0.47583
Quant. 30	0.3	0.3	33.7	89	25.2	22	0.08987	0.49479
Quant. 50	0.3	0.3	32.5	81	25.3	22	0.08648	0.47642
Quant. 100	0.3	0.3	35.1	83	25.1	22	0.09414	0.51668
Mobility	0.3	0.5	7.8	126	9.4	3	0.00109	0.00905
Aggregated	0.3	0.5	18.9	101	23.7	23	0.00979	0.13022
Quant. 5	0.3	0.5	19.7	121	24.5	24	0.00913	0.13115
Quant. 10	0.3	0.5	18.2	119	23.4	24	0.00858	0.11985
Quant. 20	0.3	0.5	18.7	114	23.7	25	0.00870	0.12454
Quant. 30	0.3	0.5	17.9	105	23.3	22	0.00869	0.12079
Quant. 50	0.3	0.5	18.7	112	24.4	25	0.00826	0.12331
Quant. 100	0.3	0.5	18.9	94	23.8	24	0.00919	0.12942
Mobility	0.3	0.7	4.4	36	3.3	0	0.00098	0.00368
Aggregated	0.3	0.7	6.2	53	8.0	0	0.00098	0.01075
Quant. 5	0.3	0.7	6.2	57	7.8	0	0.00098	0.01142
Quant. 10	0.3	0.7	5.7	64	7.2	0	0.00098	0.00944

continued on next page ...

... continued

Model	Params.		Simulation Duration			Infections		
	β	δ	Mean	Max.	Std. Dev.	Peak Time	Peak Value	Total Infected
Quant. 20	0.3	0.7	5.8	67	7.7	0	0.00098	0.01098
Quant. 30	0.3	0.7	5.8	51	6.9	0	0.00098	0.00982
Quant. 50	0.3	0.7	6.2	62	8.2	0	0.00098	0.01148
Quant. 100	0.3	0.7	6.4	73	8.4	0	0.00098	0.01243
Mobility	0.5	0.1	116.8	205	35.7	41	0.18461	0.91479
Aggregated	0.5	0.1	78.9	138	23.3	14	0.58347	0.93396
Quant. 5	0.5	0.1	79.4	143	23.1	14	0.58575	0.93798
Quant. 10	0.5	0.1	77.4	144	25.5	14	0.57327	0.91498
Quant. 20	0.5	0.1	79.4	145	22.2	14	0.58498	0.94198
Quant. 30	0.5	0.1	78.5	162	24.4	14	0.57473	0.92595
Quant. 50	0.5	0.1	79.2	134	21.7	14	0.58820	0.94394
Quant. 100	0.5	0.1	78.9	160	22.9	14	0.58453	0.93795
Mobility	0.5	0.3	86.8	251	59.6	34	0.01798	0.40172
Aggregated	0.5	0.3	31.2	63	15.4	14	0.24171	0.75196
Quant. 5	0.5	0.3	31.2	59	15.0	13	0.24453	0.76011
Quant. 10	0.5	0.3	30.6	55	15.3	13	0.23826	0.74438
Quant. 20	0.5	0.3	30.8	62	15.3	13	0.24436	0.74719
Quant. 30	0.5	0.3	30.5	62	15.3	13	0.23575	0.74629
Quant. 50	0.5	0.3	30.8	67	15.1	13	0.24325	0.75297
Quant. 100	0.5	0.3	31.7	62	14.8	13	0.24521	0.77066
Mobility	0.5	0.5	17.6	135	21.0	8	0.00261	0.04166
Aggregated	0.5	0.5	21.2	56	14.4	14	0.09804	0.51810
Quant. 5	0.5	0.5	22.3	49	14.6	14	0.09945	0.53468
Quant. 10	0.5	0.5	21.3	48	14.8	14	0.09589	0.51078
Quant. 20	0.5	0.5	21.7	50	14.5	14	0.09817	0.52745
Quant. 30	0.5	0.5	22.1	50	14.5	14	0.09686	0.52901
Quant. 50	0.5	0.5	21.7	56	14.7	14	0.09601	0.52359
Quant. 100	0.5	0.5	21.5	53	14.5	14	0.09707	0.52008
Mobility	0.5	0.7	6.7	45	5.6	3	0.00131	0.00921
Aggregated	0.5	0.7	17.2	52	15.2	15	0.03288	0.30514
Quant. 5	0.5	0.7	17.7	57	15.3	15	0.03275	0.30982
Quant. 10	0.5	0.7	18.1	59	15.4	15	0.03392	0.32005
Quant. 20	0.5	0.7	16.8	61	15.0	14	0.03270	0.29865
Quant. 30	0.5	0.7	17.8	64	15.1	15	0.03408	0.31826
Quant. 50	0.5	0.7	17.6	58	15.4	15	0.03201	0.30731
Quant. 100	0.5	0.7	17.5	59	15.2	15	0.03394	0.31354
Mobility	0.7	0.1	115.7	190	29.6	39	0.19823	0.94607
Aggregated	0.7	0.1	79.2	141	19.1	11	0.66991	0.96603
Quant. 5	0.7	0.1	77.7	134	20.3	11	0.65767	0.95304
Quant. 10	0.7	0.1	77.9	162	20.3	11	0.66279	0.95503
Quant. 20	0.7	0.1	78.9	132	18.6	11	0.66703	0.96502
Quant. 30	0.7	0.1	78.1	140	20.1	11	0.66148	0.95604
Quant. 50	0.7	0.1	79.4	153	18.4	11	0.67218	0.96902
Quant. 100	0.7	0.1	78.0	144	19.1	11	0.67090	0.96303
Mobility	0.7	0.3	100.7	264	53.5	24	0.02400	0.52502
Aggregated	0.7	0.3	29.5	59	11.1	10	0.35363	0.85913
Quant. 5	0.7	0.3	29.9	52	10.9	10	0.36101	0.86392
Quant. 10	0.7	0.3	29.7	49	11.0	10	0.35695	0.86175
Quant. 20	0.7	0.3	30.1	53	10.6	10	0.35947	0.87278
Quant. 30	0.7	0.3	29.7	53	11.0	10	0.35730	0.85990
Quant. 50	0.7	0.3	29.8	56	10.5	10	0.36279	0.87173
Quant. 100	0.7	0.3	29.7	53	10.8	10	0.35734	0.86390

