Generating Artificial Data for Bus Travel Time Predictions

Leandro S. Ribeiro[1], **Thiago P. Faleiros**[1], **Maristela Holanda**[1]

[1]Computer Science Department – University of Brasilia (UNB)
Brasília, Distrito Federal, Brazil

leandro.santos.r@gmail.com,{thiagodepaulo,mholanda}@unb.br

Abstract. *This paper proposes a simulator capable of quickly generating a large amount of data that may be used to train bus travel time predictive algorithms in an urban transport network. To validate the proposal, a case study was carried out on a bus line in the city of Brasília/DF, Brazil. In the case study, the Simulator generated data for several scenarios that differ in distinct levels of variability and these data were used to evaluate the performance of a K-Nearest Neighbor predictor in each of the scenarios.*

1. Introduction

Urban residents rely on different modes of public transportation for their daily commute. Based on that, many works have been developed aiming to improve the efficiency and the accessibility of urban transportation services [Lima and Campos 2017, Monteiro et al. 2017]. In this context, accurate and real-time travel time information for buses has an important role because, among other reasons, it can help passengers better plan their trips and minimize waiting times.

Many factors such as weather conditions, day of week, time of day, and current traffic conditions may influence bus travel times. However, the exact nature of such relationships between travel times and predictor variables is usually not known and somehow these factors need to be incorporated into prediction algorithms either indirectly through binned analyses or through direct modeling [Kormáksson et al. 2014]. Moreover, travel times in urban areas are prone to high degrees of variability due to the presence of traffic lights, congestion, geometric conditions of roads and weather conditions [Reddy et al. 2016].

High variability conditions may affect the performance of the travel time predictor. For instance, [Reddy et al. 2016] proposes bus travel time predictions under high variability conditions and concludes that Kalman Filter [Sorenson 1966] and Support Vector Machines (SVM) [Platt 1999] are promising prediction techniques to solve high variability problems. Therefore, the travel time prediction method choice should account for the degree of environmental variability.

One way to analyze the degree to which variability impacts predicting bus travel times in an urban transport network is to collect data from the network in a variety of situations with high and low degrees of variability, then predict bus travel times for each of these situations using a predictor. Finally, predictor performance can be compared for each of these scenarios. However, to run this experiment with data from a complex transport network in a large urban center and different scenarios changing traffic parameters to check simulator performance, is unfeasible, since the costs involved in creating each of the different scenarios in real world network would be prohibitive. Therefore, one way to overcome this challenge would be to use traffic simulation tools. Wen et al. in [Wen 2018] highlight safety, convenience and low cost as advantages of using a simulation.

The main challenge of this work is to propose a simulator capable of quickly generating a large amount of data that may be used to train bus travel times predictive algorithms. In addition, the simulator must be able to generate data for several scenarios, so this data may be used to evaluate the performance of prediction algorithms for each scenario. The main requirements for the simulator include: the ability to provide geographic location and velocity of buses when requested, geolocation data available through a Representational State Transfer (REST) Application Programming Interface (API), allow simulation variability degree adjustments using parameters, simple modeling and integration with other systems through a geographic database.

In order to validate the proposed tool, a case study was carried out simulating the traffic of a bus line in the city of Brasília/DF, Brazil. In this case study, the bus lines were represented as graphs in which bus stops are represented by nodes, and the roads between bus stops are the edges. The data generated by the simulation were used to train and test a machine learning algorithm to predict bus travel times. The results showed that the degree of variability affects the performance of the predictor in terms of the calculated error.

The current paper is organized as follows: in Section 2 the related works are discussed; in Section 3 the architecture of the proposed tool and its components are described; in Section 4 a case study with its respective results are presented; and finally, in Section 5 there are the conclusions.

2. Related Works

Simulation models can be classified into two types: macroscopic and microscopic [Helbing et al. 2002]. The macroscopic traffic models are restricted to the description of the collective vehicle dynamics in terms of the spatial vehicle density $\rho(x,t)$ and the average velocity $V(x,t)$ as a function of the location x and time t. In contrast, the microscopic traffic models delineate the positions $x_{\mathrm{a}}(t)$ and velocities $v_{\mathrm{a}}(t)$ of all interacting vehicles as a function of time t.

The macroscopic models are addressed to large scale traffic, treating traffic as a liquid, frequently applying hydrodynamic flow theory to vehicle behavior, such as Cellular Automaton (CA), car following model and IDM/MOBIL as widespread models used within the traffic science community [Sommer et al. 2011]. All these approaches are of equal value in terms of mobility models.

