Advanced Discrete Modelling: Hybrid modelling

Assignment 4, Group 14

Emma Castaños, Laura Crowley, Rob Dregmans, Ivar Tjallingii, Martijn van Vliet

Table of Contents

1 Introduction

1.1 Situation Sketch

The World Bank is in need of a tool that uses known traffic data to drive a model of the N1, simulate the traffic delays due to bridge breakdowns and generates live visualisations of relevant KPIs. This will enable the World Bank to make better investment decisions. Firstly, it enables the World Bank to test policies and immediately review all results during the execution of a run. Secondly, it allows for the modelling of a road real time over 24 hours and the inclusion of time related events, such as breaking down bridges at a certain point of a day during certain runs. This could add to the knowledge of the most critical road segments of the N1 road.

1.2 Solution Proposal

In order to make this modelling tool, several key components can be distinguished, see figure 1. A simulation model (Simio) takes the road and traffic data as input and then outputs information that eventually should be visualized and presented to the client. This information is stored in tables in a database on the MySQL server. The MySQL server allows the simulation model to write data to tables in the database hosted on the server and it allows the visualization engine (Python) to read data in the tables in the database. The visualization engine can visualize the results in real time (every hour) as it reads data that is being written by the simulation model in real time.

By linking these three main components (Simio model, MYSQL database, Python) we construct an integrated, distributed system which can then be used to review the model results real time (which would not be possible in Python alone).

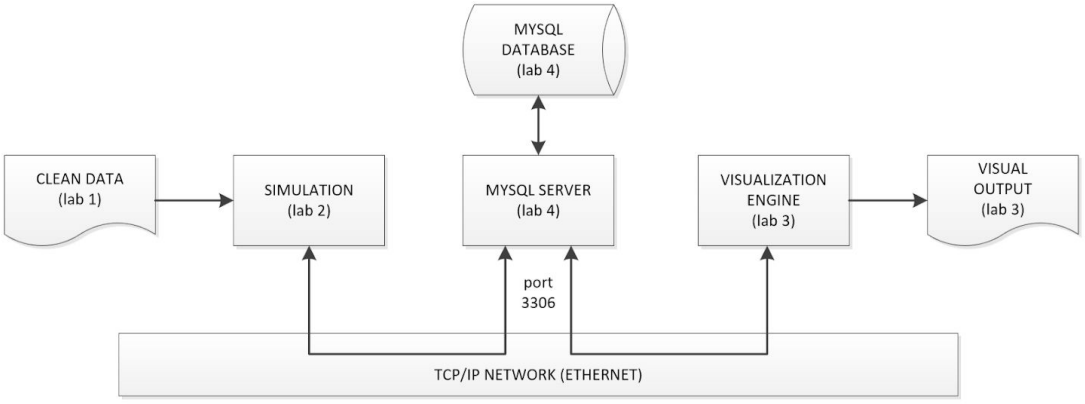

Figure 1: Setup of simulation tool

1.3 Key Performance Indicators

To measure the results of the model we defined KPIs. The most important is the average travel time. This travel time is calculated for all modes of transport and for all segments as well. This general KPI is therefore defined more specifically by four KPIs that will be used in this report. The first KPI is the Average travel time over the full N1 road, the second KPI is the average travel time from Dhaka to Chittagong, the third KPI is the traffic density of each road segment and distribution of traffic types over that segment.. These KPIs are used in chapter four to visualize the effect of bridges breaking down over time on traffic along the N1 road.

2 MySQL Database

The MySQL server (and the database and tables within) are the communication bridge between the simulation software and the visualization engine. MySQL is an incredibly powerful tool that, when set up properly, can act as this bridge. As such, a detailed explanation of the MySQL set up will be outlined so that it can be replicated.

2.1 Database Name, User ID, and Password

In order to access the database (for Simio to write data and for Python to read data), the following User ID and Password is required:

Database =	epa1351group14	
User =	root	(or if this does not work epa1351g14)
Password =	xgt65RR**	(and password xgt65RR##)

2.2 Database Contents

The “epa1351group14” database contains multiple tables with their own structures and data that they are storing. These tables are named “hourlyupdate”, “segmenttraveltimes”, and “brokenbridges,” respectively. The structure and descriptive headers of the data contained within each table will be described in tables 1-3.

hourlyupdate		
ColumnName:	**RowID1**	**SimioHour**
Simio data type:	integer	real (mins)
MySQL data type:	INT(11)	DOUBLE

Table 1: contents of the hourlyupdate table, functions as a semaphore

segmenttraveltimes				
ColumnName:	**RowID2**	**LRPName**	**VehicleType**	**SegTime**
Simio data type:	integer	string	string	real
MySQL data type:	INT(11)	VARCHAR(45)	VARCHAR(45)	DOUBLE

Table 2: contents of the segmenttraveltimes table

brokenbridges			
ColumnName:	**RowID3**	**BridgeID**	**TimeBroken**
Simio data type:	integer	string	real
MySQL data type:	INT(11)	VARCHAR(45)	DOUBLE

Table 3: contents of the brokenbridges table

2.3 Database Connections

In order to perform its bridging functions, the MySQL database is connected with Simio (the simulation model) and Python (the visualization engine).

2.3.1 Simio & MySQL

In the Simio model there is a Db Connect Element which allows the simulation model to read and write data to the MySQL database. This Db Connect Element has the following connection string which allows for the Simio and MySQL database connection to take place.

> "Server = localhost;
> Port = 3306;
> Database = epa1351group14;
> Uid = epa1351g14;
> Pwd = xgt65RR##"

During the simulation run, an Add On Process can read, write, and delete data from the tables in the MySQL database via the connection and access granted by the Db Connect Element. As a result, Simio writes its output data to the MySQL database in real time.

2.3.2 Python & MySQL

In Python, a package that connects Python and MySQL is used in order to allow the visualization engine to read data from the MySQL database. Due to the fact that various group members in this assignment were able to install different packages that achieve the same Python - MySQL connection, either the package "mysql.connector" or "pymysql.cursors" was used. The two packages achieve the same results with the same functionality. Namely, they set up a connection to the MySQL database using a connection string and then they allow for SQL statements to be executed in the Python shell on the database. The set up connection string is as follows:

> User = 'epa1351g14',
> Password = 'xgt65RR##',
> Database = 'epa1351group14'

Then, using SQL statements in the Python shell, such as "SELECT" and "ORDER BY," data from the MySQL database can be read in to python in real time.