continued on next page ...

...continued

Model	Params.		Simulation Duration			Infections		
	β	δ	Mean	Max.	Std. Dev.	Peak Time	Peak Value	Total Infected
Mobility	0.7	0.5	29.2	131	26.3	8	0.00501	0.09098
Aggregated	0.7	0.5	20.8	40	9.5	10	0.20592	0.73658
Quant. 5	0.7	0.5	21.0	37	9.3	10	0.20517	0.74647
Quant. 10	0.7	0.5	20.6	38	9.5	10	0.20291	0.73305
Quant. 20	0.7	0.5	20.8	36	9.6	10	0.20185	0.73667
Quant. 30	0.7	0.5	20.9	37	9.4	10	0.20558	0.74485
Quant. 50	0.7	0.5	20.7	40	9.6	10	0.20650	0.73671
Quant. 100	0.7	0.5	20.5	37	9.7	10	0.20117	0.72773
Mobility	0.7	0.7	10.4	72	9.0	4	0.00233	0.02108
Aggregated	0.7	0.7	16.8	37	9.7	11	0.11149	0.58592
Quant. 5	0.7	0.7	17.2	33	9.4	11	0.11328	0.60125
Quant. 10	0.7	0.7	17.0	33	9.5	11	0.11203	0.59454
Quant. 20	0.7	0.7	16.7	34	9.8	11	0.11092	0.57708
Quant. 30	0.7	0.7	16.4	36	9.8	11	0.10711	0.56712
Quant. 50	0.7	0.7	17.1	36	9.5	10	0.11250	0.59861
Quant. 100	0.7	0.7	16.8	36	9.3	10	0.11623	0.59488

