Modeling Traffic Flow With Microscopic Discrete Event Simulation

by

Casey Bowman

(Under the Direction of John A. Miller)

Abstract

Everyday billions of people around the world face the task of driving their vehicles in the traffic of their region. For many this entails entering very heavy traffic flows centered around large cities, with long commute times, on their way to work. For others, the issue is that they need to drive to a special event, or are just driving through the city on the way to another destination. Whatever the reason, drivers have a strong desire to know what the general traffic flow is along the route they plan to use.

Major cities employ traffic engineers to deal with the problem of managing the large traffic flows for which they are responsible. From routine highway and road maintenance, to redesigning existing interchanges, to constructing completely new throughways, city planners face the challenge of meeting the population's demand for efficient road networks.

For both sets of circumstances above, the desire for tools to make traffic-related decision-making easier is quite substantial. Microscopic traffic simulation has a lot to offer for modeling and forecasting traffic flows. These simulations are not only models of the overall flow of vehicles, but model the detailed interactions of the cars themselves, allowing for a depth of analysis not possible with other modeling techniques. Indeed, microscopic traffic simulation can offer prescriptive solutions to traffic problems, where, for example, city planners can try out different solutions for traffic design, without having to actually construct anything.

In order to build an effective and accurate traffic simulation model, there are many tasks that must be completed. The specific data for an area must be analyzed and used to build a realistic arrival model. A car-following model must be chosen, so that the vehicles in the simulation behave in a realistic manner.

Finally, the various parameters of these models must be fine-tuned with a calibration technique so that the models are as accurate (or as efficient) as possible. This work analyzes the arrival problem, chooses two well-known car-following models, and applies several calibration methodologies in an effort to identify the best means by which to build the traffic simulation model.

INDEX WORDS: [Microscopic Traffic Simulation, Arrival Models, Car-Following Models, Calibration, Traffic Forecasting, Time Series Analysis]

Modeling Traffic Flow With Microscopic Discrete Event Simulation

by

Casey Bowman

B.S., University of Georgia, 2003
B.S., University of Georgia, 2003
M.A., University of Georgia, 2005

A Dissertation Submitted to the Graduate Faculty of the
University of Georgia in Partial Fulfillment of the Requirements for the
Degree

Doctor of Philosophy

Athens, Georgia

2022

Modeling Traffic Flow With Microscopic Discrete Event Simulation

by

Casey Bowman

Major Professor: John A. Miller

Committee: Maria Hybinette
Ping Ma

Electronic Version Approved:

Ron Walcott
Vice Provost for Graduate Education and Dean of the Graduate School
The University of Georgia
December 2022

Dedication

to Keysa and Cass,

my parents, Donna and Danny,

and my brother, Eric,

thank you for your love and support through the years

Acknowledgments

First, I would like to thank my family who have stood by me for the very long time it has taken me to reach this point. Through thick and thin they have been there for me, and it truly means the world to me.

I would also like to thank my major professor, Dr. John A. Miller, for always supporting me, and helping me believe that I truly could achieve this goal.

I thank my committee members, Dr. Hybinette and Dr. Ma, and before his retirement, Dr. Potter. I truly appreciate the work they have done to help me along this path.

I would like to specifically thank Hao Peng, who was co-author for the paper that represents Chapter 3 of this manuscript. Our collaboration was a highlight of my time in this program.

There is a long list of friends and colleagues who have been there for me, in various ways, over the years, and helped make this possible: Dave, Mo, Ted, Ed, Matt, Lisa, Michael, Mustafa, Yulong, and Tom have been so helpful to me over the years.

Sometimes friendship is all that's needed to help someone keep going.

Contents

List of Figures

List of Tables

Chapter 1

Overview of Dissertation

1.1 Purpose of Traffic Modeling and Simulation

Traffic is a ubiquitous element of modern life in most parts of the world. A large number of vehicles on the road can make car trips difficult and even dangerous. Quite often a traveler will seek some sort of estimate about the condition of the road network ahead of time. This requires the endeavor of traffic modeling, of which traffic simulation is a popular technique.

Traffic flow and speed forecasting are of particular interest to a great number of normal drivers because most people do not like to get stuck in traffic jams. Traffic simulation models can be used to accurately predict traffic flow in the future, and drivers can use this information to decide what their departure time should be to minimize their chances for a miserable driving experience. They can also help drivers choose the best route to arrive at their destination either the fastest or with the least amount of stress.

Other forms of traffic models, such as machine learning models, can be used in similar ways as those mentioned above, to help drivers with their trip planning. However, traffic simulation models can also be used to address crucial what-if questions that city planners and engineers need to answer to design safe and efficient road networks. Physical changes to roads, such as lane additions, or refashioning a stop-sign-based intersection as a roundabout, are possible to do in a traffic simulation but would be much more difficult with a machine-learning technique. And while other techniques might be used to discover that a particular spot in a road is hazardous, a traffic simulation could be used to analyze *why* it is hazardous.

Much has already been achieved by the traffic modeling and traffic simulation communities. From the earliest days of computers, there have been researchers using them to study traffic flow, with a very large amount of work

beginning in the 1950s (Pipes, 1953) (Wilkinson, 1956) (Helly, 1959) (Gerlough, 1956). Early traffic models were mostly macroscopic in nature, meaning that only high-level modeling of the overall traffic flow was done, usually using formulations from fluid or gas dynamics.

Traffic simulation systems have been used for decades to improve traffic flow, inform road designs, and help travelers choose routes and departure times. Novel types of interchanges such as Diverging Diamond Interchanges and Michigan Lefts have been assessed through traffic simulation. In recent years software tools such as smartphone map apps have used traffic models to predict traffic congestion and travel times so that drivers have a better sense of what to expect on the road.

The work done in this dissertation has accomplished several goals. First, the creation of a microscopic discrete event traffic simulation system within the ScalaTion environment that can

(a) handle realistic vehicle arrivals,

(b) apply multiple car-following models to the movements of vehicles in the system,

(c) model limited-access highways

(d) model suburban intersections with traffic lights.

Second, explore the problem of arrival modeling in more detail, and compare and contrast forecasting methods for use with the arrival model. Specifically, the work explores how using the latest available data can drastically improve the arrival model results, which makes the simulation model much more accurate. A justification for the use of Nonhomogeneous Poisson processes for generating random arrival times is also provided, as well as a derivation of the formulas for the technique.

Third, the exploration of many different optimization procedures in the context of microscopic traffic simulation both in the process of optimizing traffic light timings and in calibrating a traffic flow model with real-world traffic data. Various types of optimization algorithms are compared and contrasted in the context of traffic flow modeling and are compared in terms of calibration accuracy, optimization improvement across epochs, and algorithm efficiency (in execution time). It is shown that deciding which optimization algorithm to choose from these comparisons is not straightforward and that the answer can change depending on the criteria being used.

Fourth, a detailed comparison of two car-following models, Gipps' Model, and the Intelligent Driver Model, using the system constructed in ScalaTion, is given. These models are compared through their use in calibrating the traffic system with real-world traffic data. Both similarities in performance and a few key differences are presented.

1.2 Traffic Flow Modeling

Suburban roads are largely dependent on traffic lights to organize and control traffic. A source of great frustration for many drivers is the perceived lack of quality in traffic light timings, meaning how long traffic lights are green for the various traffic directions, and how long they are red. The work in Chapter 2 is an effort at solving some of the problems associated with optimizing traffic light timings using a microscopic discrete event traffic simulation. Such a system needs accurate arrival modeling, a model of traffic flow through the system, and for specifically studying intersections, a model of turning behavior. Optimizing the timings of the traffic light requires the testing and comparison of various optimization algorithms.

A traffic simulation system, built in ScalaTion (Miller et al., 2010), and the required underlying structure, was created to facilitate networks of suburban intersections. The simulation system required the construction of several new features such as traffic lights, car-following models, a vehicle arrival process, and a procedure for modeling the turning behavior of vehicles.

The construction of the **Gate** class is discussed, which specifically allows for the modeling of traffic lights. The details of sources, sinks, roads, and vehicles are also discussed in the context of the simulation structure and the overall computational system ScalaTion.

The details of the vehicle arrival process and its basis in nonhomogeneous Poisson processes (L. M. Leemis, 1991) (L. Leemis, 2003) are given. Standard Poisson processes have been shown to have flaws in the arrival modeling of vehicular traffic (Q. Meng & Khoo, 2009). Traffic flow is extremely volatile and changes drastically during the course of a single day. Nonhomogeneous Poisson processes are needed to model arrivals with multiple busy periods, which is definitely how most urban and suburban traffic is structured.

Finally, with the need for realistically modeling how many vehicles turn (left or right), or go straight, at an intersection, a random variable, and turning algorithm are shown in detail. The construction of a probability distribution based on real traffic data is shown and how it is used to assign turning behavior to the vehicles in the system.

An effort was also made to optimize traffic light timings at a series of suburban intersections, as well as a discussion on what types of optimization algorithms are suitable for traffic simulation is presented. Gradient-based techniques, direct-search algorithms, and metamodel approaches are all discussed and compared.

1.3 Microscopic Simulation

There has been a large amount of effort made in the past with regard to traffic modeling. Much of this work has focused on the field of traffic simulation, though there have also been many efforts made in other areas such as machine learning and time series analysis. Chapter 3 presents much of the background of various approaches to traffic modeling and forecasting, and how these models are formulated.

First, much of the background on different approaches to traffic simulation is presented, with information on the granularity of models presented. An overview of high-level *macroscopic* traffic models is presented, as well as intermediate-level *mesoscopic* traffic models, and low-level *microscopic* traffic models. These three paradigms represent different levels of detail incorporated into the models. A genealogy of traffic flow models is given, which shows much of how the field has progressed, from the early 1950s through to the 2000s.

Second, some approaches in traffic forecasting from outside of the field of simulation are discussed, as a point of contrast to the work done in simulation. Methods such as time series analysis, machine learning, and statistical regression are discussed, as well as a rundown of some examples of traffic forecasting software that uses these methods.

Finally, some directions for future work are presented, including challenges in building discrete event simulation systems for traffic forecasting, the calibration and validation of such models, and the scaling of these models to facilitate forecasting in large traffic networks. Additionally, a discussion of some of the elements of Intelligent Traffic Systems (ITS) is presented, such as autonomous vehicles and intelligent traffic lights.

1.4 Arrival Modeling

A crucial point to understand in traffic modeling is that drivers, and therefore traffic, are different all over the world. To be useful a traffic model must be calibrated with data specific to the physical road network of interest. Until

this process is completed there is no ability to trust that a traffic model will be reliable.

1.4.1 Vehicle Arrivals

The vehicle arrival process is of particular interest since vehicle arrivals are so crucial to a realistic model of traffic flow. It does not matter how accurate a car-following model is, or any other traffic flow model for that matter, if the number of vehicles in the system is unrealistic. Vehicles do not arrive to traffic networks with any type of simple mathematical formulation. A probability distribution that can generate realistic vehicle arrivals is essential for any traffic simulation system.

This section details two aspects of building an arrival process. First, the numerical formulation used to create the probability distribution upon which the arrival model is based is presented. In this project, the distribution of arrival times in various periods of the day is assumed to be exponential, which means the number of vehicles arriving during those time periods is a Poisson counting process. Different time periods likely have different arrival rates, however, so the counting process is a Nonhomogeneous Poisson process (NHPP), and the technique used for recovering the arrival times from the counts is presented in detail.