An important feature of the python script used in this database connection is the "while loop" that allows the program to remain open and check for new entries in the "semaphore" table called "hourlyupdates." When the condition is met that a new entry has been added by Simio to the "hourlyupdates" table in the MySQL database, Python queries the other data tables in the database and performs visualizations based on that new data. This is shown in figure 2.

```
#STEP 6: Definition to update all the visualisations
def wait_for_sql():
    last_rowid = -1
    while True:
        #cnx for option 1
        cnx = mysql.connector.connect(user='epa1351g14', password='xgt65RR##', database='epa1351group14')
        #cnx for option 2
        #cnx = pymysql.connect(user='epa1351g14', password='xgt65RR##', database='epa1351group14')
        cursor = cnx.cursor()

        #selecting the latest row in hourlyupdate table
        query = ("SELECT * from hourlyupdate ORDER BY RowID1 DESC LIMIT 1")
        cursor.execute(query)

#for loop searching for a new row in the hourlyupdate table
        for (RowID1, SimioHour) in cursor:
            if RowID1 > last_rowid:
                clear_output()

                #Reading in the dataframes
                read_stt_sql(cnx)
                segmenttraveltimes = read_stt_sql(cnx)
                read_bb_sql(cnx)
                brokenbridges = read_bb_sql(cnx)

                #Visualisations
                movingaverage(segmenttraveltimes)
                vehiclespersegment(segmenttraveltimes)
                vehicletypes(segmenttraveltimes)
                #bridgesdown(brokenbridges)

                #change the value of last_rowid to the current Row ID
                last_rowid = RowID1
        time.sleep(1)
```

Figure 2: While loop that checks for updates to the semaphore table

Another important part of the python script generates the visualizations, which is explained in more detail in section 4 of the report.

3 Simio Model

The Simio model of the N1 must be able to represent the traffic over each segment of the N1 road. It is difficult to represent the actual traffic over time during a day because the available road data is sampled over a whole year and then divided by the number of days in a year. Also, the ADDT data does not specify the time interval during which the data was collected, which could include peak congestion hours or might have been taken during a not very busy time of day. As a result, certain assumptions must be made in order to model the traffic along each segment of the N1 road.

3.1 Main Assumptions

The main goal of this assignment is to build a distributed system made up from the different components in Python, MySQL and Simio. The focus was not on generating a representative road system in the Simio model. Therefore, all congestion formed due to limited road capacity related to high traffic on a certain road segments was omitted during the model creation.

The N1 road was divided into the 69 segments provided by the traffic data used in previous assignments. The ADDT data that is known for each start node of a segment will represent the traffic over that entire segment. For each segment, the KPIs (including total delay and total travel time) will be computed. It is assumed that the KPIs of each segment are appropriate proxies for overall KPIs for the whole N1 road (for example, the total travel time).

3.2 Data Preparation for Simo

For the data preparation we used as input file the traffic data per segment and transformed this data in order to use it in Simio, see section 3.2.1.

3.2.1 Converting Traffic Data

	LRPName	Lat	ControlTraffic	Lon	Chainage	Description	Length	Cat	Next Node	BridgesDelay	Heavy Truck Create	Heavy Truck Destroy	Medium Truck Create	Medium Truck Destroy	
0	LRPS	23.706028	ControlTraffic	90.443333	0.000	0	0.0		LRPSa	DestroyTraffic	28.10267	0.0	503.61365	0.0	25
1	LRPSa	23.702917		90.450417	0.814	0	0.0		LRPSb		NaN	NaN	NaN	NaN	
2	LRPSb	23.702778		90.450472	0.822	0	0.0		LRP001		NaN	NaN	NaN	NaN	
3	LRP001	23.702139		90.451972	1.000	0	0.0		LRP001a		NaN	NaN	NaN	NaN	
4	LRP001a	23.698739		90.458861	1.800	.	11.3	A	LRP002	BridgesDelay	NaN	NaN	NaN	NaN	

Figure 3: Traffic data to be used in the simulation model

For the first LRP in each segment we defined per mode of traffic, how many of the total vehicles should be created or destroyed. To explain how we arrived at these numbers, a simple example is given below. If Segment B has 100 cars per day and Segment C 150 cars per day, we assume all cars from Segment B continue on Segment C and another 50 cars have to be created at the start of Segment C. So the cell value for Segment C will say ‘50’ under Cars Create. And conversely, if Segment D only has 75 cars per day (and Segment C

still has 150), 75 of them need to be destroyed and the rest can continue. This allows the simulation model to properly mimic the ADDT data that was collected by the ministry.

The difficulty of using Simio in this way to mimic the traffic data is that it is impossible to know which cars at what time should be destroyed from one segment to the next. As such, we assume a stochastic method for overcoming this difficulty, by defining a percentage of cars that need to be destroyed in the movement from one segment to the next, where the traffic on the latter segment is less than the former. In the example outlined previously (from Segment C to Segment D), for segment D the cell value of Cars Destroy will read 0.5. This means that every car that passes Segment D has a 50% chance of being destroyed in order to arrive at the total number of cars required in Segment D. For some segments there is no difference in number of traffic, therefore those columns contain only zeros.

To limit the computational burden of running to many individual entities, we scaled down all traffic with a factor 10. For scaled ADDT values lower than 1, we rounded the values up back to 1 because otherwise our add-on-process would produce undesired amounts of traffic entities.