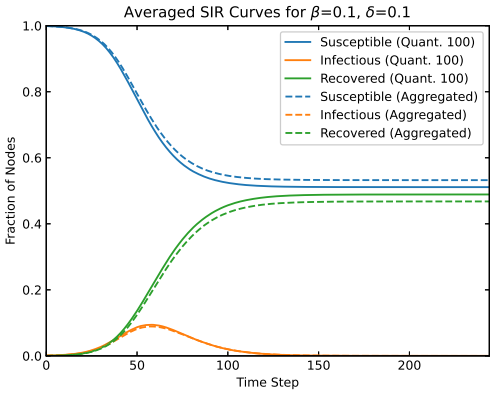
E

SIR Plots

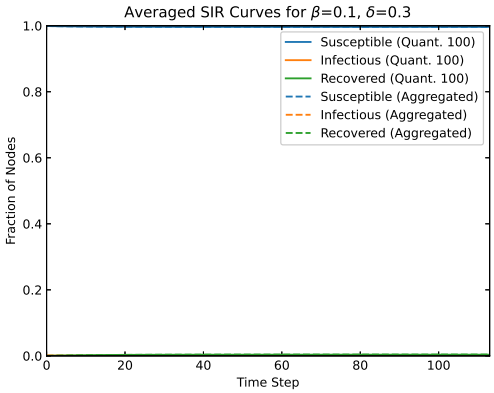
E.1. Comparison of Quantized MMP and Aggregated MMP

The SIR results of the quantized MMP with step size 100 are compared with the results of the aggregated MMP.

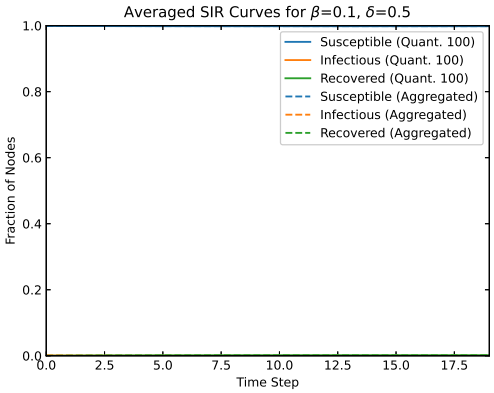
E.1.1. Comparison of Quantized MMP and Aggregated MMP for $\beta = 0.1$



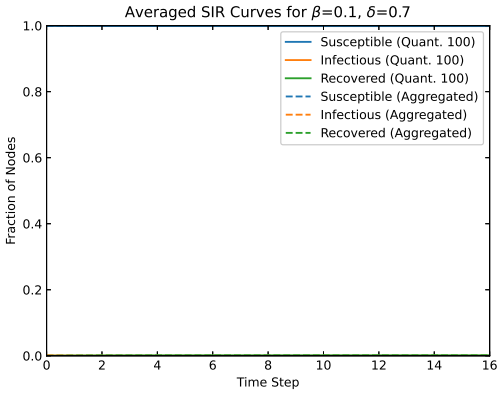
(a) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.1$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.1$ and $\delta = 0.3$.



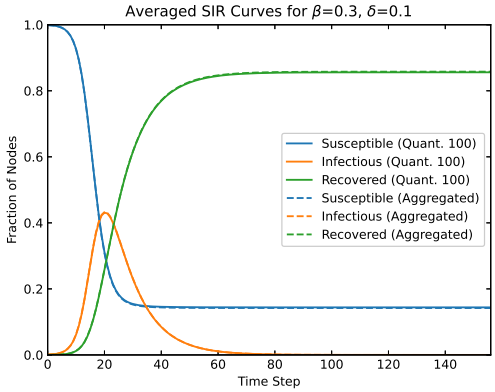
(c) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.1$ and $\delta = 0.5$.



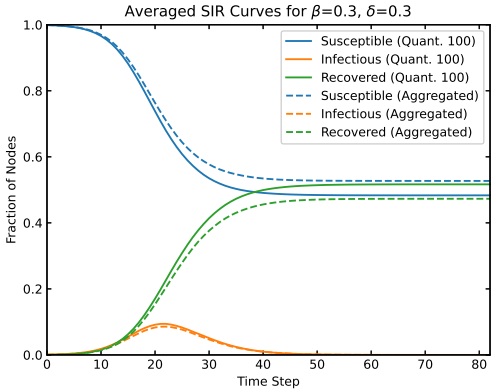
(d) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.1$ and $\delta = 0.7$.