The other aspect of the arrival process is to build techniques for creating those forecast counts. There are many ways to forecast time series, and for this project, five methods for how traffic data can be used to create the arrival process are shown. Two methods use only data from prior days and weeks to compute the approximations on which the probability distribution is built, and are therefore considered *offline* techniques. One of these methods is a simple average of previous days' data, while the other technique builds a seasonal autoregressive integrated moving average (SARIMA) model to forecast the new vehicle counts. The other three methods assume the availability of data from earlier in the day of forecasting focus, and so are considered *online* forecasting methods. Two of the methods calculate basic rates of change, one based on ratios, and one based on differences, using prior data, and apply this to the new data to improve the accuracy of the arrival model. The third method builds a SARIMA model using the data, including the data from earlier in the same day and forecasts the vehicle counts.

These forecasting methods, and the resulting NHPPs, are defined and compared, and an analysis of which techniques perform best is given.

1.5 Car-Following Models

A microscopic traffic simulation models individual vehicles, which then necessitates the modeling of vehicle *interactions*. Car-following models give a mathematical formulation to the dynamics of how vehicles react to the movement of vehicles in front of them.

Some intuition for how car-following models work is to consider the act of one vehicle approaching a downstream vehicle, and understanding that the velocity of the approaching vehicle must be adjusted as the distance between the two vehicles shrinks. Consider the simple linear relationship between velocity and spacing headway, which is defined as the distance between the front bumpers of the two cars, in Equation 1.1 (D. Meng et al., 2021):

$$v_n(t + \tau_n) = \frac{\Delta x_n(t) - \delta_n}{\tau_n} \tag{1.1}$$

where $\Delta x_n(t)$ is the spacing headway of the nth vehicle, τ_n is the reaction time, $v_n(t)$ is the velocity, and δ_n is the safe following distance. In this simplified model the velocity of the following vehicle decreases as the safe spacing between the vehicles decreases, factoring in the reaction time of the following driver.

Two specific car-following models, Gipps' Model (Gipps, 1981), and the Intelligent Driver Model (Treiber et al., 2000), described generally below, are the models chosen in this paper.

- Gipps' Model computes a new velocity for each vehicle based on several current parameters of the vehicle's movement. This velocity can then be used to calculate the vehicle's next position value along the road.
- The IDM computes a new acceleration for each vehicle based on current vehicle parameters, which then leads to the calculation of the vehicle's new velocity, and then the vehicle's new position.

1.6 Calibration

A traffic simulation model is essentially a multi-parameter estimation function that for a particular time window predicts what the traffic will be like along a specific road network. To have confidence in the accuracy of the estimates from the model it must be calibrated, that is a subset of real data must be used to try to figure out what are the best parameter values for the road network under consideration. This is achieved by using some form of optimization procedure, of which there are many to choose from. Only a subset of the data can be used

for calibration because the model would typically also need to be validated with the data not used for calibration.

Many optimization algorithms exist but specific ones were chosen for this work. **Genetic Algorithms** are a global search optimization strategy based on the theory of evolution and natural selection. Direct-search methods are local search algorithms that are gradient-free, and therefore applicable to problem spaces that are not necessarily continuous. The **Nelder-Mead Simplex Algorithm** is a popular choice from this subset of optimization algorithms. Gradients are difficult or time-consuming to produce for optimizations involving simulations, but gradient-approximation schemes can work much better. A popular choice in this paradigm is the **Simultaneous Perturbation Stochastic Approximation Algorithm**, or SPSA.

Assessing the results of a calibration procedure can be done in multiple ways. It is often the case that *calibration accuracy* is of the utmost importance, and the algorithm that produces the closest approximations is the desired choice. However, there are certainly real-world contexts where the *efficiency* of the calibration procedure is just as, if not more, important than accuracy. The optimization procedures can also be compared by looking at their rates of improvement as the optimization process executes.

Chapter 2

Modeling Traffic Flow Using Simulation and Big Data Analytics

2.1 Introduction

All over the world, populations are rising, and the need to create safe and efficient road systems becomes more and more prescient. Developing nations have long lagged behind industrialized nations in the number of vehicles in use on a day-to-day basis, but this is changing, as many of these nations begin to truly enter the ranks of the first world (Cervero, 2013). In the United States, where large tech corporations are moving toward self-driving vehicles (D. Levin, 2015), (Fisher, 2013), the need for coordinated and adaptive road systems, i.e. Intelligent Transportation Systems (ITS) is essential (Sussman, 2008), (Ran et al., 2012). However, the sheer volume of vehicles on our roads forces the need for Big Data techniques in acquiring, processing, and utilizing traffic data for the purpose of traffic system optimization (Vlahogianni, 2015).

Much work is needed to model the behavior of vehicles in such a system, and we discuss our approach in detail. A method for modeling vehicle arrivals has been created based on real data using time-series/regression techniques (Lippi et al., 2013). A model for controlling the behavior of vehicles within the system is also introduced, so that it is determined by the data, and is accomplished at simulation time, and not by a pre-simulation route planner. The behavior of vehicles relative to acceleration and velocity also needs to be modeled, and the flow of vehicles within the traffic system needs to be realistic. A major contribution of our work has been to expand the capabilities of our simulation engine to handle these modeling challenges. The engine itself is a part of ScalaTion,

a system for simulation, analytics, and optimization (Miller et al., 2010). We expanded the capabilities of our software to handle large-scale simulations, which are needed for simulating and optimizing traffic systems. And we have incorporated mathematical models for car-following behavior, as well as free-driving behavior, to create a realistic flow of vehicles once they are in the system and traveling along roads. In the near future, with the potential mass proliferation of autonomous vehicles, there will be a tremendous need to use real-world traffic data to create high-fidelity traffic simulation models. Our work is a first step in an attempt to create such models.

The structure of the paper is as follows: Section 2.2 presents a summary of previous work in the field of traffic simulation and modeling. In Section 2.3, we present the structure of our traffic system, including the models for vehicle arrivals, vehicle behavior and traffic flow, as well as some detail about the structure of our simulation system. In section 4 we briefly investigate optimization techniques that could be used to optimize characteristics of a traffic system and give some thoughts on the appropriateness of each technique in the context of traffic simulation. Section 5 gives an accounting of our results, and section 6 presents our conclusions, and some of the ideas and avenues of research we plan on pursuing in the future.

2.2 Related Work

There have been many efforts made at creating models and simulations of traffic systems within the microscopic and macroscopic simulation paradigms. Macroscopic models have generally constrained network flow models that assume continuous streams of traffic flow through nodes. Individual vehicles and their behavior are not considered. This approach requires less computational cost than other approaches but concedes a lesser amount of detail in the results of the simulation. (Tampère et al., 2011) applied dynamic network loading to create models of traffic systems using simple merge-and-diverge models to represent the different types of connections roads can make inside a traffic system. These models consider the traffic system as a network of nodes and edges, with vehicle volumes equating to network flow, and the overall goal being to optimize this network flow over the entire system. They focus mainly on deriving generic requirements and constraints that such models must fulfill. (Flötteröd & Rohde, 2011) add to this work by building a more robust model for representing traffic flow.

Car-following models have been used to model traffic flow and the behavior of vehicles for a long time. One of the earliest models was proposed in (Gipps,

1981) and computes accelerations based on the differences between successive vehicles' velocities and locations. Gipps updated his models for his work on the MULTSIM traffic simulation system (Gipps, 1986). Other car-following models include the Optimal Velocity Model (Bando et al., 1995), the Generalized Force Model (Helbing & Tilch, 1998), the Full Velocity Difference Model (Jiang et al., 2001), and the Intelligent Driver Model (Treiber et al., 2000). Somewhat recently (Y. Li et al., 2011) formulated a car-following model based on the headways, velocities, and accelerations of multiple preceding vehicles.

The majority of car-following models are time-step driven and update mathematical formulas for each timestep, however, some models use event-based methods. In (Wiedemann, 1991) Wiedemann devised a psychophysical car-following model which combines mathematical concepts with observed psychological phenomena in drivers' reactions to events while driving. Another psychophysical model was presented in (Schulze & Fliess, 1997), where accelerations are only updated when certain thresholds in distances and speeds with leading vehicles are crossed.

Microscopic simulation models provide a much greater amount of detail than macroscopic models since individual vehicles and their behavior are represented with much more complex algorithms to control their movement and decisions. The obvious trade-off is that this requires a much greater computational cost, as simulations will usually contain hundreds or even thousands of vehicles in the system at the same time. The open-source traffic simulation platform SUMO-Simulation of Urban MObility was introduced in 2001 (Behrisch et al., 2011) which has been used by many researchers to validate their own models and to optimize characteristics of traffic systems. Another microscopic simulation platform is VISSIM, which is time step based and was used by researchers at the Georgia Institute of Technology (Hunter et al., 2006) to create traffic simulations based on real-world data. They generate vehicles using a Poisson counting process to produce random interarrival times.

Many simulation systems, including SUMO and VISSIM, are capable of using Open Street Maps to generate road networks. This feature makes modeling real-world traffic systems much easier, and it lends more credibility to the traffic simulations themselves.

Modeling traffic flows and vehicle arrivals is essential to simulating traffic systems in all three paradigms, and there has been much effort put into researching methods to create these models. (Lippi et al., 2013) used time series analysis and support vector regression to forecast traffic flows for short-term time periods.

Many different approaches have been used to optimize traffic light timings. (Spall & Chin, 1997) applied neural networks to the problem using the simulta-

neous perturbation stochastic approximation (SPSA) algorithm in the context of macroscopic simulation. (Ezzat et al., n.d.) used the third-party simulation software ExtendSim to create, execute, and optimize their traffic models. The software uses an evolutionary optimization approach. They based their system performance on both queue lengths and vehicle waiting times. (Osorio & Chong, 2012) used metamodels to optimize simulations of transportation systems. Their metamodel is based on a system of linear and nonlinear equations, which they test for suitability in reducing traffic congestion in a large-scale traffic system.

2.3 System Structure

In a real-world road network, there are a few specific events that occur as vehicles move around the system:

Arrivals In a simulation, only a restricted area of the road system is included and there must be a model that controls the arrival of vehicles to the network

Traffic Flow Once vehicles are in the network, their behavior as they move along roads should be as realistic as possible

Turning When vehicles arrive at intersections, they must choose a direction to continue their travel

In this system, we have formulated several essential models that govern the movement, arrival, and decisions of vehicles in the network. A model for the flow of traffic and the behavior of vehicles as they either drive freely or follow other vehicles is given. Also, models for vehicle arrivals and turning behavior, both of which are based on real-world vehicle count data, are presented.

2.3.1 Simulation Structure in ScalaTion

Our simulation model was built upon the ScalaTion system (Miller et al., 2010). Figure 2.1 shows a sample traffic system built in ScalaTion. The system uses several different types of components from the ScalaTion architecture, as well as components that were created specifically for traffic simulation purposes. ScalaTion's simulation system is discrete-event, however, since most car-following models are time-driven, we created a component to regularly schedule the car-following formulas to be updated.