	LRPName	Lat	ControlTraffic	Lon	Chainage	Description	Length	Cat	Next Node	BridgesDelay	Heavy Truck Create	Heavy Truck Destroy	Medium Truck Create	Medium Truck Destroy
0	LRPS	23.706028	ControlTraffic	90.443333	0.000	0	0.0		LRPSa	DestroyTraffic	28.10267	0.00000	503.61365	0.00000
18	LRP009	23.705028	ControlTraffic	90.519333	8.503	0	0.0		LRP009a	DestroyTraffic	9.07026	0.00000	563.14732	0.00000
27	LRP012	23.691805	ControlTraffic	90.542611	11.497	0	0.0		LRP012a	DestroyTraffic	2.81217	0.00000	80.70161	0.00000
32	LRP013	23.686527	ControlTraffic	90.550417	12.524	0	0.0		LRP013a	DestroyTraffic	30.20397	0.00000	60.98483	0.00000
61	LRP022	23.625222	ControlTraffic	90.604972	21.629	0	0.0		LRP022a	DestroyTraffic	4.68592	0.00000	80.67760	0.00000
89	LRP031	23.559972	ControlTraffic	90.654806	30.774	0	0.0		LRP031a	DestroyTraffic	0.00000	0.00000	0.00000	0.00000
96	LRP033	23.548750	ControlTraffic	90.667833	32.607	0	0.0		LRP033b	DestroyTraffic	0.00000	0.06473	0.00000	0.25370
128	LRP043	23.529416	ControlTraffic	90.751750	42.269	0	0.0		LRP043a	DestroyTraffic	0.00000	0.15867	0.00000	0.15867
179	LRP067	23.497277	ControlTraffic	90.967472	65.726	0	0.0		LRP067a	DestroyTraffic	0.00000	0.08911	95.41667	0.00000

Figure 4: Traffic data with the create and destroy ratios for segment transitions

3.2.2 Determining Bridge Breakdowns

During Assignment 3, each road segment received an importance score (based on their criticality and vulnerability scores). In this assignment, during the simulation run bridges will will break down based on these scores, along the top five most important segments of the N1. Using Python, the the top 5 most important segments based on their ‘segment score’ were selected, see figure 5.

	Road_Name	Link_No	Link_Name	Start LRP	End LRP	Total_A	Total_B	Total_C	Total_D	segment_score
35	N1	N1-36	Barabkunda Z1087-Fouzderhat Z1016 (Left)	LRP209	LRP225	18	6	2	2	0.096062
6	N1	N1-7	Langalband - Mograpara Chowrasta (Left)Z1089 (...	LRP013	LRP013	4	9	2	0	0.051326
68	N1	N1-69	Whykong Z1133 - Teknaf	LRP433	LRP467	74	25	38	3	0.045194
11	N1	N1-12	Daudkandi Z1062 - Pennai Z1053 (Left)	LRP033	LRP043	0	1	5	0	0.042074
23	N1	N1-24	Miabazar Z1046 - Chauddagram(Int.with Z1045) (...	LRP105	LRP105	7	12	0	0	0.041373

Figure 5: The top five most important road segments

In the Simio model, we will sample from all bridges that are located along these segments to break down a few of them using an add-on-process, see section 3.3.2 for more detail. For all Python operations during this step, please see Assignment4DataPrep-Jupiter-Notebook.

3.3 Model Overview

In this section we will provide an overview of the structure of the road model in Simio. We will address how the correct amount of traffic was generated on each road segment, the mechanism and logic behind broken bridges, and how the data was written to the MySQL database. The Simio model is generated using four main mechanisms:

- Auto creation of transfer nodes based on a data table in Simio
- Simulation of traffic over road segments via add-on-processes
- Break down of bridges over a simulation run
- Storing simio output in SQL databases for processing in Python

See figure 6, of a general overview of our total simulation tool which includes the most basic structure of the Simio model.