In [Wen 2018], traffic simulations of vehicles in a connected environment, traveling through a commercial area, were carried out with the PARAMICS tool to test the collection of traffic data and the prediction of travel times, four types of models were constructed for the analysis of travel times: linear regression, multivariate adaptive regression spline, stepwise regression and elastic net. The results showed that the approaches had similar performance in terms of Root Mean Square Error (RMSE).

Similarly, in [Barceló et al. 2010] it is presented a comprehensive list of traffic simulation softwares describing their approaches to model building and implementation. The microscopic approach is represented by the softwares VISSIM, AVENUE, Paramics, Aimsun, MITSIM, SUMO and DRACULA. The macroscopic approach is represented by

METANET. Although most of these tools are complete in terms of functionality, a simpler model may be implemented to generate a large volume of data with less computational effort.

When there is a concern about the exact positions of simulated nodes, macroscopic and mesoscopic models cannot offer this level of detail, then only microscopic simulations are considered. Microscopic simulations model the behavior of single vehicles and interactions between them [Sommer et al. 2011]. However, there are different disadvantages, such as a high computational time and the requirement of detailed information, which might limit the applicability of a microscopic model to medium networks and those that do not operate in real time [Adacher and Tiriolo 2018].

Considering the trade-off between macroscopic and microscopic models, one of the challenges of this work is to propose a simulator that can provide the exact position of each simulated node, but with a good applicability in real-time applications and without the need for high computational time, merging characteristics of the macroscopic and microscopic models. Considering the high computational cost required in the general simulator softwares, and the lack of specific simulator to generate training data for bus travel time predictive algorithms, we propose a dedicated simulator able to model several external events that may occur in daily traffic and to generate data useful for experimental evaluation of predictive algorithms. The proposed simulator seeks to achieve these objectives by reducing the complexity of the model, and the results maintains the necessary coherence for the analysis of the performance of travel time predictors.

3. Simulator Architecture

The Simulator consists of four software components and a Geographic Database. The software components are: Time Controller, Line Simulator, Trip Simulator, and Location REST Service. The diagram with the components is shown in Figure 1. Each software component will be described in the following sections.

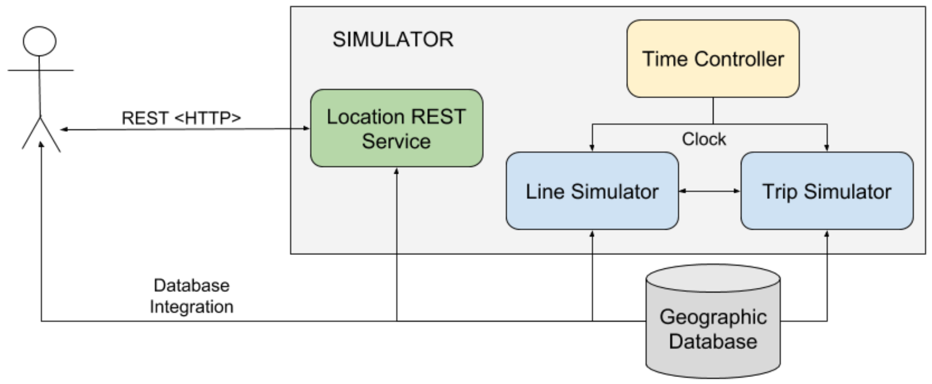

Figure 1. Simulator Architecture.

3.1. Line Simulator

The Line Simulator performs scheduled tasks according to a parameterized period. The main activities performed by this component are: updating edge status, updating neighboring edges influence, updating edges average velocities, updating node delays, and updating edges in geographic database. The responsibility of this component is to update attributes related to the behavior of roads and bus stops.

To simulate the occurrence of events, such as accidents or bad weather, each edge is classified as: normal, light event, moderate event, and severe event. The normal status represents a situation without events. In other words, it is not under the influence of external events that may negatively impact the traffic flow, whereas the other statuses represent the occurrence of events that impact the traffic flow with increasing degrees of severity from light to severe.

During the edge's status updating process, if an edge has normal status, a pseudo-random probabilistic event determines an increasing probability of severe, moderate and light events. This means the occurring probability of a light event is greater than the occurring probability of a severe event. If no event occurs, the edge retains normal status.