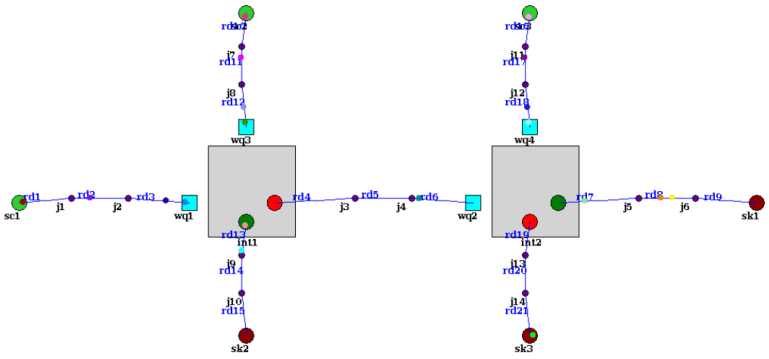


Figure 2.1: Sample Traffic System

Sources generate vehicles at interarrival times using a predefined random variate. In our system, this random variate is powered by real-world data so that vehicle arrivals are realistic. The construction of these random variates is described in a later section.

The concept of a Transport is that of a component that moves actors from place to place in a simulation. We advanced our available methods of movement by creating a **Road** component. A Road is used to move vehicles from one intersection to another with the motion controlled with the formulas outlined above. Roads are lightweight components that function as a guide so that vehicles have an easier time knowing their locations within the system. Functionally, the motion of the vehicles is controlled by predefined formulas.

Traffic signals are simulated using a **Gate** component, which can be used to control the flow of traffic by cycling between red and green phases. When a Gate is shut, this means the traffic light is in a red phase, and so the motion formulas produce a deceleration until the velocity of the lead vehicle is reduced to zero. There is no need for following cars to care about the state of the traffic signal, as they are merely reacting to the behavior of the car in front of them. When the Gate opens, signifying a green light, the lead vehicle begins to accelerate and move across the intersection. At this point, the vehicle will either turn or go straight, which depends on a turn choice algorithm described later in the paper.

The cars themselves are modeled using a **Vehicle** component, which is a specific example of a **SimActor** component. A Vehicle records its acceleration, velocity, and location along a Road at each time increment, and also holds a reference to the vehicle immediately preceding it. If there is no preceding vehicle, then it is the lead vehicle and can drive freely, which means it will asymptotically

approach the maximum speed. This behavior will only change if a traffic signal turns from green to red in front of it, which will necessitate a deceleration.

Sinks receive vehicles that are exiting the system. They also record the amount of time a vehicle spent in the system, as well as how many vehicles exited through it, which are both important metrics for analyzing, and eventually optimizing, the characteristics of a traffic system.

Each of the components can be located in a realistic fashion using GPS coordinates so that distances between landmarks are realistic. Currently, our model uses lines or simple curves for Roads, but we plan to implement more complex curvature of roads in the future. The data necessary for such models are harder to come by, but mapping services such as Google Maps, and OpenStreetMaps can often be used to achieve such constructions.

2.3.2 Traffic Flow and Car-Following Models

A car that is either in the lead or is far enough behind the immediately preceding car can be thought of as a free-driving vehicle. These vehicles will only be affected by the distance to, and state of, a traffic signal. If the traffic signal is red and the vehicle is close enough, then it must begin to brake. Our current braking model for freely driving vehicles is given by using the basic physics formulas

$$s = s_0 + vt + \frac{1}{2}at^2 \tag{2.1}$$

$$u = v + at \tag{2.2}$$

Assuming $s_0 = 0$, and substituting $t = (u - v)/a$ from Equation 2.2 into Equation 2.1, rearranging the result yields the formula

$$a_{new} = -\frac{v^2}{2s} \tag{2.3}$$

where v is the vehicle's current velocity and s is the distance between the vehicle's current location and the traffic signal. If the lead vehicle does not have to brake for a red traffic signal, then we use the following free-driving acceleration model

$$a_{new} = \min\{|\omega a_f + (1 - \omega)a|, |\delta(v_f - v)|\} \tag{2.4}$$

where a_f is the maximum free acceleration, ω is a weight parameter, a is the current acceleration, v_f is the maximum free velocity, v is the current velocity, and δ is a scaling parameter. This formula takes a weighted average of the maximum free acceleration and the current acceleration, which has the effect of gradually

increasing the velocity toward the maximum. However, if the current velocity is very close to the maximum velocity, then the new acceleration should be based on this, which will keep the lead car from going faster than the speed limit. The second formula is based on Gipps' basic model (Gipps, 1981), however, there is no component for the distance between cars since our formula is for freely driving vehicles.

Vehicles which are following another vehicle, and are close enough that they cannot drive freely use the Intelligent Driver Model (Treiber et al., 2000) to update the acceleration, velocity, and location of the vehicles.

2.3.3 Modeling Arrivals

Traffic systems are extremely complex, and creating a realistic model of vehicle arrivals is essential for simulations of traffic. To create such a model, real-world data collection is vital. In the U.S.A. there are a growing number of municipalities and states which are setting up data collection stations on roads and highways. The data being collected is largely in the form of vehicle counts, that is, the data collection station counts all vehicles which pass by it in a set time interval. Quite often, the actual times of the vehicle pass-bys are not kept. Therefore the granularity of the time intervals is very important.

Data Collection and Processing

To create valid and realistic simulations of traffic systems real-world data must be collected and analyzed. A large number of municipalities around the globe have started collecting data on traffic, which provides researchers with a great opportunity to create accurate models of traffic systems. The data used in our project is in the form of vehicle counts collected at multiple sensors along a suburban road in Kenmore, Washington, U.S.A. The sensors provide vehicle counts for every 5-minute interval of the day during a 17-week period starting in September 2013 and ending in January 2014. The data set contains information from both directions of the main roadway, broken down by lane, as well as information about most of the side streets.

We believe that the choice of road and intersection in our data reflects a typical suburban traffic system and that our work can be applied to almost any similar system. This also highlights the need for Big Data techniques for modeling more complex traffic systems. A high ratio of drivers will move through progressively busier traffic systems as they go to work each day, which means different traffic models will be needed to represent their entire commute. This

will require tremendous amounts of data, and processing of that data, to create simulations with which to work.

Vehicle Generation

There is a very large amount of vehicle count data available from around the world, and this data can be used to create mathematical models of vehicle arrival rates. It seems the majority of previous work in the area of traffic simulation uses vehicle arrival models which are based on simple Poisson counting processes, which are not necessarily even appropriate for modeling the arrivals of vehicles into a traffic system. We chose to model arrivals using a modeling approach that creates vehicles in times that are close to real-world data.

Two approaches are typically used. The first is to model the interarrival times of vehicles by modeling the time headway of vehicles on a road, which is defined as the distribution of times between vehicles passing the same geographical location on the road. The second method of vehicle generation is to model the number of cars that should enter the traffic system in any given interval of time.

Poisson counting processes, which are based on exponential distributions, are quite often used to model the time headway for vehicles entering a traffic system. (Q. Meng & Khoo, 2009) suggest that the Poisson process is not appropriate for use in modeling vehicle arrivals and that a better approach is to use self-similar processes which are used to model network traffic that exhibits fractal characteristics. (Leland et al., 1994) Other distributions that have been suggested include the Log-Normal distribution (L. Li et al., 2010), and the Gamma distribution (Dey & Chandra, 2009).

We must account for the ebb and flow of vehicle volume, as most busy intersections display a bimodal distribution of vehicle counts over the course of a normal business day. The two peaks correspond to the usual busy periods of morning and afternoon rush hours when the majority of workers are traveling to and from work, respectively. (L. Leemis, 2003) suggests that Non-Homogeneous Poisson Processes (NHPP) can be used to model arrivals in systems that exhibit multiple busy periods.

Our approach is to fit each day's vehicle count distribution with a polynomial, which is then used to drive an NHPP that generates interarrival times of vehicles to the system. Another option that we will explore in the future is using Poisson regression to create a fit to the data, and use this instead of a polynomial. We would likely need to use a Non-Homogeneous version of Poisson regression in our work.

Standard Poisson processes use a constant rate parameter λ, which is not appropriate for use in a model which will contain fluctuating arrival rates throughout the life of that model. The NHPP solves this problem by allowing for either a rate function $\lambda(t)$, or a vector of rates. Since we are estimating the counts for a whole hour, we use the latter approach, where we generate a discrete approximation to our polynomial curve and take these function values for our rate vector. This rate vector is used to build a cumulative rate vector, which represents how the arrivals build up over time. This cumulative rate vector is a piecewise-constant approximation to the cumulative intensity function. Using an exponential random variate, arrival times can be generated using linear interpolation. Since we have data from each direction heading into an intersection, it is possible to generate realistic interarrival times that display the varying traffic densities between main roads and side streets.

We validated the NHPP using a Kolmogorov-Smirnov (KS) test, which is a widely used method to compare samples. Based on the results of the test we believe that the vehicle counts are distributed as an NHPP. Figure 2.2 shows that our vehicle arrivals, generated by an NHPP, are very close to the vehicle counts from our real data set. The two samples have a KS-statistic of 0.0113026952 which passes a 95% confidence test.

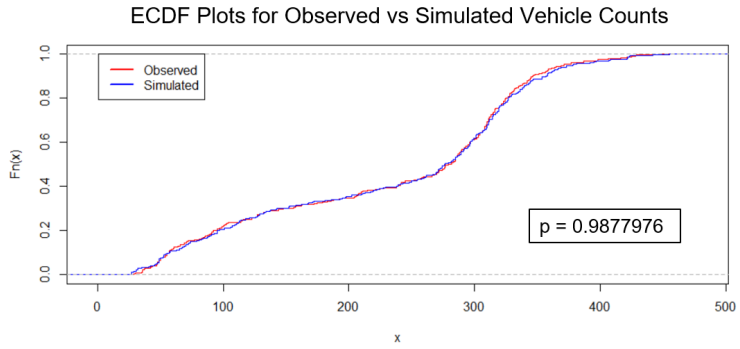


Figure 2.2: Visualization of KS Test For Real vs. Simulated Arrivals

2.3.4 Turning Behavior of Vehicles

When it comes to route choice, several different approaches have been used. (Esser & Schreckenberg, 1997) chose to give their vehicles predefined routes that detail the roads they will travel along within the system, usually based on origin-destination tables. Another common technique is to use simple randomizers that choose the next road based on a discrete random variable. A simple discrete random variable, which only uses constant probabilities to generate values, is

not appropriate since the probabilities of turn choices are certainly affected by the time of day, and even which day of the week you are representing.

We feel these methodologies can be improved upon by using real lane vehicle count data. Our approach is to use this lane data to create a probabilistic choice model that vehicles use to decide their route when they reach an intersection. This is fairly straightforward to do when you have access to the individual lane vehicle counts, including turn lanes.

First, we use the lane data to decide the vehicle counts for each of the three choices of turn left, go straight, and turn right. Second, we convert these vehicle counts into a percentage by dividing them by the total vehicle counts for all lanes. These percentages are shown in Figure 2.3.