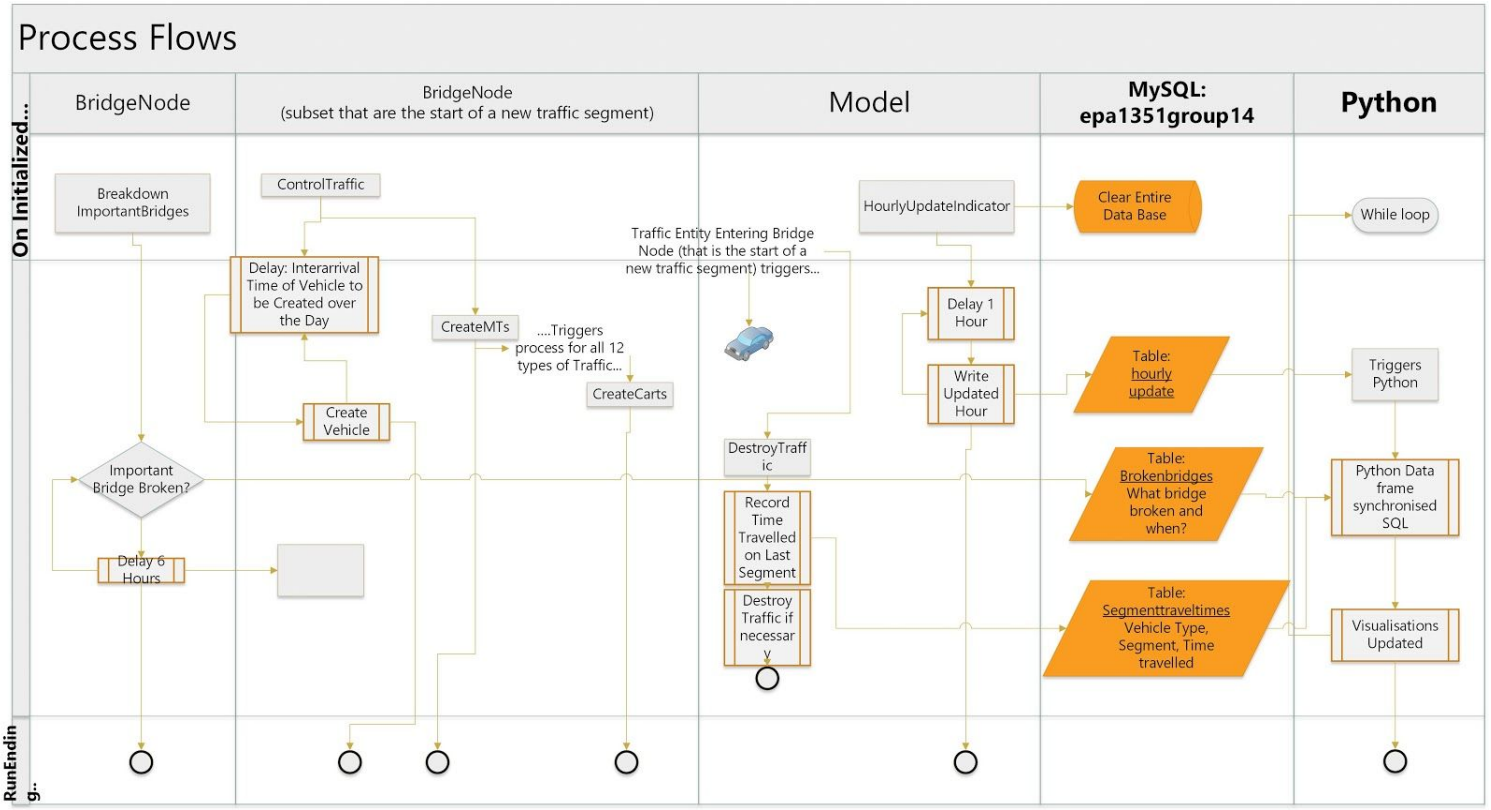

Figure 6: Overview of the Simio model, with MySQL and Python integration

3.3.1 Auto Creation of the N1 Road

We generate the model in a similar way as we did in assignment 2. Therefore we do not explain this step too extensively. Compared to assignment 2, the Bridge Table datatable in Simio now contains more information about certain LRP points. We still use the Bridge Table database to generate all LRP point using their Latitude and Longitude information. However, now the data table is also used to trigger various add-on-processes and to provide add-on-processes with required information to create and destroy the correct amount of traffic of a certain type (see section 3.3.2, for a detailed explanation). The bridge table now contains information about which bridges are sampled to be broken down, see section 3.3.3.

3.3.2 Simulation of Traffic Along Road Segments

The traffic on each road segment is controlled by CreateXs and DestroyXs add-on processes for each distinct road segment (where X represents a given vehicle type).

3.3.2.1 Create Traffic

The traffic of each category is created using the ControlTraffic add-on-process which triggers all add-on processes related to the creation of every traffic category. We briefly summarize how this works for the creation of medium trucks, see figure 7 for the add-on-process.

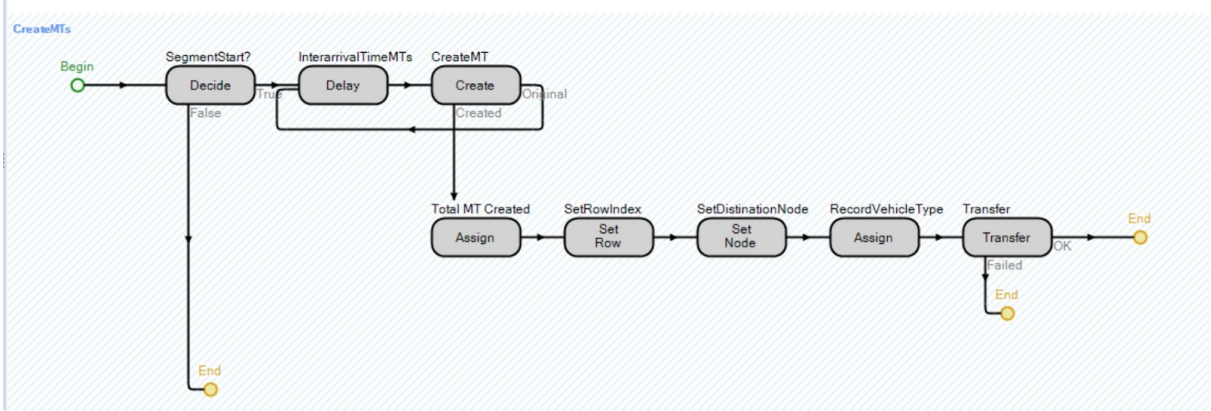

Figure 7: Add on process creation of Medium Trucks

The creation add-on-processes are run on initialize and continue to generate traffic throughout the whole run, based on the inter arrival time (because of the loop after 'CreateMT'). The basic logic of this add process is as follows:

- **Step 1:** Check if the transfer node is a start of a segment and if any medium trucks needed to be generated
- **Step 2:** Delay process to spread out the interarrival time across the day.The delay time is the time between each arrival. Thus if 50 trucks needed to be generated each hour, the interarrival time is 0.02 hour. The number of trucks that need to be generated are read from the BridgeTable for each road segment.
- **Step 3:** Create one new entity of mytraffic
- **Step 4:** Storage of total created Medium Trucks in the model (for debugging)
- **Step 5:** Assign index and destination node for the transfer step

- **Step 6:** Storage vehicle type (medium truck) and execute the transfer from free space to the next node in the model. We accept some travel time variation here because of transfer to the next node.

3.3.2.2 Destroy traffic

As the traffic that flows down a segment will reach the end of that segment, the traffic needs to adjusted to match the ADDT data of the next segment. Here there are two options for the Simio implementation of the traffic conversion concept introduced in section 3.2.1:

1. ***The traffic density of a certain category on the next road segment is higher than on the previous one.***
   This means that per unit of time, additional traffic of that category needs to be created. When this is the case, the control traffic add-on-process is used to generate precisely that amount, for example using the data in the column 'HeavyTruckCreate'. Within our data preparation see section 3.2.1, we have calculated exactly the amount of entities that need to be extra created to match the total traffic density of that entity category of that segment.
2. ***The traffic density of a category of the next segment is lower compared to the previous segment.***
   Then the traffic coming from the preceding segment will be scaled down to the traffic density of the new segment, thus destroying a certain percentage of all traffic that flows in from the previous segment. Herefore, we use the add-on-process 'DestroyTraffic'.

DestroyTraffic is triggered whenever a entity enters a new traffic segment. Therefore, it also forms the end of the previous traffic segment, so the first step in this process is to write the time traveled on the previous segment in the database (table segment travel times). In the next step, the row index of the table is updated and the start time of the new segment is assigned to the entity instance. This state value is used to calculate the time traveled at the beginning of the next segment.

After the data relating to travel time has been saved, the add-on-process determines if it needs to destroy incoming entities. In the case of option 1, the probability of destroying entities is set to zero, so all traffic is passed on to the next segment. If the probability is higher than zero, then that form the percentage of traffic of that category that is being destroyed over time. So for the example in figure 3, it will first check the vehicle type with 'Is HeavyTruck?'. The 'Destroy it?' process will determine based on the probability of that road segment if a passing Heavy Truck is being destroyed where the 'Destroy' process will execute the destroy-operation. This operation is repeated for every distinct vehicle type, for example for the Medium trucks below.