If an edge is under the effect of some event during the edge's status updating process, the event regression probability is evaluated. A severe event regresses to a moderate event, while a moderate event regresses to a light event and a light event regresses to the normal status. Again, the regression probability of a light event is greater than the regression probability of a moderate event, which is greater than the regression probability of a severe event. Figure 2 illustrates the edge status state machine.

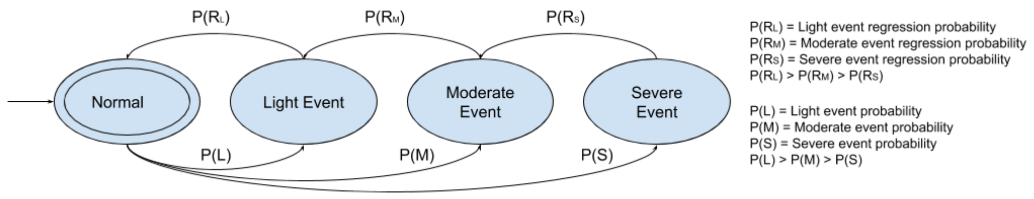
P(R_L)
P(R_M)
P(R_S)
Normal
Light Event
Moderate Event
Severe Event
P(L)
P(M)
P(S)
P(R_L) = Light event regression probability
P(R_M) = Moderate event regression probability
P(R_S) = Severe event regression probability
P(R_L) > P(R_M) > P(R_S)
P(L) = Light event probability
P(M) = Moderate event probability
P(S) = Severe event probability
P(L) > P(M) > P(S)

Figure 2. Edge Status State Machine.

To represent the influence of neighboring edges, four influence classifications were created: severe, moderate, light and absent. An edge is under severe influence when the edge immediately downstream has a severe event, on the other hand, an edge is under moderate influence when the edge that is after the edge immediately downstream has a severe event. Finally, an edge is under light influence when the edge immediately upstream has a severe event. If the edge does not fit into any of the above rules it will be under absent influence. Figure 3 illustrates the influence of an edge under severe event on its neighboring edges.

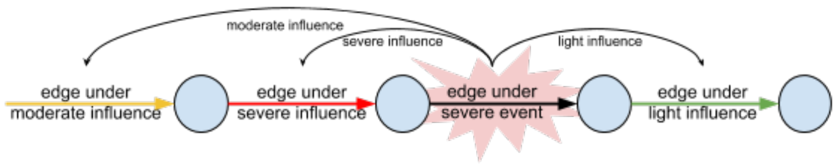
moderate influence
severe influence
light influence
edge under moderate influence
edge under severe influence
edge under severe event
edge under light influence

Figure 3. The Neighboring Edges' Influence.

After updating edge status and edge influence, edge velocity is updated. Each edge has an average velocity that is initially represented by the maximum velocity allowed in that edge. In the edge updating average velocity first step, the path maximum velocity is multiplied by a correction factor that varies according to the edge status. This factor is always a number less than or equal to one. The result velocity will always be less than or equal to the maximum edge velocity allowed. The higher the severity of the event, the lower the correction factor. This correction factor has a normal distribution so that one simulation parameter determines the mean and another determines the standard deviation.

In the second update step, the result calculated in the first step is multiplied by a peak time correction factor. This factor is not applied if the current time is outside the interval of the peak hour. Similar to the status correction factor, the peak time correction factor has a parameter that determines the Gaussian mean and a parameter that determines the Gaussian standard deviation, but in this case, the Gaussian mean is not a constant value. Rather, it is determined by a linear discontinuous function that decreases in the first half of the peak hour window and increases in the second half of this window, so that the velocity variation becomes a little smoother. The discontinuous linear function can be replaced by any other mathematical function to better model this phenomenon.

Finally, in the third update step, the result of the second step is multiplied to the influence correction factor – the greater the influence of the neighboring edge, the lower the influence correction factor and the lower the final average velocity. As with all other correction factors, this factor has a normal distribution parameterized by the mean and the standard deviation.

Node delays represent the amount of time buses spend at bus stops. These delays also have a normal distribution with mean and standard deviation parametrized. Presently, to simplify the process, the peak time window does not affect node delays. However, it is possible to apply a peak time correction factor to these delays, increasing the delay at peak hours.

The last activity of the Line Simulator is to update the Geographic Database with edge average velocities. This allows any graphical tool that can integrate with a geographic database to monitor edge status and edge average velocity at simulation time.