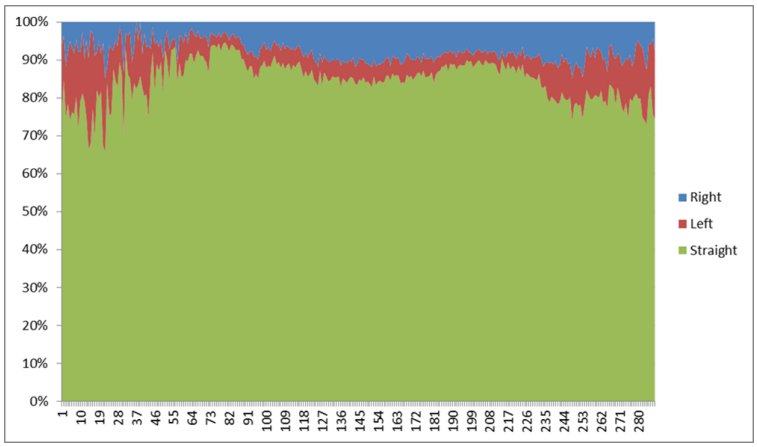


Figure 2.3: Percentages for Each Turn Decision

This gives us an estimate of the percentage of cars turning left, going straight, or turning right during each time interval throughout the day. We then generate polynomials $p_l(t)$ and $p_s(t)$ (representing the probabilities of turning left and going straight, respectively) using the percentage values for turning left and going straight. These polynomials are shown in Figure 2.4.

In both Figures 2.3 and 2.4 it is quite clear that this particular intersection has a main road and a much less traveled intersecting road since the majority of traffic is going straight, meaning they are continuing on the main road.

Finally, we create a discrete random variable based on these polynomials and a $U(0, 1)$ uniform random variable:

This method of creating turn choices is, we believe, a novel approach to the problem. It allows for much more flexibility to represent complex intersections, where the relative percentages of turn decisions can change drastically throughout the day. Not taking this issue into account leads to less accurate simulations which weakens their effectiveness as tools to analyze the real world.

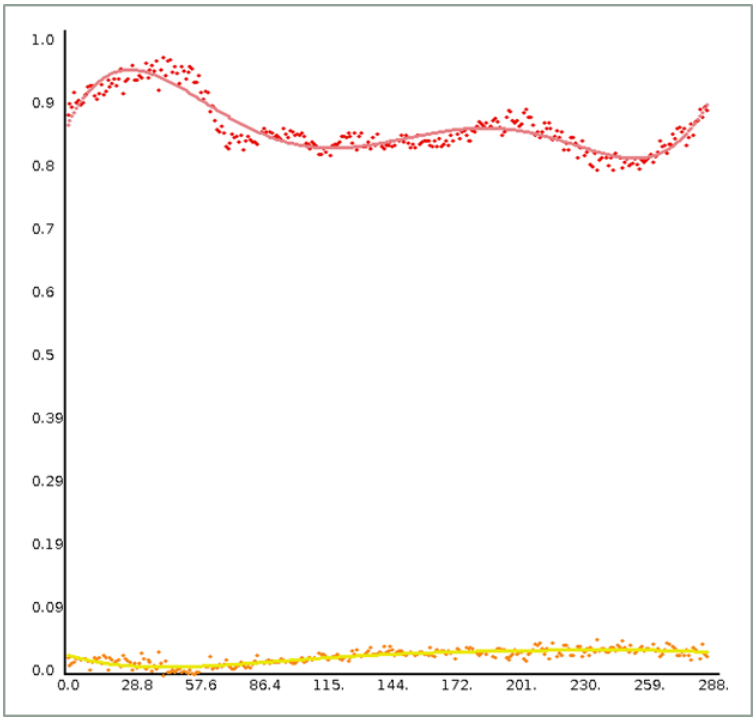


Figure 2.4: Polynomials for Going Straight (Red) and Turning Left (Yellow).

2.4 Simulation Optimization

There are many options to choose from when attempting to optimize the characteristics of traffic systems. These are stochastic simulations which makes it possible for the same input vector to yield different results from two independent simulation runs. Also, very small changes in the characteristics of traffic systems will likely have no real effect on the outcomes of simulations. So only somewhat large changes in these characteristics are particularly useful. However, it can be difficult to overcome the issue of noise generated by using real-world data, which can lead to massive differences in results when common sense would imply there should not be. All of these issues make the selection of optimization techniques a difficult one. Below, we briefly discuss the strengths and weaknesses of a few optimization techniques.

2.4.1 Gradient Techniques

Very small changes to the values of traffic system characteristics, such as traffic signal timings and speed limits, will not realistically result in significant changes

Algorithm 1 Turn Choice Algorithm

```
1:  procedure GenerateTurnChoice(t)
2:      u ← U(0, 1)                          ▷ Uniform random variable
3:      if u < p_l(t) then
4:          output 0                          ▷ Turn left
5:      else if u < p_l(t) + p_s(t) then
6:          output 1                          ▷ Go straight
7:      else
8:          output 2                          ▷ Turn right
9:      end if
10: end procedure
```

in the flow of traffic. This makes gradient-based techniques difficult to use since gradients are computed using very small changes in each variable. Finding a proper scale for these perturbations is an optimization problem in its own right, which adds an additional layer of complexity to the optimization.

BFGS

The BFGS (Broyden, 1970), (Fletcher, 1970), (Goldfarb, 1970), (Shanno, 1970)' quasi-Newton method makes use of the gradient and an approximation to the Hessian (matrix of second-order partial derivatives) of a function to iteratively move toward a solution. Line search algorithms are usually employed to decide how large of a step to take in each iteration. One problem with quasi-Newton methods is that the computation of the gradient requires many simulation runs. In fact, if the gradient is being computed using a symmetric difference quotient, there will be two simulation runs for each variable in the input vector. Some of this inefficiency can be removed through parallelization, but not all of it.

SPSA

The Simultaneous Perturbation Stochastic Approximation algorithm (Spall, 1998b) is an effort to remove much of the inefficiency of quasi-Newton methods by simultaneously perturbing all variables at once, resulting in only two simulation runs regardless of the number of variables in the input vector.

2.4.2 Gradient-Free Optimization

Nelder-Mead Simplex

This method (Nelder & Mead, 1965) is an unconstrained, derivative-free, direct-search optimization algorithm based on evaluating the objective function at vertices of a simplex. Each iteration typically requires only a few objective function evaluations, and so can be computationally less expensive than many other methods. The goal is to gradually decrease the function values at the vertices of the simplex as it is transformed. (Barton & Ivey Jr, 1996) show that there are some potential problems with applying the Nelder-Mead algorithm to stochastic objective functions in simulations, mainly due to the fact that changes to the simplex can take place erroneously, based on the stochastic nature of the responses.

Tabu Search

Another technique is to apply the Tabu search algorithm (Glover, 1989) to an integer domain of input values, as this search guarantees that you will not revisit input vectors already deemed to be sub-optimal. (Dengiz & Alabas, 2000) used the Tabu search algorithm for simulation optimization and found that it clearly outperformed a random search, giving credibility to its use. According to (Fu et al., 2005), with the cost of running each simulation being so high, the tabulation of the search space greatly improves the time efficiency of the optimization. With a stochastic objective function, as in simulation, there is always the chance that points previously considered will give different results if you revisit them.

Genetic Algorithms

If we restrict the timings of traffic signals to be integers, then the set of all feasible timing combinations is discrete. However, for most traffic systems this set is still too large to do an exhaustive search. Genetic algorithms (Holland, 1992) have been shown to have good optimization abilities in many applications. (L. Wang, 2005) introduced a hybrid approach to simulation optimization using GAs and Neural Networks (Werbos, 1974) when there is an unknown form to the objective function.

Genetic Algorithms are well suited for optimizing many characteristics of traffic systems because many of the values involved are discrete. It may not be helpful to think of traffic light timings as a continuous space of possibilities since small changes in timings are unlikely to cause noticeable differences in real-world vehicle behavior. Other characteristics such as the number of lanes

on a road, and the speed limit of the road are also going to be discrete sets, which are well suited for use with genetic algorithms.

2.4.3 Response Surface Methodology

Response surface methodology (G. Box & Wilson, 1951) holds some promise for traffic simulation optimization because all response values are computed before the optimization algorithm is applied, so no simulation runs are required during the optimization process itself. The process starts with a predefined lattice of input values and a simulation run is computed with each lattice point. This creates an implied surface of function values which can then fit by more well-defined surface using regression techniques. This surface can be any type of surface, though typically quadratic surfaces are chosen for their ease of use. This more well-defined surface can then be optimized using standard techniques. The process can also be repeated to identify better areas of the surface on which to focus.

This method is attractive because the stochastic nature of the simulations is removed during the optimization phase. Also, the choice of optimization technique is only dependent on the type of surface that has been created. Gradient and non-gradient techniques alike are all possibilities for this optimization. There are some potential drawbacks to RSM though. Response surfaces usually need to be fairly large in scope, which might require a large lattice of points on which to conduct simulation runs. If a multi-stage RSM process is being used, then the number of simulation runs needed to build the response surfaces might be too large.

2.5 Results

We believe our method of using an NHPP model for generating vehicles closely matches observed arrival times from real-world data. Changes in vehicle congestion as time passes, are handled naturally using our technique. This process can be done ahead of time so that the simulations rely only on the random variate constructed from the data. Our model for deciding turning behavior, which was based on real data, seems to closely align with what can be observed at normal intersections. And our traffic flow models show realistic movement of vehicles as lead cars drive freely, and following vehicles decide new acceleration and velocity changes based on the vehicles in front of them.

The implementation of these three models, namely, a vehicle arrival model, a turning model, and a traffic flow model, has greatly improved the capabilities

of our traffic simulation system. While there is still work to do, we believe we have created a system that can accurately simulate many real-world traffic scenarios. The validation plan is to compare traffic counts over an entire day produced by the simulation model with those recorded by the sensors for the road system under study. Counts are affected by the sources, vehicle speeds, the leading car model and following car model, the car turning model, the duration of traffic lights, and the synchronization of multiple lights.

We have also researched the possibility of optimizing the various characteristics of traffic systems through simulation optimization techniques. Several common techniques are examined and critiqued in the context of traffic simulation, specifically the optimization of traffic light timings. It appears to be a challenge to use gradient-based techniques in this context, as the issues with scaling and noise make it difficult to find good search directions and step sizes.

2.6 Conclusions and Future Work

In this paper, we presented a model for simulating traffic systems in a microscopic simulation paradigm. Realistic models for vehicle arrivals and turning behavior were created which closely match real-world data. We also presented formulas for the motion of vehicles within the system. Specifically, we implemented car-following behavior as well as free-driving behavior, which improves the quality of the overall model.

There is still much work to be done to improve our system. Currently, there is little facility for including multiple-lane roads, and no ability for vehicles to change lanes. There is also no current way in our system to model different modes of traffic as all vehicles are treated exactly the same. Our future work will include differentiating between different types of vehicles such as large trucks, buses, and possibly even bicycles and pedestrians. There is also a need to improve the mathematical formulas governing the flow of vehicles.

In the future, we plan to implement the use of open-source mapping information to create traffic simulations of real road systems. Open Street Maps is already being used by many traffic simulation systems, and its inclusion in our system is a definite goal.

We also plan to refine our model so that we can apply it to the many "what-if" scenarios in the domain of traffic research. For example, special events near a traffic system can greatly increase vehicle counts in that area. Another idea is to study the response of a traffic model to traffic accidents, and perhaps to formulate automated response plans for signal timing that can handle such

unforeseen occurrences. We also plan on exploring other types of traffic system structures such as roundabouts, amongst many others.

Lastly, we would like to investigate a growing area of interest in traffic modeling, which is autonomous vehicles. The future of driving is likely to include such vehicles, and traffic simulations that include them will be needed. Traffic systems with such vehicles will require the use of Big Data Analytics techniques to deal with extremely large amounts of real-time data and to react to changing conditions in an effective and timely manner. Autonomous vehicles provide the opportunity to create truly cooperative traffic systems, where the vehicles actually work together to improve efficiency and provide safe travel to human occupants.