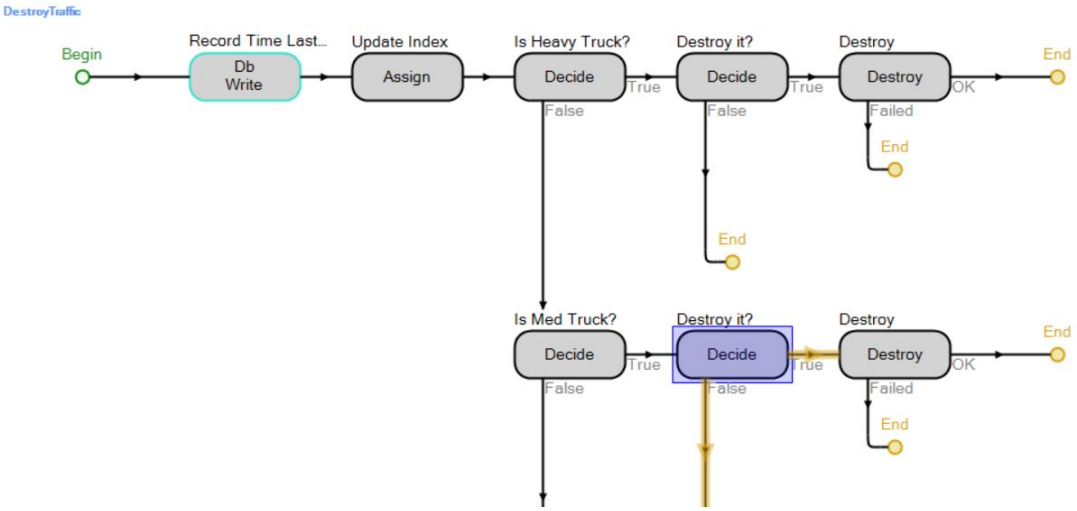

Figure 8: Add on process creation of Medium Trucks

3.3.3 Breaking Down Certain Bridges

Within the model, bridges on the most important segments of the N1 are randomly selected to be broken down. There are in total 154 bridges on those segments. With the add-on-process presented in figure 9, we break down five bridges over each simulation run.

This add-on-process is run on initialize for every transfer node, so it first needs to check if the transfer node is actually an 'important' bridge. Then it will decide, based on a probability, if the bridge is broken down. This probability is set to a value that along each run, approximately five bridges break down throughout the whole run.

Because we review our results real time, we break down bridges not only at the end of the run, but also throughout the run. To achieve this, we use a delay process of six hours to break down bridges along the run. The probability within 'Break down sample' has been calculated according to this delay time. So if we want five bridges broken down in 24 hours, the probability of the 'Break down sample' is set to 1/#importantbridges/4 * 5. Using the Db-Write function, we store in the broken bridges database file which bridges are broken down.

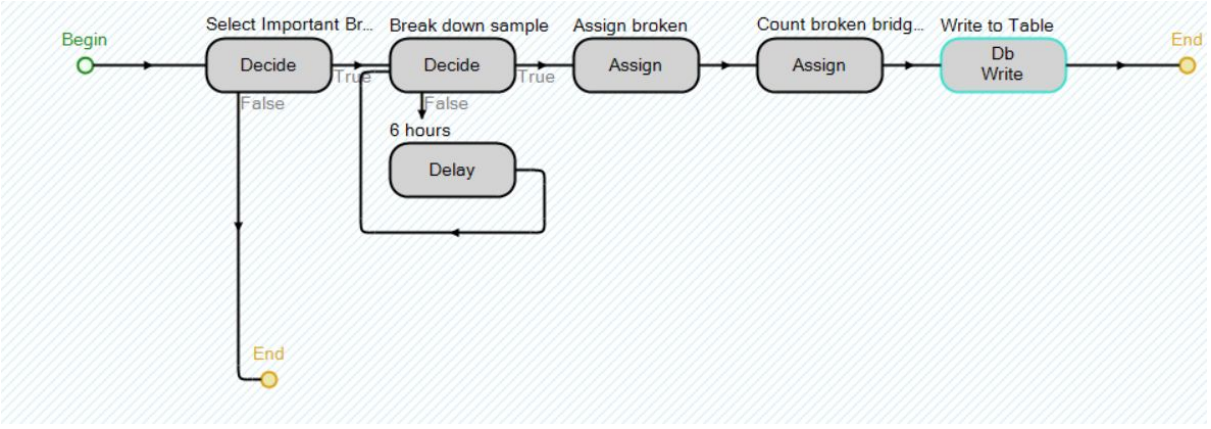

Figure 9: Add-on-process breaking down bridges in Simio

3.3.4 Storing Simio Output in Database

The SQL schema contains three tables, as described in section 2. Simio uses different AddOnProcesses to write the appropriate data to each of these tables during the model run. The first step is to create the connection from Simio to the database. This is done using the Connection String (defined in section 2.3.1) in the Db Connect Element in the Model Definitions.

On initialisation on the model, the function HourlyUpdateIndicator is triggered, as shown in Figure 10. This first uses dbExecute to clear each of the tables in the dataframe, one after another. A loop of dbWrite and Delay is used to write the TimeNow to the hourlyupdate table, after every hour is passed in Simio. This is used for Python to be triggered that it has to synchronise its dataframe with MySQL.

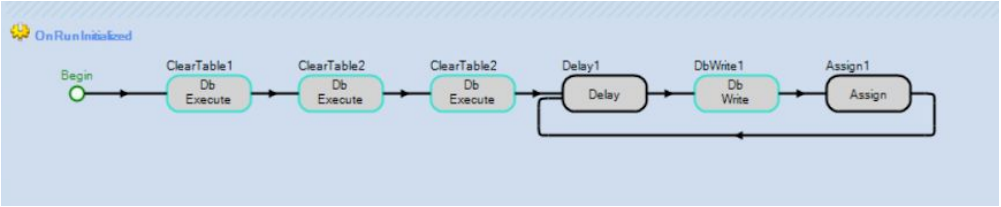

Figure 10: HourlyUpdateIndicator Add on Process

A second AddOnProcess named TrafficDestroy is triggered, whenever an entity enters a BridgeNode that is the beginning of a new segment on the road, see Figure 11. This process is used for regulating the traffic on the road segment, but is also used for data base writing. Each MyTraffic entity has a state variable “SegmentStartTime”. On entering this process the dBWrite is used to create a new row in segmentstarttime table, recording the RowID2, LRPNName, VehicleType, and (TimeNow - SegmentStartTime) to indicate the total travel time of that vehicle type on the previous segment. Assign is used to update RowID2 each time.