Figure E.1: SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.1$ various values of δ.

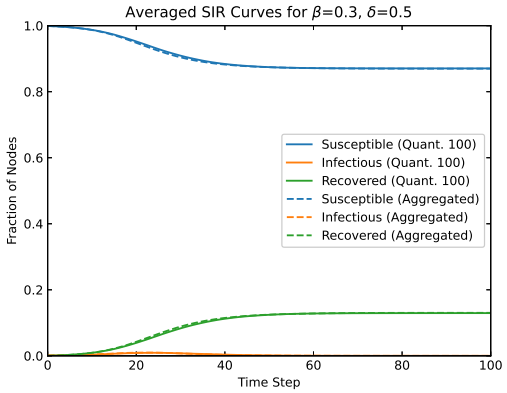
E.1.2. Comparison of Quantized MMP and Aggregated MMP for $\beta = 0.3$



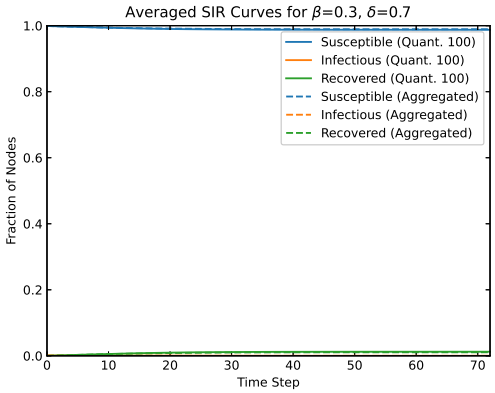
(a) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.3$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.3$ and $\delta = 0.3$.



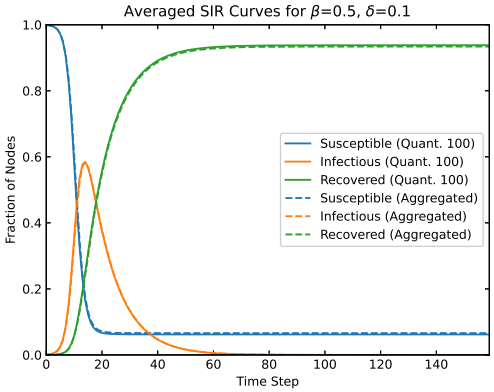
(c) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.3$ and $\delta = 0.5$.



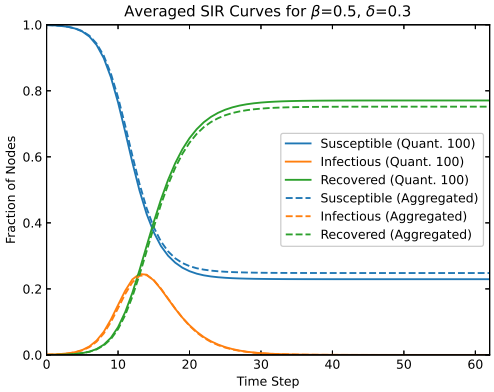
(d) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.3$ and $\delta = 0.7$.

Figure E.2: SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.3$ various values of δ.

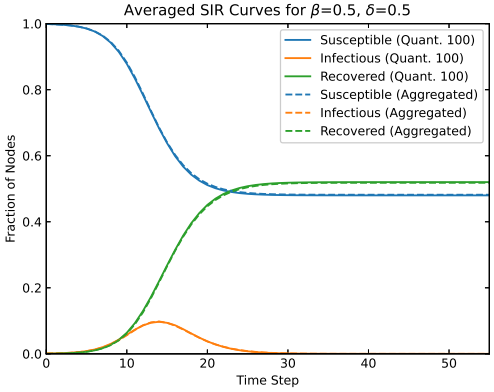
E.1.3. Comparison of Quantized MMP and Aggregated MMP for $\beta = 0.5$