Comprehensive real-world traffic planning will require sophisticated software systems to maximize the benefits of traffic simulation. The ability to integrate our models with other traffic simulation software, in an effort to broaden the capabilities of the entire system, could greatly aid in the use of traffic simulation for city planning, which is an increasingly important endeavor.

Chapter 3

Microscopic Discrete-Event Traffic Simulation

3.1 Introduction

Our busy traffic systems only continue to get busier as the number of cars on the road increases substantially from year to year. The challenge we all face is to create a safe and efficient road system that can handle the ever-increasing demands we put on it. Both simulation of traffic systems and traffic forecasting can be essential techniques in developing such systems and will be needed to investigate traffic scenarios ahead of time.

Consider the ability to use simulation to predict the consequences of major road construction before a project is even started. With autonomous vehicles on the horizon, there is an opportunity to see the impact of cooperative driving where vehicles are also teammates instead of simply individuals (Sichitiu & Kihl, 2008). There are several major steps involved in producing a simulation of a traffic system. For the simulation to be useful, it must be based on real-world data of some kind. There are many types of relevant data and how each is used to build a simulation model will be discussed (Hellinga, 1998). Once the data has been secured, processed, and analyzed, models of the various simulation components can be created. These include arrival models, traffic flow models, and the models of the roads themselves. Discrete-event simulations (DES) are well-suited for accurately representing traffic systems since they allow for modeling changes as they are needed, as opposed to discrete-time simulation (DTS), which updates a model at specified times. For instance, when vehicles are created, a DES can produce them at the exact time they should be, while a

DTS will have to use a random variable to decide how many cars were produced since the last update, and then also calculate where they should be along their respective roads.

Short-term traffic forecasting can be used to help drivers navigate through or around congestions, accidents, and other complex traffic situations (Vlahogianni et al., 2014). Traffic parameters of interest for forecasting include travel time, traffic volume, traffic speed, queue length, etc. Traffic apps such as Waze, Google Traffic, and INRIX are commonly used apps for forecasting traffic. These forecasting techniques are useful for simulation purposes for a few reasons. First, they can provide a basis for modeling certain characteristics of a simulation such as vehicle arrivals, or likely routes through a traffic network. Forecasting can also be used as a means to validate a simulation's structure and results. Traffic forecasting is therefore an indispensable tool for the successful modeling and simulation of traffic networks.

Recently advancements in mobile and wireless technologies have enabled an increased amount of traffic data to be collected (Herrera et al., 2010). Increasing numbers of permanent and temporary sensors are also collecting great volumes of data. The availability of large quantities of high-resolution traffic data has greatly facilitated the improvement of both simulation of traffic systems and short-term forecasting in recent years.

Predictive modeling of traffic is inherently difficult because of variability in drivers, variability in roads, until recently, a limited amount of data to work with, highly chaotic and dynamic systems. In addition, many factors come into play, including weather, events, accidents, etc. Furthermore, traffic systems become very complex very quickly as the scale of the network is enlarged.

The rest of this paper is organized as follows: Section 2 discusses various types of simulation models. Data collection and analysis are given in Section 3. Section 4 focuses on different short-term traffic forecasting models. Challenges and directions for future work to improve the accuracy and robustness of traffic modeling and simulation are considered in Section 5. Finally, our conclusions are presented in Section 3.5.

3.2 Types of Simulation Models

A traffic simulation model exists in any of three paradigms: macroscopic, mesoscopic, and microscopic. A model can also be implemented using either DTS or DES. Buss compared the results of discrete-time and discrete-event simulations when using differential equations to calculate changes to the system (Buss & Al Rowaei, 2010). They found that the choice of time-step has a large effect on

the accuracy of the DTS models, but also that errors resulting from a DES approach were smaller in general than the DTS approach. However, (Lieberman & Rathi, 1997) believe that DTS systems are a better choice for large traffic systems that require a great amount of detail. A discrete-event simulation system with thousands of vehicles will likely process many more updates to the system than would a discrete-time simulation. As processors become faster with higher parallelism and distributed computing techniques continue to improve, the efficiency of DES should get better as well, which along with the better accuracy of such an approach, likely makes DES the better choice for traffic simulation moving into the future.

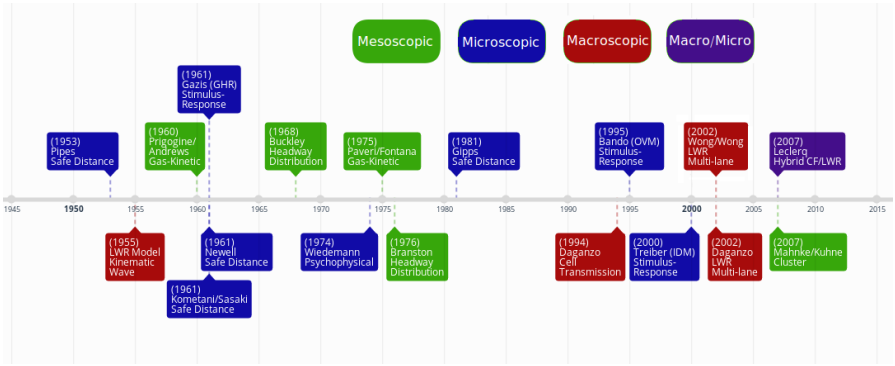


Figure 3.1: Timeline of traffic simulation models. (Gipps, 1981), (Pipes, 1953), (Kometani & Sasaki, 1961), (Lighthill & Whitham, 1955), (Richards, 1956), (Newell, 1961), (Gazis et al., 1961), (Prigogine & Andrews, 1960), (Buckley, 1968), (Paveri-Fontana, 1975), (Wiedemann, 1974), (Branston, 1976), (Bando et al., 1995), (Daganzo, 1994), (Treiber et al., 2000), (Daganzo, 2002), (Wong & Wong, 2002), (Leclercq, 2007), (Mahnke & Kühne, 2007).

3.2.1 Macroscopic Models for Traffic Simulation

Most of the early work in traffic modeling was in the paradigm of macroscopic traffic models. Greenshields' work is the starting point for the traffic flow models that would come later (van Wageningen-Kessels et al., 2015). His work comparing velocity to traffic density led to some of the early breakthroughs in the field and inspired many researchers to pursue traffic modeling.

A notable contribution was provided by (Lighthill & Whitham, 1955) and (Richards, 1956), with the formulation of the Lighthill-Whitham-Richards (LWR) kinematic wave model of traffic flow. The idea behind this application of kinematic wave theory is that changes in traffic flow propagate backward through traffic in a wave-like fashion and that multiple waves can even collide forming kinematic "shock waves". However, deficiencies in the basic LWR model

were identified (Daganzo, 1997) stemming from the fact that the LWR model makes some unrealistic assumptions about traffic flow. First, the original model assumed an instantaneous change of velocity, which would imply an infinite acceleration. Secondly, all vehicles in a geographically defined neighborhood or platoon are assumed to have the same desired velocity. However, real-world data has shown that vehicles in a platoon will have their own desired speeds and this will lead to the platoon getting spread out and eventually disentangling (Daganzo, 1995).

3.2.2 Mesoscopic Models for Traffic Simulation

Mesoscopic models utilize elements of both the macroscopic and microscopic paradigms of traffic flow modeling. Specifically, individual vehicles are considered and modeled, but the overall flow is controlled by macroscopic features (Zhou & Taylor, 2014). (Buckley, 1968) proposed a traffic flow model based on arrival modeling using a semi-Poisson distribution. (Branston, 1976) also discussed a traffic flow model based on an arrival model. (Prigogine & Andrews, 1960) proposed gas-kinetic traffic models with (Paveri-Fontana, 1975) improving on the concepts later. (Helbing, 1997) created a multilane version of this model. Later, a generic gas-kinetic model was introduced by (S. P. Hoogendoorn & Bovy, 2001). The INTEGRATION model (Van Aerde & Yagar, 1988) was originally mesoscopic in nature, though it has seen significant evolution since the original formulation (Van Aerde et al., 1996).

3.2.3 Microscopic Models for Traffic Simulation

When vehicles are in close proximity in the same lane, then one car is following another car and must change its own behavior as the car just in front of it also changes. These changes in behavior will result in changes to either the acceleration or velocity of the vehicle. The reasoning behind the changes boils down to either a general response to stimuli or because of a desire to avoid collisions. When humans make such decisions we either press the accelerator pedal or the brake (decelerator) pedal. Thus, the decision boils down to what the new acceleration of the vehicle should be. There have been many models proposed for producing new accelerations for the following vehicles. The new acceleration value is used to update the velocity of the vehicle, which is then used to update the position of the vehicle. Many car-following models have been proposed with most falling into the classifications below (Brackstone & McDonald, 1999).

Stimulus-Response Models

Stimulus-response models assume drivers change their behaviors based on different stimuli. If drivers are not traveling at their own desired velocities then they will choose to either accelerate or decelerate depending on their speeds. Drivers will also adjust their speeds depending on either the spacing between them and the car in front of them or the relative velocity of the car in front of them.

Some of the earliest work on car-following models was done by (Gazis et al., 1961), creating the GHR model, which has inspired many other models. For example, Bando et al. proposed the Optimal Velocity Model (OVM) (Bando et al., 1995), (Bando et al., 1998), and Treiber et al. designed the Intelligent Driver Model (IDM) (Treiber et al., 2000), (Kesting et al., 2010).

Collision Avoidance Models

Collision Avoidance models work based on the idea that drivers will maintain distances that will prevent collisions with the vehicles in front of them. Some of the earliest work in collision avoidance models was done by (Pipes, 1953) and (Kometani & Sasaki, 1961). (Gipps, 1981) refined the ideas considerably and is still considered to be the leading model in the collision avoidance paradigm (Ciuffo et al., 2012). Several of the leading models are presented below.
The GHR model is shown below in equation 3.1

$$\dot{v}_n(t+\tau) = p\frac{\dot{x}_{n-1}(t) - \dot{x}_n(t)}{[x_{n-1}(t) - x_n(t)]^l} \tag{3.1}$$

The IDM is shown below in equations 3.2 and 3.3

$$\dot{v}_n(t+\tau) = a_n \left(1 - \left(\frac{v_n(t)}{V_n}\right)^{\delta} - \left(\frac{s^*(v_n(t), \Delta v_n(t))}{s_n(t)}\right)\right)^2 \tag{3.2}$$

where

$$s^*(v_n(t), \Delta v_n(t)) = s_0 + v_n(t)T + \frac{v_n(t)\Delta v_n(t)}{2\sqrt{a_n b_n}} \tag{3.3}$$

The OVM is shown in equations 3.4 and 3.5

$$\dot{v}_n(t) = \gamma\left(v^*\left(s_n(t)\right) - v_n(t)\right) \tag{3.4}$$

where

$$v^*(s) = V_0 \left(\tanh(s - c_1) + c_2\right) \tag{3.5}$$

Gipps' Model is presented in equations 3.6, 3.7, and 3.8.

$$v_n(t + \tau) = \min \left[v_F(t + \tau), v_C(t + \tau) \right] \tag{3.6}$$

where

$$v_F = v_n(t) + 2.5 a_n \tau (1 - v_n(t)/V_n)(0.025 + v_n(t)/V_n)^{1/2} \tag{3.7}$$

and

$$v_C = b_n \tau + \sqrt{b_n^2 \tau^2 - b_n [2[x_{n-1}(t) - s_{n-1} - x_n(t)] - v_n(t)\tau - v_{n-1}(t)^2} \tag{3.8}$$

Cellular Automata Models

Cellular Automata (CA) models are microscopic models because individual vehicles are created for the simulation, but space is no longer continuous, and vehicles move between "cells" that model individual locations on the road, with a small length to allow a single vehicle to occupy a cell at any given point in time. (Cremer & Ludwig, 1986) created an early CA system that modeled everything using boolean operations. (Nagel & Schreckenberg, 1992) showed that the CA approach results in behavior predicted by macroscopic models. The proposed rules for motion in this approach were extremely simple:

1. **Acceleration:** if the velocity v of a vehicle is lower than v_{max} and if the distance to the next car ahead is larger than $v + 1$, the speed is advanced by one $[v \rightarrow v + 1]$.
2. **Slowing down (due to other cars):** if a vehicle at

Another expansion of the idea was proposed in (Helbing & Schreckenberg, 1999) in which the CA model is combined with the Optimal Velocity Model.