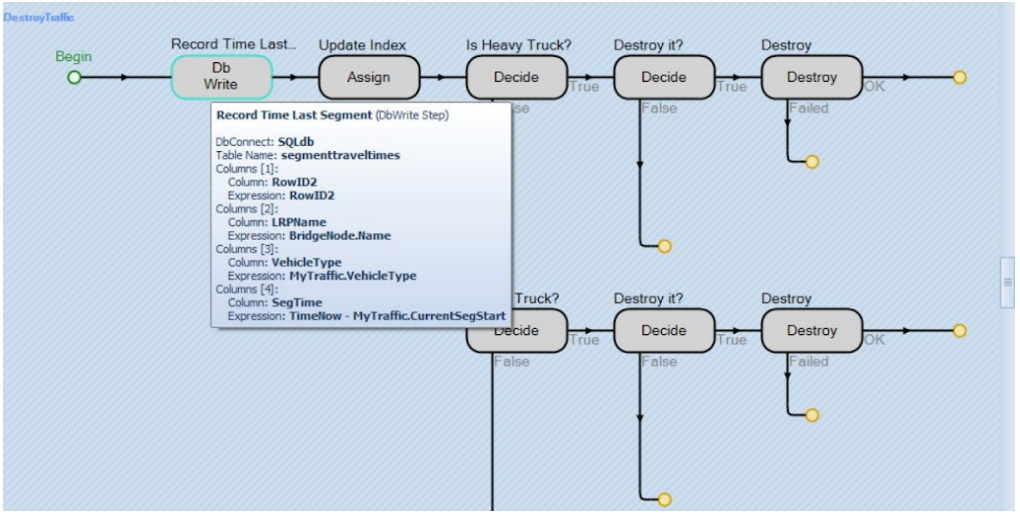

Figure 11: TrafficDestroy Add on Process

Finally, in the BridgeNode process BreakdownImportantBridges, rows are written to brokenbridges SQL table to keep track of which bridges are broken down, and at what time.

As shows in Figure 12, this process uses dbWrite after it has decided (by conditionality and probability) that a bridge is to be broken at that time. Assign is used to update the RowID3, so as to avoid overwriting the data table in the server.

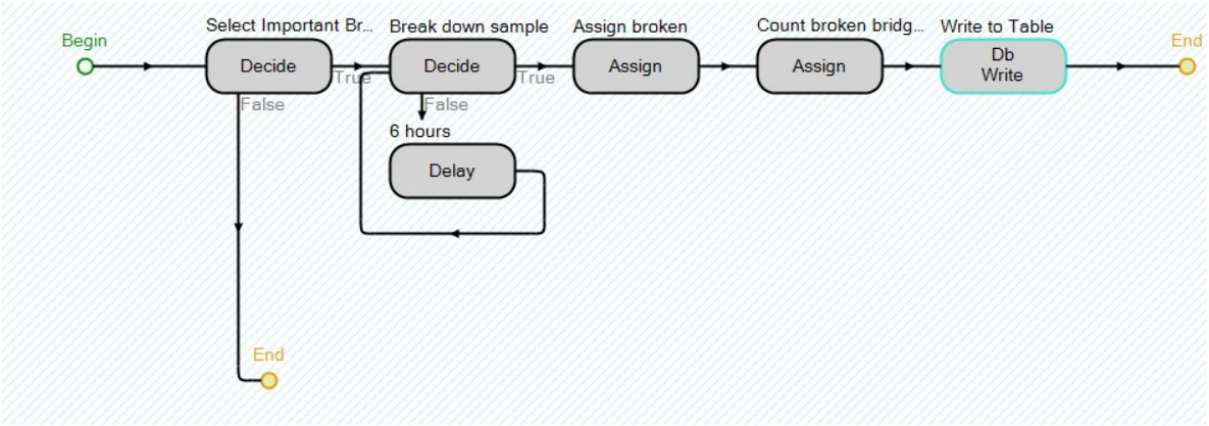

Figure 12: BreakdownImportantBridges Add on Process

3.5 Verification of Simio Model

Our model consist of several separate components that ultimately are being employed in order to model the traffic over time on the N1 road. Small errors in programming can easily result in large error in simulation results. Therefore it is important to do a proper verification of the model results.

3.5.1. Verification of Traffic Density on Road Segments

To verify the flow of traffic in the Simio model, some test runs were completed using our Simio model and SQL database table output. We inspected a model run of 48 hours for the N1-28 segment. According to the ADDT data, the model should produce the following traffic data, see table 4.

We inspected the values the model generated over 48 hours for two traffic types at this segments, namely for Heavy and Medium trucks. We conclude that for the Medium trucks, the amount of traffic created matches the ADDT data farrelly well. The numbers do not match precisely, this due to the rather long warm up period the model needs to evenly distribute all traffic throughout each road segment.

The heavy trucks behaviour is not satisfactory, their must be a bug somewhere. This behaviour was actually the result of an error in the add-on-process of creating heavy trucks. The token the process that executes the CreatMTs add-on-process was waiting. We altered the model by setting the token to NoWait in the execute step of the ControlTraffic add-on-process. This solved the behaviour for the Heavy Trucks.

	#Heavy Trucks over N1-28	**#Medium Trucks over N1-28**
Model results	18	1850
ADDT Data	120	1920

Table 4: Verification of traffic flows

3.5.2. Verification of Breaking Down Bridges

We also performed a verification of the number of bridges breaking down over time. We ran another simulation run over 48 hours. During the simulation run, a total number 10 bridges were broken down, see table 5, which indicates that the add-on-process that regulates the breaking down of bridges generates satisfactory model output.

	#Broken bridges
Model results	10
Results based on input	10

Table 5: Verification of broken bridges

4 Real-Time Visualizations

Based on the definitions developed previously, visualizations were developed to communicate the results of the delay time per segment, travel times and traffic density per road segment. The N1 road was analyzed because of its geographic and economic importance, and the developed tool could be scaled for analysis and visualization of every road in the Bangladesh road network with the right data and computation power.

The N1 is the most important road in Bangladesh and runs from Dhaka in the north to Teknaf in the south. The first part of the N1 road is especially important since it leads from the main capital Dhaka to the harbor in Chittagong and enables a lot of economic cargo to be transported.

4.1 Travel Time of Busiest Segments

For the first visualization, we show the 8 segments (this amount of segments was selected for illustrative purposes and can easily be adjusted) that are the busiest along the N1 road. We show per segment the total travel time per mode of transport. This results in the following stacked bar graph.