3.2. Trip Simulator

Similar to the Line Simulator, the Trip Simulator executes scheduled tasks according to a parameterized period. The main activities of this component are: updating bus position, updating bus velocity, and updating Geographic Database with bus positions. The responsibility of the Trip Simulator is to update the behavior attributes of the buses.

Each graph element behaves differently, whether it is an edge or a node. When a fraction of time is available to a bus that is on a node, this time is consumed while the bus stands on the bus stop. However, when the bus is on an edge and a fraction of time is available, it moves over the edge at a distance that is a function of its instantaneous velocity and the available time. The travel times spent by all the buses in each of the stops or crossing each of the edges are recorded in the Geographic Database at the exact moment that the vehicle leaves the respective graph element.

The instantaneous bus velocity is a function of the graph element average velocity. Graph nodes always have zero average velocity, while the average velocities of edges are calculated by the Line Simulator. The instantaneous velocity of the buses is the multiplication of the edge average velocity by a velocity oscillation factor. Consequently, two buses on the same edge do not necessarily have the same instantaneous velocity. The buses' instantaneous velocities have a normal distribution, with mean equals to edge average velocity and parameterizable standard deviation, which is the velocity oscillation factor.

Node delays are multiplied by a delay oscillation factor, so two buses do not necessarily remain the same time period in a certain bus stop. The delay oscillation factor also has a normal distribution with mean equals to the node average delay and parameterizable standard deviation – the delay oscillation factor.

Finally, the Geographic Database is updated with the positions of the buses, allowing other tools to monitor these positions at runtime through database integration.

3.3. Location REST Service

The Location REST Service allows retrieving information from the entire bus fleet through a Hypertext Transfer Protocol (HTTP) call. The information is made available through a JavaScript Object Notation (JSON) document and includes name, line, velocity, and location (latitude and longitude) of all buses.

This service, in addition to allowing system integration independent from the platform, the operational system, or programming language, it also makes possible to obtain bus fleet data in a real time.

3.4. Time Controller

The Time Controller is nothing more than a clock that controls the simulation time. All other system components use the Time Controller clock instead of the operational system clock. The Time Controller clock speed is set by the ”Time Multiplier” parameter. If the parameter is set to two, then the time passage in the simulation will be twice as fast as the real time passage. Therefore, this component allows the simulation to be executed both in real time and in an accelerated time, so that weeks of traffic simulation can be executed in a few hours.

3.5. Geographic Database

The Geographic Database, illustrated in Figure 4, stores information from nodes and edges, bus locations over time, and travel time information of all edges and nodes. In addition to maintaining the entire simulation history the database also works as an integration point between the Simulator and other systems.

The ”Location” table, in Figure 4, stores the buses' geographic location and velocity at a given time. The ”TravelTime” table records the time a bus has remained in a given graph element. The ”GraphElement” table has all graph nodes and edges including each element name, size, maximum speed allowed , and the bus line name. When the element is a node, the ”Graph Element” table has the geolocated point with the node location. However, when the element is an edge this table stores the set of geolocated points and lines that represent the edge trace and position.

3.6. Simulator Parameters

Adjusting the Simulator parameters, a traffic situation with high variability can be created: constant changes in bus speeds, occurrence of many external events and many other traffic

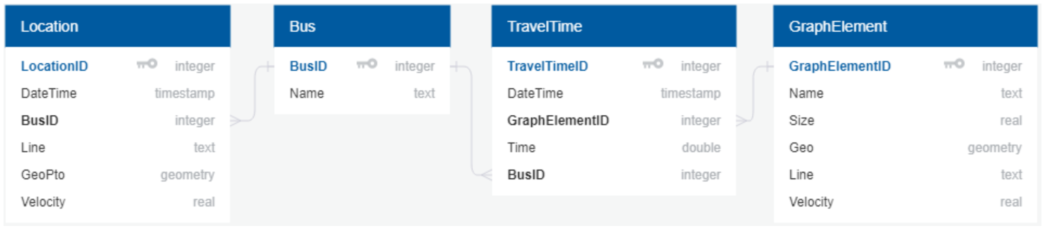

Figure 4. Geographic Database.

situations. Likewise, other adjustments can create a situation with low variability: with constant velocities and little or no occurrence of external events.

In addition, by adjusting the parameters it is possible to define the number of buses that will be running during the simulation, the windows of the peak hours, the Line Simulator and the Trip Simulator update periods, and the time multiplication factor. Table 1 lists all Simulator parameters.