Psycho-Physical Models

Another class of car-following model is based on the idea that drivers can only have reactions if they have *perceived* a driver ahead of them that they should react to. (Wiedemann, 1974) is often credited with the early work in the field.

3.3 Data Collection and Analysis

Traffic systems produce large volumes of data and it is crucial to know the relevant types, their availability, and how they can be used for traffic modeling (Hellinga, 1998). Below are some of these relevant data types.

- **Vehicle Count Data**: Most states provide public access to vehicle count data, usually through a web interface. The frequency of the counts is important for accurate arrival models. Hourly traffic data is very common to find but is not nearly as accurate as five-minute intervals.

- **Speed Limit Data**: Speed limit data can be hard to find, but there are sources available for estimates of the correct values. The Google maps API provides a function for retrieval of speed limits for a particular road segment. Google does not guarantee that these speed limits are accurate. OpenStreetMap can also be used to estimate speed limits, but some advanced work must be done. Each state publishes the maximum speed limit for each type of road they have. These road types are typically included in the description of a road in OpenStreetMaps, so an indirect estimate of the speed limit can be done. Note that this method may only be suitable for roads in the United States. Other countries and regions may not have published maximum speeds as in the U.S.

- **Geo-Spatial Data**: Traffic flow is heavily influenced by the shape of the road, especially if collisions are a part of the simulation study. Blind curves, steep hills, narrow lanes, etc. all have an impact on traffic. Much of this data is publicly available and can be processed using GIS software. Geospatial data is also often used to build the road structure in the simulation even if the data is not being used to affect the traffic flow.

- **Travel Time Data**: Typically, these data provide travel times between two locations. Some governments provide such data, e.g., the UK (data.gov.uk), Ireland (data.gov.ie), and Australia (data.qld.gov.au, data.vic.gov.au).

- **Accident Data**: An online system would require accident data to efficiently reroute vehicles. These data are available through web interfaces of state governments or traffic apps such as Waze.

- **Event Scheduling Data**: In cities, events such as sports, concerts, celebrations, graduations, etc. are usually available ahead of time.

- **Construction Data**: Road construction data are typically posted by state and local governments.

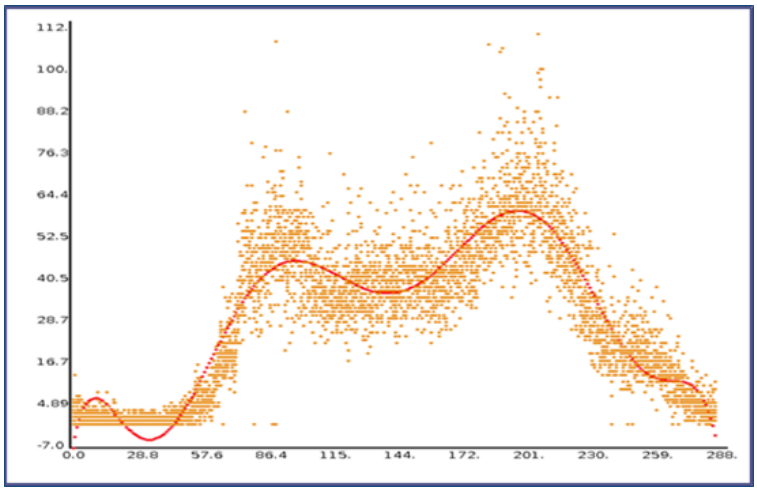


Figure 3.2: Vehicle counts vs. a polynomial fit

Vehicle counts can be used to determine interarrival times for the system and its sources. Figure 3.2 shows the vehicle counts on Mondays for 17 weeks and polynomial regression is fit using the data. Speed limit data can be used to estimate the range of speeds at which vehicles would operate. Geo-spatial data could even be incorporated to model accelerations and decelerations due to changes in the physical shape of the road. For instance, severe curves in a road would necessitate decelerations. (Wilkie et al., 2012) use GIS data to create three-dimensional models for use in traffic simulations. (X. Wang, 2005) discussed the integration of GIS methods and data with simulation models and visualization techniques.

The most conventional way to collect traffic data is through inductive loop detectors that are already deployed on many roads (Leduc, 2008). Magnetic fields are generated by loop detectors in order to detect vehicles, which are mostly made of metals. When vehicles pass through loop detectors, traffic count data can be recorded. If two loop detectors are very close to each other, they can also calculate the speeds of the vehicles that pass by.

Traffic data may also be collected by probe vehicles, which can be vehicles specifically deployed on the roads for real-time traffic data collection or commercial vehicles like taxis equipped with GPS chips (Leduc, 2008). Typically satellites and cellular networks can be used to transmit information such as locations and speeds of probe vehicles. The CarWeb system proposed by (Lo et al., 2008) utilizes GPS and Mobile networks to obtain position and speed

data on non-freeway roads, which are less likely to have a comprehensive system of in-road sensors. With the relatively recent drastic increase in the number of smartphones, the amount of data that can potentially be collected has also sharply increased. Popular traffic apps such as Waze, Google Traffic, and INRIX may collect anonymous data from users who are using the apps while driving.

Other ways to collect traffic data may include using automated toll collection stations to collect travel times data (El Faouzi et al., 2009); (S. Hoogendoorn et al., 2003) discussed using aerial images to extract more detailed data from road systems, specifically for use with microscopic traffic simulations; recently, in (Bhaskar & Chung, 2013), the use of Bluetooth scanners to collect traffic data was suggested.

3.4 Types of Forecasting Models

Many forecasting models have been developed for forecasting time course data. These models may include statistical time series models and machine learning models. The applications of forecasting traffic variables or metrics using different models have been an area of growing research interest. Our interest in the field results from the idea that traffic forecast models can be used to drive elements of traffic simulations. For example, an accurate forecast of traffic volume can be used to simulate vehicle arrivals to the network.

3.4.1 ARIMA Family of Models

One of the most intuitive approaches to forecast traffic variables or metrics of interest such as traffic volume or travel time is to use the data from the recent past to predict the value of the variable in the immediate future. It would be reasonable to assume that the traffic volume on a particular road fifteen minutes later would be highly correlated to the current traffic volume. The univariate autoregressive integrated moving average (ARIMA) family of models (G. E. Box & Jenkins, 1970) commonly used in time series analysis can be a good candidate for such a task.

The autoregressive (AR) portion relates the current variable of interest (e.g., travel time) to the same variable at the last p time points. Let Z_t represent the travel time at time t between two Traffic Control Sites, and define $Y_t = Z_t - \mu_Z$, where μ_Z is the mean of the Z time series. In particular, the pth order autoregressive model for Y_t, AR(p), may be expressed as

$$Y_t = \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \epsilon_t$$

where ϕ_i is the parameter/coefficient associated with the ith lag Y_{t-i} and ϵ_t is the white noise process.

As an example, the ScalaTion project may be downloaded from http://www.cs.uga.edu/~jam/scalation_1.3/README.html. The dataset used contains minute-by-minute travel time data from midnight to 1:00 pm from Traffic Control Site 2127 to 175 in Dublin, Ireland (Council, 2016). This simple test creates an AR(1) model to forecast travel times 15 minutes into the future. By using the "rolling forecast origin" technique (Hyndman & Athanasopoulos, 2014) commonly used to validate time series models, only the most recent n (in our case $n = 60$) instances are used as training data to produce 15-step ahead forecast on the $n + 15$-th time point. To run this test, simply type "run-main apps.analytics.Traffic_AR" in the sbt console. An R^2 value of 54.2% can be obtained. The forecasted (red) and actual (black) travels times are shown in Figure 3.3.

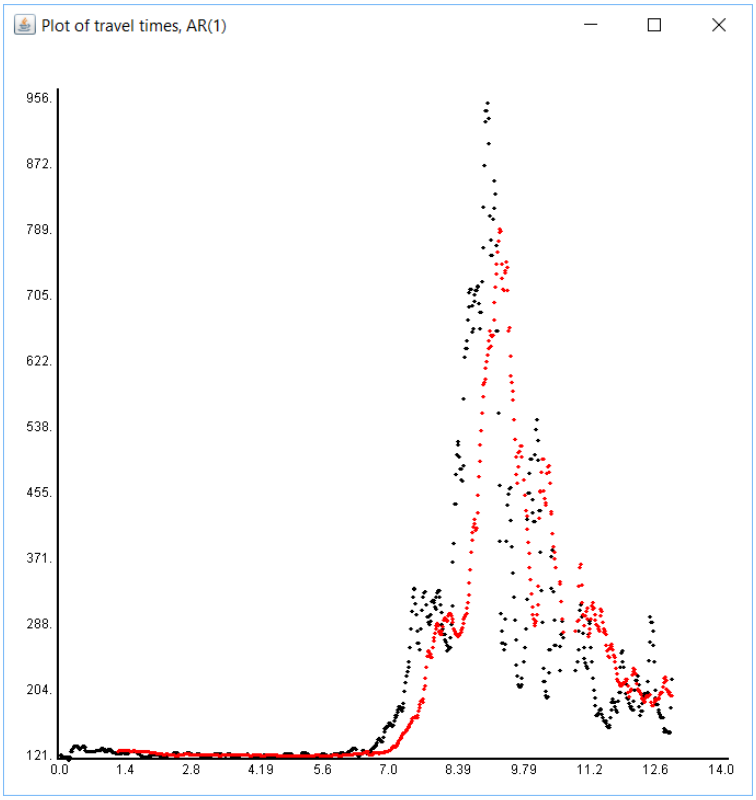


Figure 3.3: Travel Times: Forecasted (red) vs. Actual (black)

The ARIMA family of models has been used for short-term traffic forecasting for almost four decades (Ahmed & Cook, 1979), (M. Levin & Tsao, 1980). However, a major shortcoming of the ARIMA family of models is its inability to quickly respond to sudden changes in the traffic condition such as conges-

tion caused by an accident or sharp increase and decrease in traffic volumes (Vlahogianni et al., 2004). In recent years, the ARIMA family of models has mostly either been used as a baseline comparison or a component of a hybrid or more generalized model.

An extension to ARIMA with seasonal components is called the SARIMA model. Seasonality typically denotes similar or repeated patterns in the univariate time series for every fixed period of time, such as the daily traffic volume patterns on workdays. SARIMA was first used to forecast the traffic flow of urban freeways in (B. Williams et al., 1998). A recent study in (Kumar & Vanajakshi, 2015) demonstrated the potential of SARIMA models fitted with only a limited amount of input data for traffic flow forecasting when compared with non-seasonal ARIMA models.