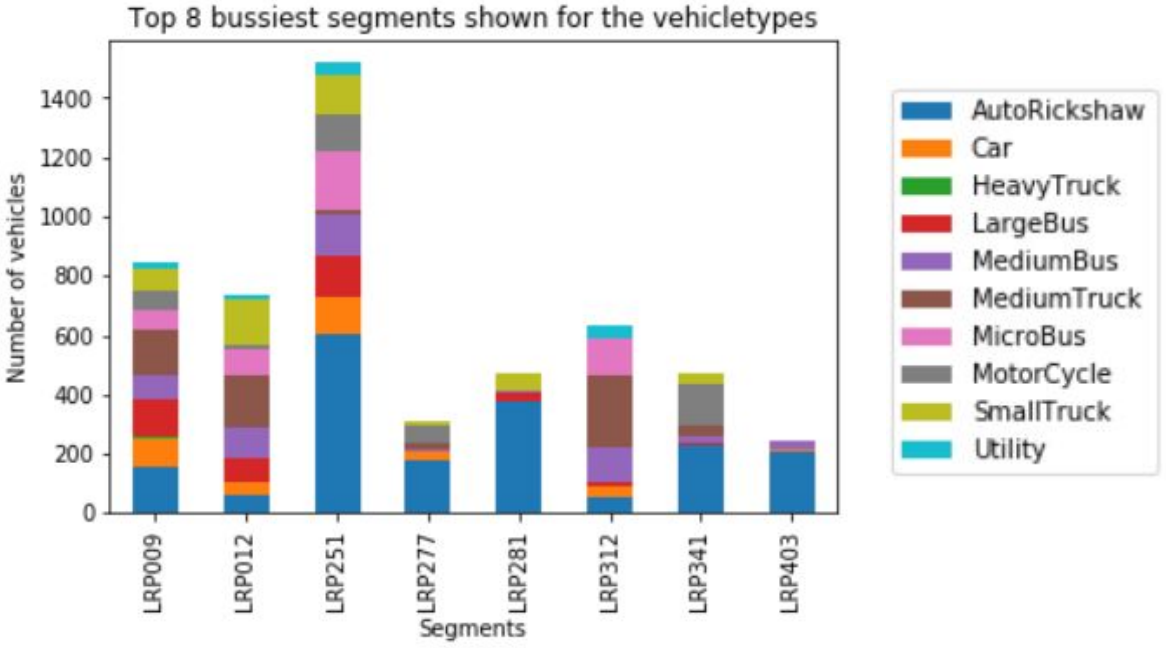

Figure 13: Top 8 busiest segments shown for all vehicle types

From figure 13 can be concluded that the segment that ends with LRP251 is the busiest segment and is mostly because of the AutoRickshaws.

4.2 Prevalence of Vehicle Type Per Segment

For the second visualization, we show how the different modes of transport are spread over the different road segments. For this visualisation only the busiest 8 segments are shown (this variable again can easily be changed). For each vehicle type, the graph displays on which segments it is most present.

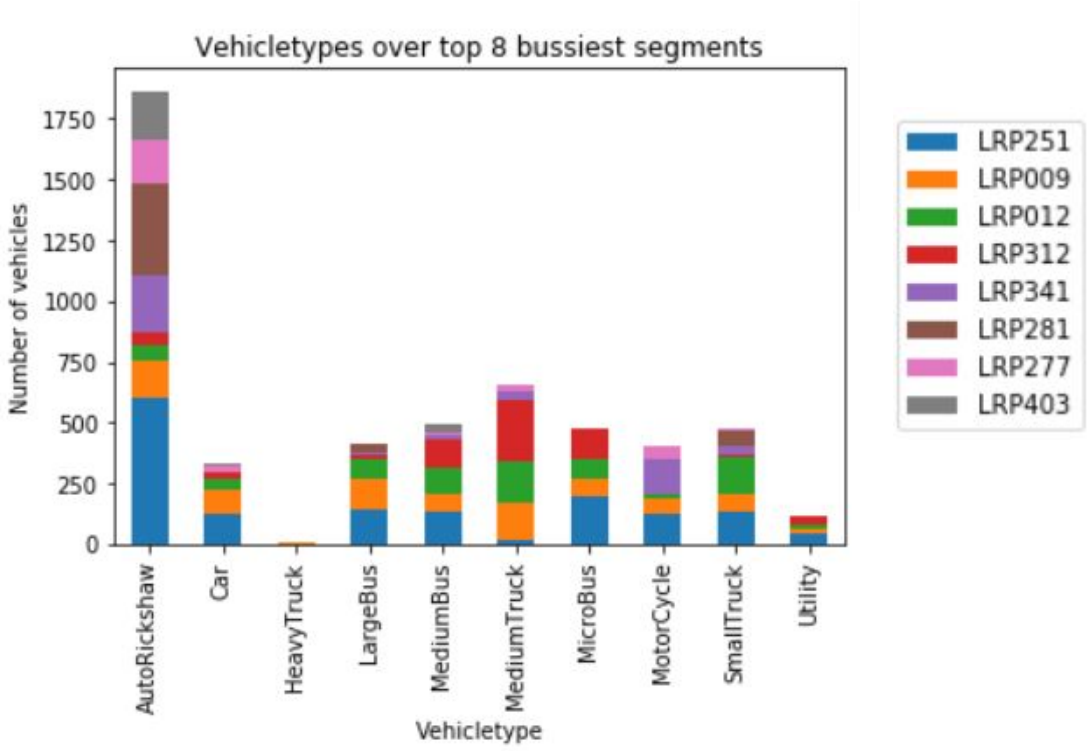

Figure 14: Vehicle types over top 8 busiest segments

From figure 14 we can conclude that there are more autorickshaws than any other form of transport and that this mode of transport is mostly present on segment ending with LPR251. This conclusions is in line with the conclusion from figure 13.

4.3 Average Travel Time

For the third visualization, the average travel time is displayed. This is divided into two categories, the average travel time over the full N1 and only the portion of the N1 from Dhaka to Chittagong. This average travel time is computed over all the data generated by the simulation model up until the moment of visualization. The travel time will therefore become more accurate over the course of the model run. This moving average is shown over time and displayed for both road categories (full N1 and Dhaka-Chittagong N1) in the graph below.