Table 1. Simulator Parameters

Name	Description
fleetSize	Number of buses
severeEventProb	Severe event probability
moderateEventProb	Moderate event probability
lightEventProb	Light event probability
normalCorrectionFactor	Normal status correction factor
lightCorrectionFactor	Light event correction factor
moderateCorrectionFactor	Moderate event correction factor
severeCorrectionFactor	Severe event correction factor
correctionFactorSD	Correction factor standard deviation
severeEventEndProb	Severe event ending probability
moderateEventEndProb	Moderate event ending probability
lightEventEndProb	Light event ending probability
absentInfluence	Absent influence factor
lightInfluence	Light influence factor
moderateInfluence	Moderate influence factor
severeInfluence	Severe influence factor
influenceSD	Influence factor standard deviation
morningPeakTime	Morning peak time
afternoonPeakTime	Afternoon peak time
tripSimulatorUpdate	Trip Simulator update period
lineSimulatorUpdate	Line Simulator update period
timeMultiplier	Simulation time multiplier factor
delayOscillationFactor	Node delay oscillation factor
delayOscillationFactorSD	Standard deviation of the node delay oscillation factor
velocityOscillationFactor	Edge velocity oscillation factor
velocitOscillationFactorSD	Standard deviation of the edge velocity oscillation factor
peakTimeCorrectionFactor	Peak time correction factor
peakTimeCorrectionFactorSD	Standard deviation of the peak time correction factor

4. Case Study

A case study was carried out using simulated data to evaluate a travel time predictor performance using scenarios with different variability degrees. The simulations were performed on a bus line located in downtown Brasília / DF, Brazil.

Geolocation data from the Federal District urban transport network, on the right side of Figure 5, were obtained from a GeoServer [Contributors 2015] maintained by the Federal District Transit Department (DFTRANS). For the case study, only the "CIRCULAR-W3-SOUTH-NORTH-L2-NORTH-SOUTH" bus line was used, on the left side of Figure 5.

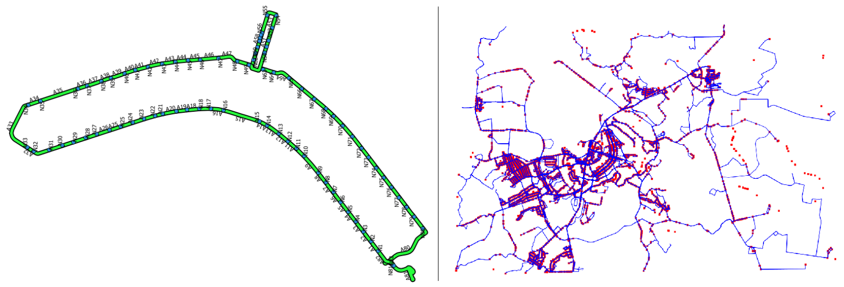

Figure 5. Federal District Urban Transport Network.

The bus line was transformed into a graph with 82 nodes named after N1 to N82 and 82 edges named after A1 to A82. The bus stops are represented by nodes, while the paths between bus stops are the graph edges. Travel time predictions take two nodes into account: the origin node (O) and the destination node (D), and the single path composed of one or more edges (A), which connects these two nodes, as shown in Figure 6.

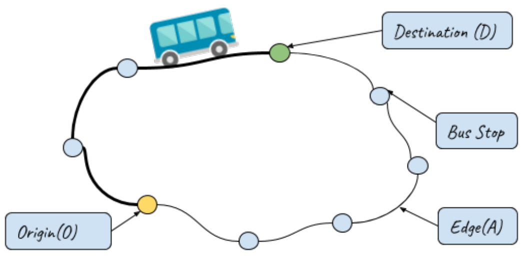

Figure 6. Bus Line Graph.

To create different levels of variability, simulator and predictor parameters were changed. The parameters were divided into 5 sets C_i, C_j, C_k, C_l, and C_m. Each of these sets has 3 different configurations. For instance, $C_i(0)$ is the parameter set C_i in configuration 0 while $C_m(2)$ is the parameter set C_m in configuration 2. The parameter sets C_i, C_j, C_k, and C_i are simulator parameters and are listed in Table 1 with their respective descriptions. The parameter set C_m has a single predictor parameter, which determines the quantity of previous travel times used to train the prediction algorithm. Since there are 4 simulator parameter sets – 1 predictor parameter set, and each one with 3 possible configurations – the total amount of simulated scenarios is 81, and the total amount of predicted scenarios is 243. Table 2 shows the 5 parameter sets and their values for each configuration.