3.4.2 State-Space Models and Kalman Filter

Suppose one desires to study the relationship between a traffic metric or variable of interest and other relevant traffic variables, the univariate ARIMA family of models may not be sufficient for such a task. For example, information on departure times and past travel times can give valuable insights into forecasting future travel times, but the univariate ARIMA family models are not able to take advantage of the extra information for prediction. One solution would be to use multivariate generalizations of the ARIMA models, such as in (Schimbinschi et al., 2017), which showed that a vector autoregressive (VAR) based model can outperform baseline ARIMA models in traffic flow predictions. Another solution would be to use the Kalman Filter (Kalman et al., 1960), a widely applied algorithm in multivariate time series analysis.

In essence, the Kalman Filter (Kalman et al., 1960) attempts to use the estimate of the state of a system (e.g., the state may consist of traffic volume and velocity) and the degree of uncertainty of the estimate of the state (to account for noise and measurement errors) at time t, and produces an estimate or forecast of the state and its degree of uncertainty at time $t+1$. Any relevant measurements of some external influences on the state (e.g., weather condition) at time $t+1$ (with its own degree of uncertainty) can also be used to adjust the estimate of the state of the system at time $t+1$. A model that utilizes the Kalman filter algorithm falls into a more general category of state-space models.

The Kalman Filter was first used to forecast traffic volume in (Okutani & Stephanedes, 1984). Application of travel time forecasting using the Kalman Filter was done in (Chien & Kuchipudi, 2003). In (Stathopoulos & Karlaftis, 2003), the state-space models using the Kalman Filter outperform simple ARIMA models in traffic volume forecasting. The difference in performance, in terms

of the mean absolute percent error (MAPE), is as high as 8%. A more recent study in (Guo et al., 2014) combined the Kalman filter with univariate time series models to produce better forecasts of traffic flow rate than the individual models.

3.4.3 Regression

Aside from multivariate time series analysis, another way to forecast a traffic variable using other traffic or traffic-related variables is regression, a very common technique in predictive analytics. The goal of regression is to find a function that best describes a given dataset. In other words, the differences between the observed values of the traffic variable of interest and the fitted values (the predicted values of the traffic parameter of interest given by the function) must be minimized. Typically, the function maps multiple predictors (traffic-related variables) to a single response (the traffic variable of interest). A major class of regression techniques is known as parametric regression, in which the forms of the function (e.g., linear, quadratic, higher-order polynomials, generalized linear models, etc.) is pre-determined and therefore the goal is to find the coefficients (parameters) on the predictors so that the function may best describe the dataset. A widely used yet simple parametric regression model is the linear regression model

$$Y = b_0 + b_1 X_1 + b_2 X_2 + \cdots + b_n X_n + \epsilon$$

where Y is the response variable; X_j is the jth predictor, $j = 1, 2, \ldots n$, where n is the total number of predictors; $b_0, b_1, \ldots b_n$ are the coefficients/parameters that need to be fit; and the ϵ random variable is used to represent residuals or errors. This model may also be generalized to polynomial regression models by including non-linear predictors (e.g., squared values of predictors).

In (Nikovski et al., 2005), linear regression was shown to be competitive with some non-linear techniques such as neural networks and k-nearest neighbors when performing univariate forecasting of short-term travel times. The resolution of the data is 5 minutes and only data from the past two time points (5 and 10 minutes prior) are used as predictors in linear regression. In (Zhang & Rice, 2003), a linear regression-based model in which the coefficients/parameters vary according to departure time (as opposed to being constant in standard linear regression) was proposed, outperforming baseline predictors that rely solely on current traffic information or historical averages.

Non-parametric regression, unlike parametric regression, does not have a pre-determined form. Both the structure and parameters of the model must be learned from the data.

In (Smith et al., 2002), a study is conducted to compare traffic flow forecasting accuracies of SARIMA and non-parametric regression. Even though SARIMA was found to be superior in terms of MAPE, non-parametric regression can nevertheless be a valid alternative in scenarios where SARIMA may not be applied.

3.4.4 Neural Networks

The regression techniques can provide simplicity with the pre-defined form of the function to be fitted, but this may also be a disadvantage because traffic situations are often complex, involving sudden extremes due to accidents, bad weather, etc., which can be difficult to capture by using a fixed form of function. Neural Networks, which can be understood as a form of multi-layer nonlinear regression, could be an alternative for such a task. In recent years, Neural Networks have captured the attention of many researchers in the field of traffic forecasting (Karlaftis & Vlahogianni, 2011).

Neural Networks are a data-driven computation model in machine learning. Since the number of parameters that need to be learned in a Neural Network can be very large, Neural Networks typically require a lot of data. This restriction, however, is no longer a major issue as more and more data are becoming available in this so-called age of Big Data.

A Feedforward Neural Network is made up of interconnected layers of artificial neurons, inspired by the structure of a brain. The incoming signals of a neuron are first aggregated, then the aggregated signal is passed through an activation function to produce an output signal for the neuron to send forward to other connected neurons. The strength of the connections among the neurons is learned from the input data.

Much research has been devoted to using Neural Networks for traffic forecasting since the early 90s (Dougherty, 1995). Recently, in (Schimbinschi et al., 2015), a comparative study of different techniques was done on short-term traffic volume forecasting techniques. The forecasting problem was formulated as a classification problem (high traffic, low traffic). Neural Networks were found to generally have superior performances in both forecasting accuracies and efficiencies than RUSBoost (Seiffert et al., 2010), Linear Discriminant Analysis (LDA), classification trees, Support Vector Machines (SVM) using Radio Basis Functions (RBF) kernel, Naive Bayes and k-Nearest Neighbors (kNN).

Traffic volume data collected by 1084 sensors for 2033 days were used in this study, and there were 96 data points per day, in 15-minute intervals. In an initial algorithm selection step, both logistic regression and forward feeding neural networks rank high in terms of forecasting accuracies and running time, outperforming other techniques including RUSBoost (Seiffert et al., 2010), LDA, classification tree, SVM using RBF kernel, Naive Bayes and kNN. A more detailed comparison between logistic regression and feed-forward neural networks suggests that feed-forward neural networks generally perform better.

3.4.5 Other Forecasting Techniques

Other traffic forecasting techniques include Bayesian Networks (Sun et al., 2006), functional regression and classification (Chiou et al., 2012), among others. More comprehensive reviews on traffic forecasting may be found in (Vlahogianni et al., 2004), (Vlahogianni et al., 2014), and (Mori et al., 2015).

3.4.6 Traffic Apps

Some of the most popular traffic apps include Waze, Google Traffic, and INRIX. Typically, those apps rely on drivers using the apps while driving to collect real-time anonymous data through their mobile devices that are present in vehicles. Information such as the speed of mobile devices reflecting the speed of vehicles and the number of mobile devices moving around on different roads can be used to provide the current traffic condition (Herring et al., 2010).

3.5 Challenges and Future Work

Below are some specific challenges that we feel are important to society and need to be addressed in the field of traffic simulation in the future.

3.5.1 Discrete-Event Simulation

Many microscopic traffic simulation models use a DTS approach. (Florian et al., 2008) proposed that DES is more efficient, where the system would only update vehicles when it needs to. They mention that DES will also allow for times in a continuous space instead of a discrete space, which should improve results and reflect reality to a higher degree. They present a discrete-event system using a simplified car-following model based only on the positions of the lead and following cars, the response time of the driver in the following car, and the effective vehicle length of the following car. (Sumaryo et al., 2013) and (Salimifard &

Ansari, 2013) both considered the problem of using discrete-event modeling in traffic light simulation. (Burghout et al., 2006) developed a mesoscopic traffic simulation model for use with a discrete-event, hybrid mesoscopic-microscopic simulation. (Thulasidasan et al., 2009) used a parallel DES system for large-scale microscopic traffic simulation.

3.5.2 Calibration And Validation

Calibration and validation are difficult to carry out for several reasons. For instance, collecting enough data is difficult, and there may not be enough for calibration or validation. Also, geography is important and models are highly dependent on the location under analysis. It can also be difficult to account for the variability found among drivers. It is best to use real road data to validate traffic models, but closed courses with a controlled collection of drivers have also been used. (Henclewood et al., 2012) argued for using real-time calibration for online traffic simulation systems. They point out that even in the same geographic area, parameters found through calibration can be inappropriate at other times of the day.

3.5.3 Scaling

Traffic simulations are difficult to scale up as the number of vehicles and the complexity of their interactions can explode in even a fairly small geographic area. Recently, with growth in the areas of parallel and distributed computation, efforts have been made at improving the scale of traffic simulations. (Fujimoto, 2015) presented an overview of the current state of parallel and distributed simulation, and includes traffic simulation as a major motivator for the field. (Thulasidasan & Eidenbenz, 2009) proposed FastTrans, a parallel simulator using distributed memory techniques for traffic simulation. They also show how the choice of search algorithm for routing can affect the overall performance of the system. (Hanai et al., 2015) discussed the problem of multiple runs of what-if scenarios for large-scale traffic simulations and proposed a filtering technique to reduce the number of scenarios (Suzumura & Kanezashi, 2013), (Kanezashi & Suzumura, 2015). (Zehe et al., 2015) presented a tutorial on a cloud-based simulation service, with a specific implementation example being the simulation of urban traffic.

3.5.4 Autonomous Vehicles

Autonomous vehicles will open up many new avenues of research into traffic systems and will bring about new opportunities for traffic management. A solution could be to have inter-vehicle communication. (Sichitiu & Kihl, 2008) gave an overview of various inter-vehicle communication methods. (Suh et al., 2017) discussed data-driven transportation systems.

(Ishikawa & Arai, 2015) considered the impact of intelligent vehicles that can relay important information to other vehicles to prevent traffic jams. (Fernandes & Nunes, 2010) discussed implementing vehicle-to-vehicle communication within the SUMO (Simulation of Urban MObility) simulation system (Krajzewicz et al., 2012). (Pereira & Rossetti, 2012) proposed microscopic traffic simulation as a testbed for theoretical aspects of autonomous vehicles. (Hasebe et al., 2003) formulated an extension to the Optimal Velocity Model in which a vehicle can look ahead or behind some number of cars to calculate its new acceleration.

3.5.5 Intelligent Traffic Lights

Intelligent traffic light systems would be highly adaptive and also cooperative with other traffic lights. Traffic data sensors can send information to the lights and the timings can be adjusted to meet the current demand of the intersection. If the problem cannot be solved by a single traffic light, then other traffic lights are adjusted within a specified radius (perhaps adjustable as well), to attempt to reach a solution.

(Zozaya-Gorostiza & Hendrickson, 1987) and (Radwan et al., 1990) both designed knowledge-based expert systems to adjust the parameters of traffic lights at single intersections. (Hunt et al., 1982) created the SCOOT (Split Cycle Offset Optimisation Technique) system which has been used extensively in England, as well as other parts of the world, for adaptive control of traffic lights. (Semarak, 1996) proposed a fuzzy-logic approach to controlling traffic lights.