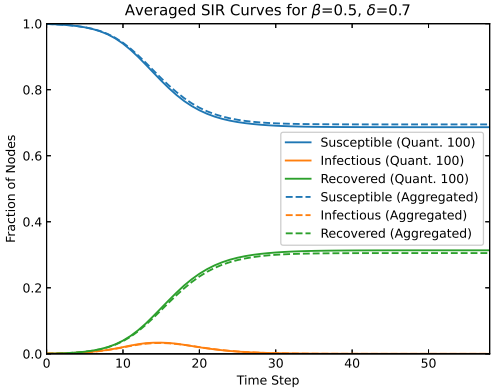
(a) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.5$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.5$ and $\delta = 0.3$.



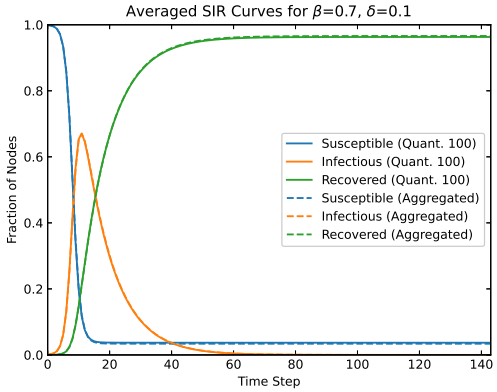
(c) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.5$ and $\delta = 0.5$.



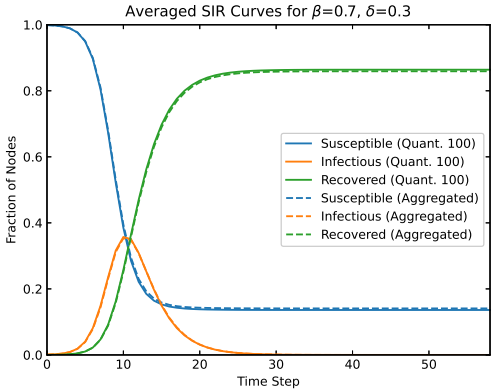
(d) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.5$ and $\delta = 0.7$.

Figure E.3: SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.5$ various values of δ.

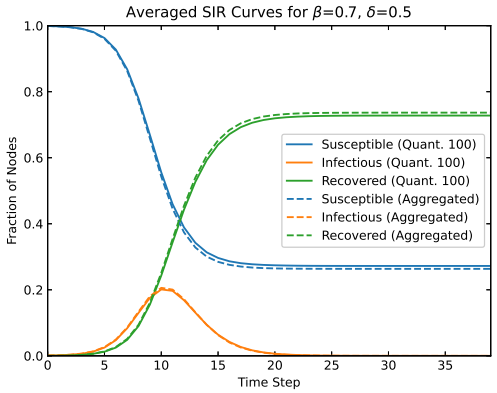
E.1.4. Comparison of Quantized MMP and Aggregated MMP for $\beta = 0.7$



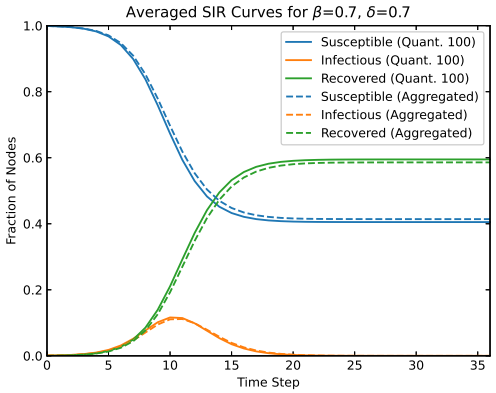
(a) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.7$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.7$ and $\delta = 0.3$.



(c) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.7$ and $\delta = 0.5$.