During the experiments, the simulator ran for 3 hours for each of the scenarios described, with a fleet of 82 buses and a time multiplication factor of 60. Then, for 3 hours of simulation a little more than a week of traffic data was generated.

The prediction of travel times was performed by a K-Nearest Neighbors classifier (KNN) [Aha et al. 1991]. This classifier was chosen due to its speed and the simplicity of

Table 2. Simulation Scenarios

	Ci		
Parameter	**Ci(0)**	**Ci(1)**	**Ci(2)**
severeEventProb	0,0000	0,0005	0,0010
moderateEventProb	0,0000	0,0010	0,0020
lightEventProb	0,0000	0,0020	0,0040
	Cj		
Parameter	**Cj(0)**	**Cj(1)**	**Cj(2)**
lightCorrectionFactor	0,90	0,80	0,70
moderateCorrectionFactor	0,75	0,65	0,55
severeCorrectionFactor	0,60	0,50	0,40
peakTimeCorrectionFactor	0,80	0,70	0,60
	Ck		
Parameter	**Ck(0)**	**Ck(1)**	**Ck(2)**
lightInfluence	1,00	0,90	0,80
moderateInfluence	1,00	0,80	0,70
severeInfluence	1,00	0,70	0,60
	Cl		
Parameter	**Cl(0)**	**Cl(1)**	**Cl(2)**
delayOscillationFactorSD	0,01	0,05	0,10
velocitOscillationFactorSD	0,01	0,05	0,10
	Cm		
Parameter	**Cm(0)**	**Cm(1)**	**Cm(2)**
qtdPreviousTrips	6	4	2

its implementation. This approach is one of the simplest and oldest methods used for pattern classification. It often yields efficient performance and, in certain cases, its accuracy is greater than state-of the-art predictors [Hassanat et al. 2014, Hamamoto et al. 1997]. The KNN predicts the travel time of a test example using the average past travel time among its k-nearest (most similar) neighbors in the training set.

The data used to train and test the models were: the current travel time, the period of the day represented by a real number contained in the interval $[0; 24)$, the day of the week, the average bus speed, and η previous travel times where η is determined by the predictor parameter "qtdPreviousTrips".

In the KNN process, the number of neighbors k needs to be determined. To determine the parameter k, an experiment was carried out creating prediction models for a random sample of 8 edges extracted from the real traffic network. The set of sampled edges are labeled as $\{A_1, A_9, A_{33}, A_{35}, A_{51}, A_{58}, A_{80}, A_{81}\}$, where A_i is the edge in the geographic database indexed by value i. The value of k was ranged from 1 to 52, and for each value of k the model was created and evaluated by calculating the absolute mean error. Figure 7 presents the absolute mean errors calculated during the evaluation of the prediction model as a function of k for the 8 sampled edges. The curves initially have a decreasing behavior, and around $k = 4$ there is an inflection point and the curves starts to increase. Due to this behavior the value 4 was chosen for parameter k.

After estimating the value of k, KNN prediction models were created for all edges and for many parameter scenarios. The simulation data were divided into 70% for training and 30% for test, and for each test example, the value of Mean Absolute Error (MAE) was calculated, corresponding the absolute difference between the predicted travel time and

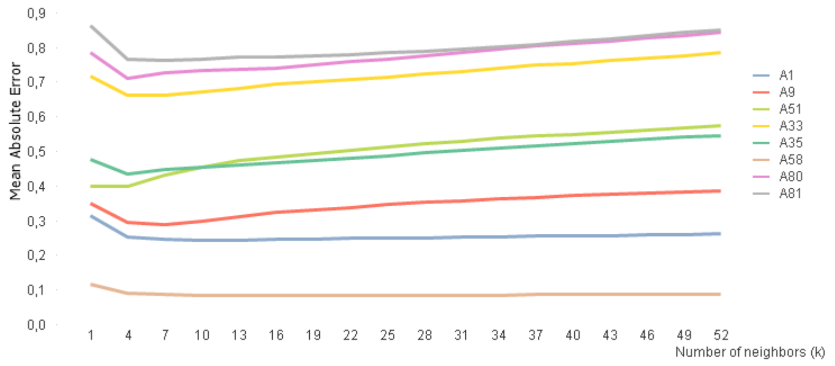

Figure 7. Mean Absolute Errors.

the real travel time.