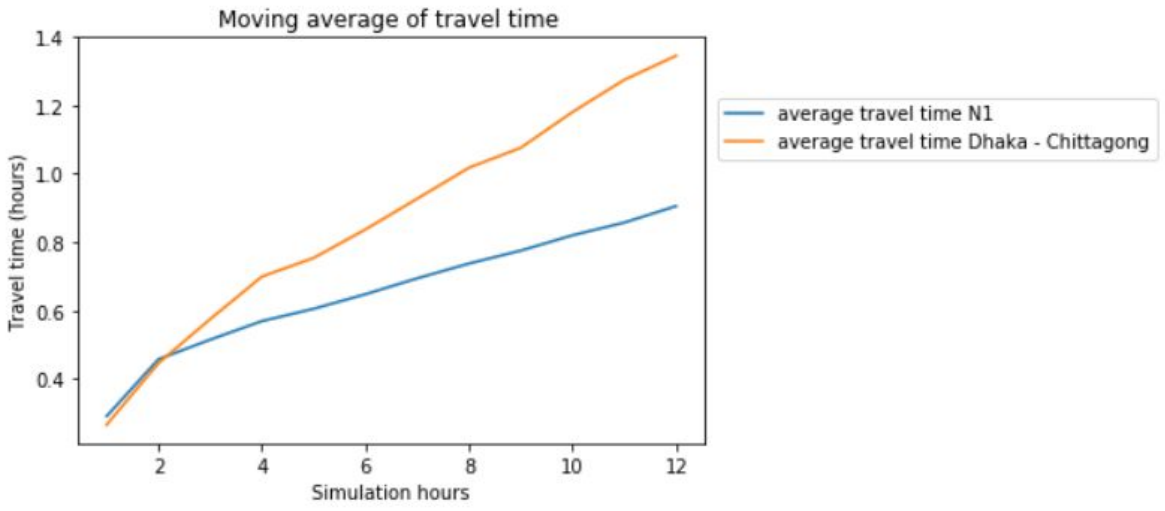

Figure 15: Moving average of travel time

Figure 15 shows that the difference in terms of average travel time between the full N1 and the part until chittagong differs. This is mainly because there is more transport on the first part of the road

4.4 Bridge Breakdown Location

The last visualization shows the bridges that are broken down overlaid on a map of Bangladesh. The green points represent the bridges that are still functioning and the red crosses represent the bridges that have broken down.

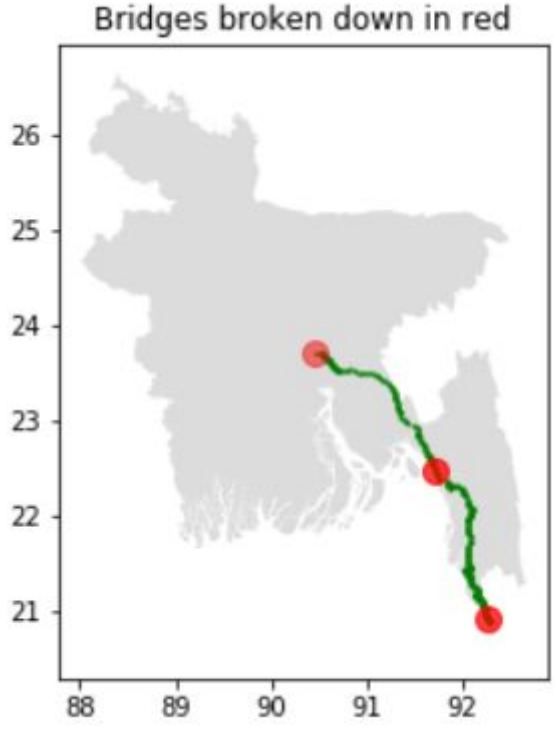

Figure 16: The location of the bridges that are broken down (in red)

5 Conclusion

5.1 Limitations

Some areas of the project were limited by data availability and time constraints. The main limitations are summarized in sections 5.1.1, 5.1.2, and 5.1.3 and the associated recommendations are described in section 5.2.

5.1.1 Simio

The Simio model was created, for the purposes of this assignment, to mimic data that had been collected by the government of Bangladesh. This data was context poor and did not offer insight into how traffic patterns varied over time, thus the assumption had to be made that the data was representative of the real life system, although it might not be. For a real client, the quality of the data that was driving the entire analysis was not specific or current enough to offer useful insight for investment decisions.

We were unable to run the full runtime of 72 hours due to computational limitations. For our final run, we only ran the model for 14 hours (we run the model for over an hour). Given that our model needs a rather long warm up period, the results are probably affected by this short run time. Because of the high number of entities in our model our model ran very slowly in combination in MySQL. In future work, we should therefore focus more on scaling the entities in order to lower the computational burden of the simulation tool.

5.1.2 MySQL

We did not achieve setting up remote access to a MySQL server that could be accessed by a computer (or multiple computers) running the visualization Python script or the Simio simulation, for example. As such, MySQL data dumps had to manually be shared across group members during the project, which would certainly also be a hassle for a client as well.

5.1.3 Python

The Python script that queried the MySQL database and created plots was implemented, but with some limitations and risks. Specifically, a while loop in combination with a semaphore table was used in the Python script to ensure that while Simio was writing data to the MySQL database, Python remained active in order to pull data from MySQL and update the plots accordingly. This was a risky method to implement as while loops can also cause trouble when they go wrong, especially as it would require a client to be able to debug the Python code in order to resolve the issue.

5.2 Recommendations

The limitations of this work offer space for actionable recommendations to improve the result. If these recommendations are implemented in future work they would greatly increase the usability, interoperability, and significance of the project results for the user.

5.2.1 Simio

In future work, improved data collection and increased data context would be used to create a Simio model of the N1 road that would reflect intra-daily and inter-daily traffic patterns. Better data would directly translate to better insights for the client.

5.2.2 MySQL

Setting up remote access to the MySQL database could significantly increase interoperability of the distributed modules designed and used in this assignment. This is important future work considering that a main benefit of implementing distributed modules (especially in the context of this assignment) is the ability then afforded to decentralizing the information creation, analysis, and presentation and the associated computing resources.

5.2.3 Python

In future work, a trigger from MySQL could be used instead of a while loop to update the data in Python and its visualizations. This would then result in a push instead of pull data update. The MySQL trigger, would push a signal and or the new data to Python, which would then update the visualization plots. If implemented in the future, this feature would also increase ease of use of the final result for the client.