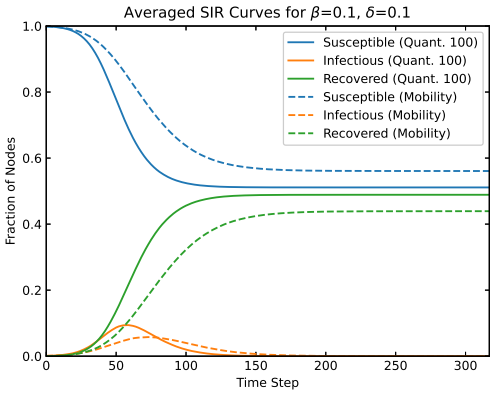
(d) SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.7$ and $\delta = 0.7$.

Figure E.4: SIR curves of the quantized MMP model with step size 100 and the aggregated MMP model for $\beta = 0.7$ various values of δ.

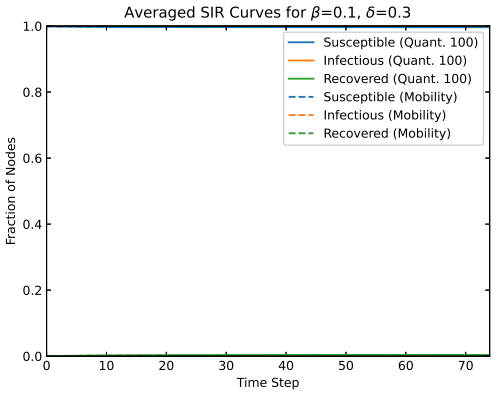
E.2. Comparison of Quantized MMP and Mobility Process

The SIR results of the quantized MMP with step size 100 are compared with the results of the mobility process.

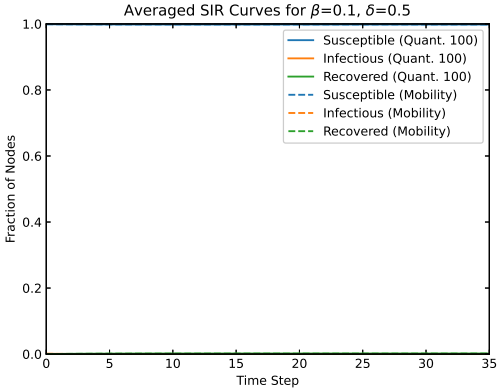
E.2.1. Comparison of Quantized MMP and Mobility Process for $\beta = 0.1$



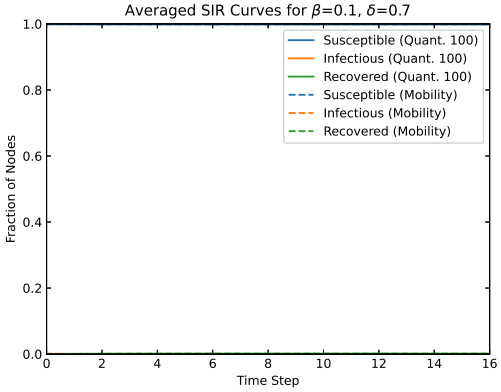
(a) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.1$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.1$ and $\delta = 0.3$.



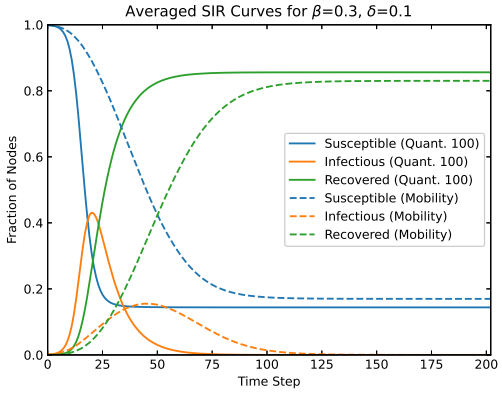
(c) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.1$ and $\delta = 0.5$.



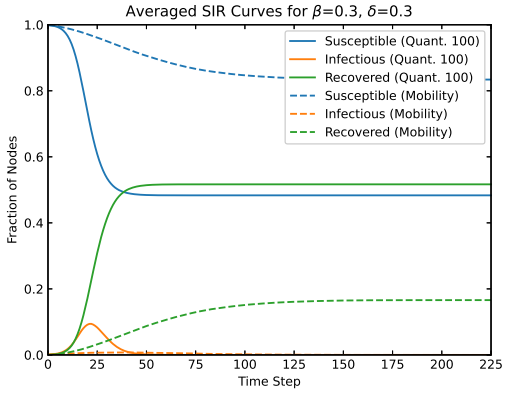
(d) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.1$ and $\delta = 0.7$.