The results predicted by KNN from generated data with several scenarios are summarized in Figure 8. Figure 8 represents a plot matrix, in which each plot shows travel time predictor mean absolute errors for each of the 243 prediction scenarios. Columns of the plot matrix represent subset of edges A_1, A_{33}, and A_{80}, while lines represent configuration groups C_i, C_j, C_k, C_l, and C_m. Numbers 0, 1 and 2 in plot matrix legends are related to simulation scenarios presented in Table 2. Not all predicted travel time results are presented here due to space limitations to plot all parameter combinations for each edge. The complete results and data can be found in `https://github.com/curupiras/results.git`. Nevertheless, the subsample presented in Figure 8 represents the pattern found in all edges.

In Figure 8, the plot matrix in line C_i shows that the greater the probability of the events, the greater the predictor error. Similarly, it can be observed in line C_j that the smaller the correction factors, the more the events and the peak time impact on bus speeds and the greater the predictor error.

Similarly, line C_k represents the impact of influence between neighboring edges on the predictor error and line C_l represents the impact of delay and velocity oscillation factors on the predictor performance. At some moments, a lower influence between neighboring edges and smaller oscillation factors end up increasing the predictor error, however this is not the rule. In most cases, the greater the oscillation factors, and the greater the influence of the neighboring edge, the lower the predictor performance, thus, the greater the error.

Finally, in line C_m, it is worthy to note that the quantity of previous travel times used in the predictor algorithm training has very little impact on its performance. However, it is possible to realize that when a greater quantity of previous travel times is used to train the model, there is a slight improvement in the predictor performance.

5. Conclusion

In this work, a simulator capable of quickly generating a large amount of data for travel time prediction algorithms was presented. The implementation was able to provide the geographic position and speed of each simulated bus, upon request, either using REST API calls or geographic database integration.