Figure E.5: SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.1$ and various values of δ.

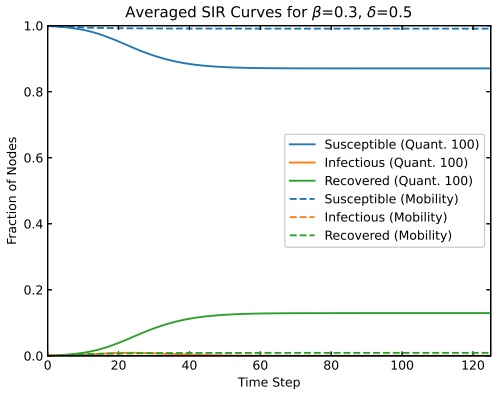
E.2.2. Comparison of Quantized MMP and Mobility Process for $\beta = 0.3$



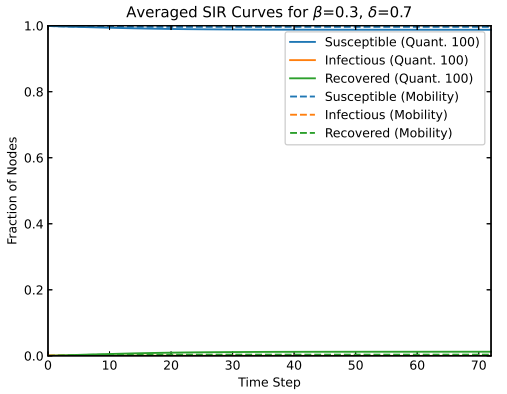
(a) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.3$.



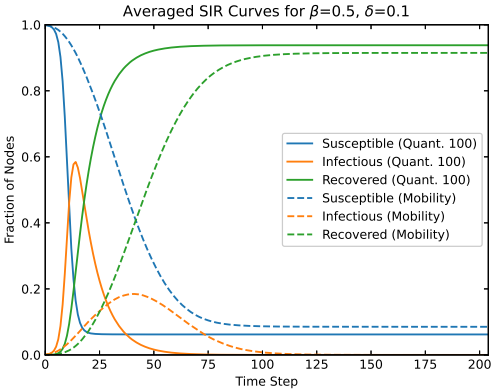
(c) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.5$.



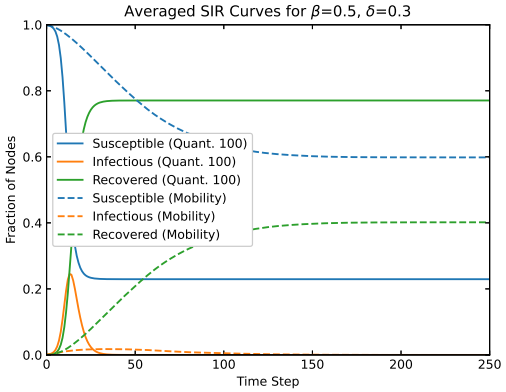
(d) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and $\delta = 0.7$.

Figure E.6: SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.3$ and various values of δ.

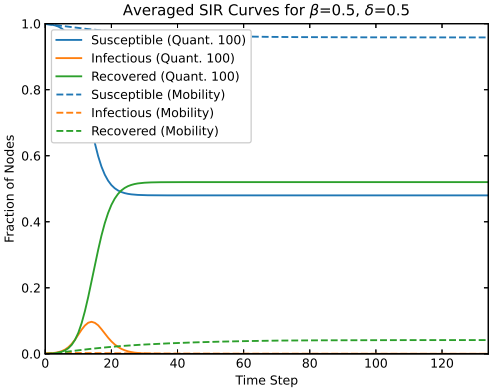
E.2.3. Comparison of Quantized MMP and Mobility Process for $\beta = 0.5$



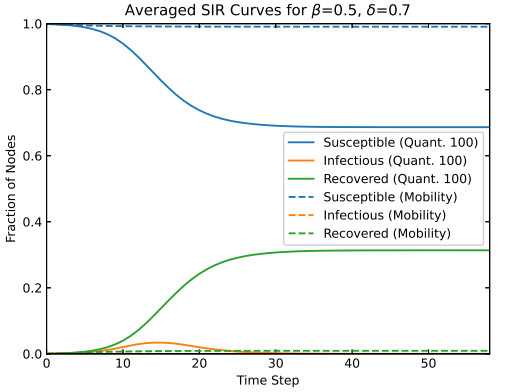
(a) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.5$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.5$ and $\delta = 0.3$.



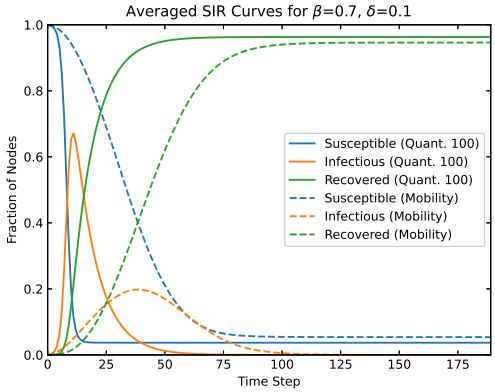
(c) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.5$ and $\delta = 0.5$.



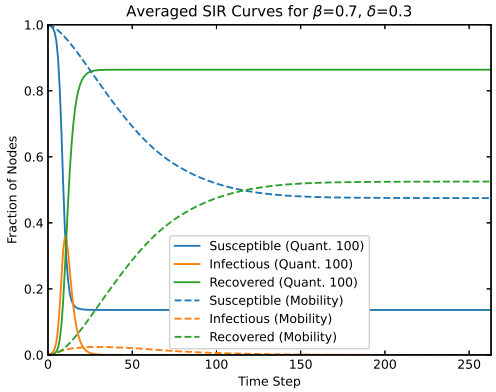
(d) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.5$ and $\delta = 0.7$.

Figure E.7: SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.5$ and various values of δ.

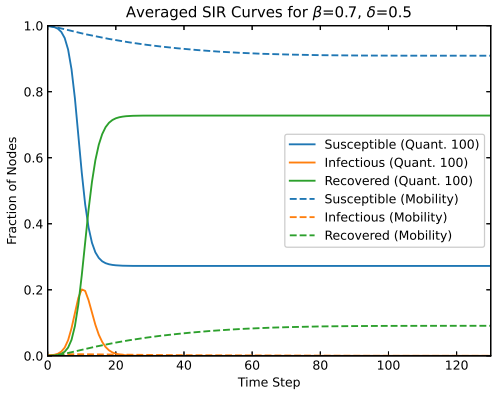
E.2.4. Comparison of Quantized MMP and Mobility Process for $\beta = 0.7$



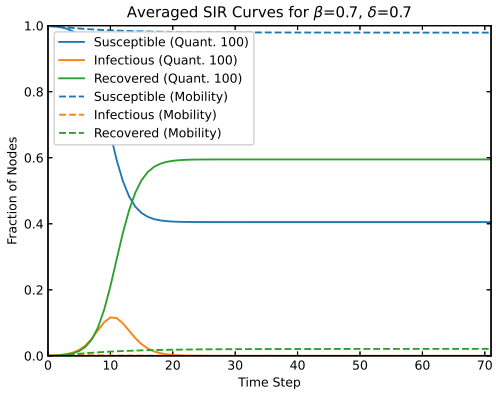
(a) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.7$ and $\delta = 0.1$.



(b) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.7$ and $\delta = 0.3$.



(c) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.7$ and $\delta = 0.5$.



(d) SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.7$ and $\delta = 0.7$.

Figure E.8: SIR curves of the quantized MMP model with step size 100 and the mobility process for $\beta = 0.7$ and various values of δ.