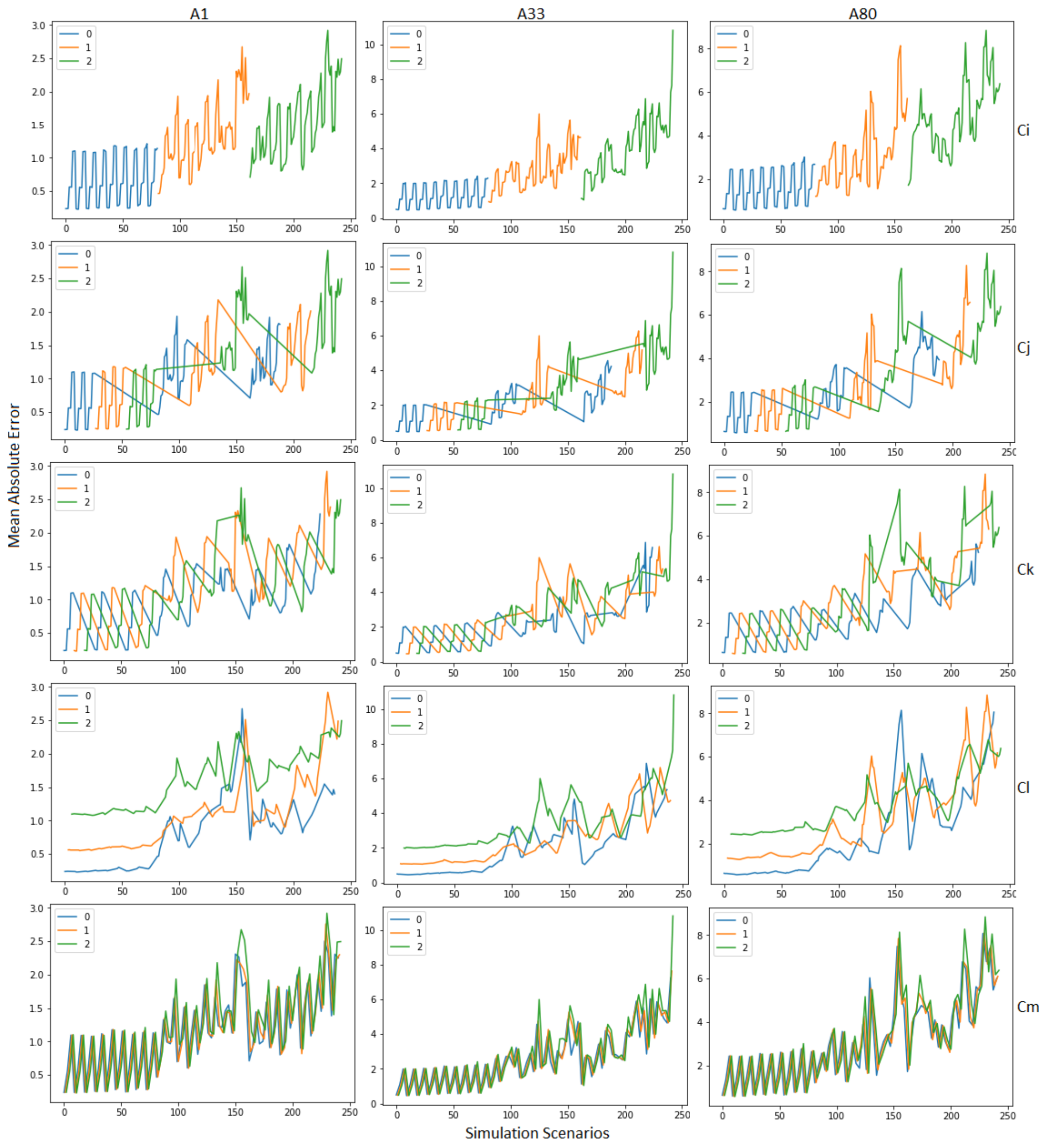

Figure 8. Predictor Performance.

Furthermore, to validate the implementation, a case study was carried out on a bus line in the city of Brasília. Over 20 months of traffic data with different levels of variability were generated in approximately 10 days, proving the Simulator's ability to generate a large quantity of data in a short period of time for several different scenarios. The simulated data were used to train and test a KNN bus travel times predictor and the predictor algorithm performance was evaluated in terms of mean absolute error. The results showed that, in general, the higher the degree of variability of the traffic environment, the lower the performance of the prediction algorithm.

Future work includes: comparing simulated data with real traffic data; testing the simulator with other bus lines; the use of other prediction algorithms and the analysis of their performance; refinements of the model to simulate traffic in multiple lanes; and to add the possibility of multiple paths choice.

References

Adacher, L. and Tiriolo, M. (2018). A macroscopic model with the advantages of microscopic model: A review of cell transmission model's extensions for urban traffic networks. *Simulation Modelling Practice and Theory*.

Aha, D. W., Kibler, D., and Albert, M. K. (1991). Instance-based learning algorithms. *Machine learning*, 6(1):37–66.

Barceló, J. et al. (2010). *Fundamentals of traffic simulation*, volume 145. Springer.

Contributors, G. (2015). Geoserver–open source server for sharing geospatial data.

Hamamoto, Y., Uchimura, S., and Tomita, S. (1997). A bootstrap technique for nearest neighbor classifier design. 19:73 – 79.

Hassanat, A. B., Abbadi, M. A., Altarawneh, G. A., and Alhasanat, A. A. (2014). Solving the problem of the k parameter in the knn classifier using an ensemble learning approach. *CoRR*, abs/1409.0919.

Helbing, D., Hennecke, A., Shvetsov, V., and Treiber, M. (2002). Micro-and macro-simulation of freeway traffic. *Mathematical and computer modelling*, 35(5-6):517–547.

Kormáksson, M., Barbosa, L., Vieira, M. R., and Zadrozny, B. (2014). Bus travel time predictions using additive models. In *Data Mining (ICDM), 2014 IEEE International Conference on*, pages 875–880. IEEE.

Lima, A. M. and Campos, J. (2017). How reliable is the traffic information gathered from web map services? In *GEOINFO*, pages 234–245.

Monteiro, C. M., Martins, F. V. C., and Junior, C. A. D. (2017). Optimization of new pick-up and drop-off points for public transportation. pages 222–233.

Platt, J. C. (1999). 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods*, pages 185–208.

Reddy, K. K., Kumar, B. A., and Vanajakshi, L. (2016). Bus travel time prediction under high variability conditions. *Current Science (00113891)*, 111(4).

Sommer, C., German, R., and Dressler, F. (2011). Bidirectionally coupled network and road traffic simulation for improved ivc analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15.

Sorenson, H. W. (1966). Kalman filtering techniques. In *Advances in control systems*, volume 3, pages 219–292. Elsevier.

Wen, X. (2018). A work zone simulation model for travel time prediction in a connected vehicle environment. *arXiv preprint arXiv:1801.07579*.