3.6 Conclusions

There has been a tremendous amount of work in the past decades to accurately model and simulate traffic networks, though there are still many questions and avenues of research to be worked on. In this paper, we have presented many of the most important results from the past, and given an indication of the important issues still left to be resolved. There is a discussion on autonomous

vehicles, which have attracted increasing attention from both researchers and consumers. We have also briefly reviewed some techniques commonly found in the field of traffic forecasting, which is an important goal of traffic simulation. It is important to pursue research in both forecasting and simulation to build more reliable traffic prediction systems. More reliable prediction and simulation systems will allow us to solve many of the problems still plaguing our roads, such as traffic congestion and safety. They will allow, as an example, for more efficient programming of traffic lights, which could have an immediate impact on traffic congestion. Forecasting, modeling, and simulation will also certainly play a large role in the continued development of autonomous vehicles, which hold the hopes of having much safer roads to travel.

Chapter 4

Arrival Modeling

4.1 Introduction

Many types of simulations require the modeling of entities that arrive to the system at various randomly distributed times. This requires the generation of a sequence of arrival times $T = \{t_1, t_2, ..., t_n\}$ where $0 \leq t_1 \leq t_2 \leq ... \leq t_n$. Each entity e_i is then generated at the corresponding arrival time t_i.

Law (2007) defines an *arrival process* as $\{N(t), t \geq 0\}$ where $N(t) = max\{i : t_i \leq t\}$ is the number of events that occur at or before time t. Another important time-oriented value is defined as $A_i = t_i - t_{i-1}$ which is the *interarrival time* between two consecutive entities. In terms of traffic modeling, this interarrival time A_i represents the initial *time headway* between two consecutive vehicles in the same lane of traffic, which is a crucial value in terms of safe driving behavior.

The distribution of interarrival times is not a time series, but it is related to the time series of arrival counts of entities to the simulation. One method for generating a random variable for interarrival times involves using the time series of arrival counts. If arrival counts can be forecast, and a distribution is assumed for that time series data, then it is possible to use that distribution to recover arrival times, and therefore interarrival times for the time series. A procedure for this process when assuming the vehicle arrivals come from a nonhomogeneous Poisson distribution is presented in Section 4.2.1.

4.2 Related Work

The problem of determining vehicle arrival times is related to the problem of representing time headway distributions. Time headway is defined as the amount of time that passes between the front bumper of a lead car at a location and the

front bumper of the car immediately behind it. Determining the time headway distribution influences how one would go about modeling vehicle arrival rates. A long-used distribution for the time headway is the exponential distribution, where an associated Poisson distribution would represent the number of arrivals in a fixed length of time t. However, there are many other proposed distributions for time headway and arrivals. Touhbi et al. (2018) study time headways on urban roads in Marrakesh using exponential distributions, shifted exponential distributions, log-normal distributions, shifted log-normal distributions, Pearson III distributions, and Pereto IV distributions. They applied a parametric Kolmogorov-Smirnov (KS) Test as a goodness-of-fit analysis, and find that for their data the Pareto distribution seemed to fit the best.

Al-Ghamdi (2001) performed an analysis of time headway in Riyadh, Saudi Arabia, using many distributions. He states that the question of which distribution best fits the time headway is difficult, and remains so after his work. Still, he claims that the gamma distribution seemed to work well at modeling time headways for arterial (urban) roads in a wide range of traffic flows, an Erlang distribution seemed to model well time headways on high-traffic freeways, and that negative exponential, shifted exponential, and gamma distributions all performed well for modeling time headways on low-to-medium traffic freeways.

Ye and Zhang (2009) also conclude that the type of time headway distribution depends on the traffic flow itself, but also explore if the type of vehicle changes the time headway distribution. The data used in their work includes information on the length of vehicles, which they use to distinguish between smaller and larger vehicles. They find that the vehicle type does have an effect on the type of distribution that best fits the time headway data. They used a KS test as a goodness-of-fit measurement of the distributions.

Li and Chen (2017) present a good literature review of time headway modeling techniques through the years, with a rundown of the many distributions that have been used to model time headway. They also present several car-following models as a means of integrating time headway models into the larger traffic flow theory. Their conclusion is that time headway has grown in interest in the past few decades and that this has led to new techniques and families of models for time headway.

4.2.1 Poisson Process Models

According to Law (2007), the Poisson process is "probably the most commonly used model for the arrival process of customers to a queueing system." Çinlar proved that the number of arrivals over any time length in a Poisson process is a Poisson random variable with parameter λ (Cinlar, 1975) and that interarrival

times for a Poisson process are therefore independent and independently distributed exponential random variables. One advantage that Poisson processes have over arrival processes based on other types of distributions is that Poisson processes do only have a single rate parameter. Other types of distributions are more complex, and necessarily the algorithms for creating arrivals from them are complex as well.

A standard Poisson process has a single rate parameter λ, and therefore only models arrival processes with one expected arrival rate. Any system that experiences multiple busy periods, therefore, requires a model that can handle that. The Nonhomogeneous Poisson Process (NHPP) fulfills this requirement and utilizes a rate *function* $\lambda(t)$ that specifies the arrival rate for any time point t. In practice, $\lambda(t)$ is estimated through various techniques and used to generate arrival (and therefore *interarrival* times) through a process called inversion (Cinlar, 1975).

There have been several methods proposed for estimating $\lambda(t)$. Kao and Chang (1988) form the estimate as a piecewise polynomial rate function and use a process proposed by Bratley et al (2011) to simulate arrival times from the rate function. Klein and Roberts (1984) propose an algorithm that approximates $\lambda(t)$ with a piecewise linear function. They go on to show how an algorithm for generating arrival times can be derived by integrating each of the linear pieces and finding a stochastic expression representing the probability of the next arrival time T_i being contained in the time interval $[t_{i-1}, t_i]$, and then solving this expression for T_i.

Leemis (1991) developed a simple discrete-time algorithm that generates arrival times based on a piecewise-linear approximation $\hat{\Lambda}(t)$ of the cumulative rate function $\Lambda(t)$. $\hat{\Lambda}(t)$ can be based on any number of realizations of the NHPP over the time period under consideration. In this project, a forecast example of a single realization is used. Then the piecewise linear section of $\hat{\Lambda}(t)$ between two time points is given by

$$\hat{\Lambda}(t) = \frac{in}{n+1} + \frac{n(t-t_i)}{(n+1)(t_{i-1}-t_i)} \tag{4.1}$$

for $i = 0, 1, 2, ..., n$ and n is the number of time points in the time interval.

The algorithm produces an exponential random value E_j, where j is the index of the arrival time being computed, and locates the time interval on which it fits into the cumulative rate function, shown in Equation 4.2

$$\hat{\Lambda}(t_{i-1}) \leq E_j \leq \hat{\Lambda}(t_i) \tag{4.2}$$

where i represents which interval in the cumulative rate function E_j fits. The new arrival time T_j is then produced with Equation 4.3.

$$T_j = dt \left(i + \frac{E_j - \hat{\Lambda}(t_{i-1})}{\hat{\Lambda}(t_i) - \hat{\Lambda}(t_{i-1})} \right) \tag{4.3}$$

where dt represents the length of one increment of time. This formula essentially interpolates on the specific piecewise linear segment of the cumulative rate function specified by i and computes the time value T_j associated with that point. The process is illustrated in Figure 4.1.

The procedure continues until $E_j > \hat{\Lambda}(t_n)$ for some j and n represents the last time point in the time window. The set $\{T_1, T_2, \ldots, T_{j-1}\}$ represents all produced arrival times. The interarrival times can be computed by taking all differences between consecutive arrival times. An implementation of this algorithm can be found in the Appendices in Table A.4, and this procedure is used in this project to generate arrivals to the simulation in a manner described below in section 4.3.

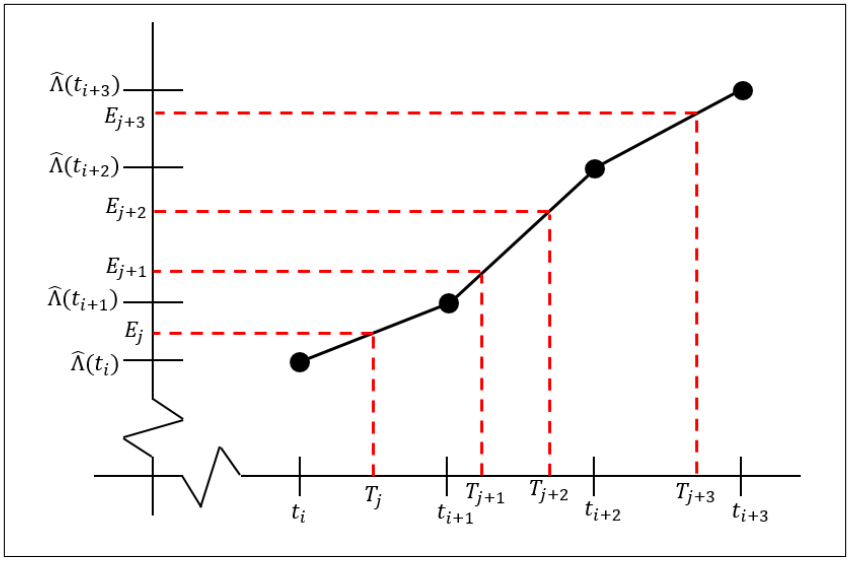


Figure 4.1: Calculating Arrival Times From $\hat{\Lambda}(t)$.

4.2.2 Time Series Models

Naturally, the set of vehicle counts throughout the day is an example of a time series, and as such traditional time series modeling techniques can be applied to the problem of turning these counts into arrival times. Autoregressive integrated moving average (ARIMA) models (Whittle, 1951), (G. E. Box & Jenkins,

1962) have been a classic method of modeling time series for a very long time and have a rich history. Seasonal ARIMA models (G. E. Box et al., 1967) are used to represent time series that show seasonal behavior of some kind.

Barua et al. (2015) use an ARIMA model to represent the arrivals of vehicles to a traffic light. They found that ARIMA models produce more accurate results than Poisson-based approaches, but that the formulation and optimization of ARIMA models is more time-consuming, so there is a tradeoff between paradigms. They mention that they believe their technique could easily be extended to predicting arrivals on freeways.

Yang et al. (2015) compare several different models for time headway distributions including an ARIMA model, a neural network, and a Generalized Additive Model (GAM). They use MAPE as a loss function and determined that the ARIMA model was quite worse at modeling time headway than the other two models, with the neural network performing the best.

The use of SARIMA models is defended in (B. M. Williams & Hoel, 2003) where they explain that using differencing spaced by one week is enough to induce stationarity of the time series, which had previously been asserted by Okutani and Stephanedes (Okutani & Stephanedes, 1984).

Conceptually SARIMA models seem to be an appropriate choice for a time series model since traffic flows tend to be very similar day-to-day, e.g. Tuesdays might look like Mondays, or they might just look like the previous Tuesday. Either way, traffic data shows clear seasonality, which can be seen in Figure 4.2. This image was created by taking the first eight Tuesdays of 2017 and the second eight Tuesdays of 2017 and creating two time series from those values. The specific times of the day are only the 6:00 am to 6:00 pm time frame that was used throughout this project. The figure shows the general seasonality of the data, but also shows that there are days when the traffic data is different. Whether due to traffic accidents, holidays, or special events, the fact that the data is *usually* a mostly regular seasonal time series, but sometimes has entire days of different behavior makes the problem of traffic forecasting and arrival modeling a difficult one to solve.

4.3 Arrival Process Modeling

The physical traffic network being modeled in this work are the northbound lanes of U.S. Highway 101 between E. San Martin Ave. and Tennant Ave. This includes four traffic sensors, but no on-ramps or off-ramps. The first sensor, therefore, acts as the entry point to the simulation, and the fourth sensor acts as

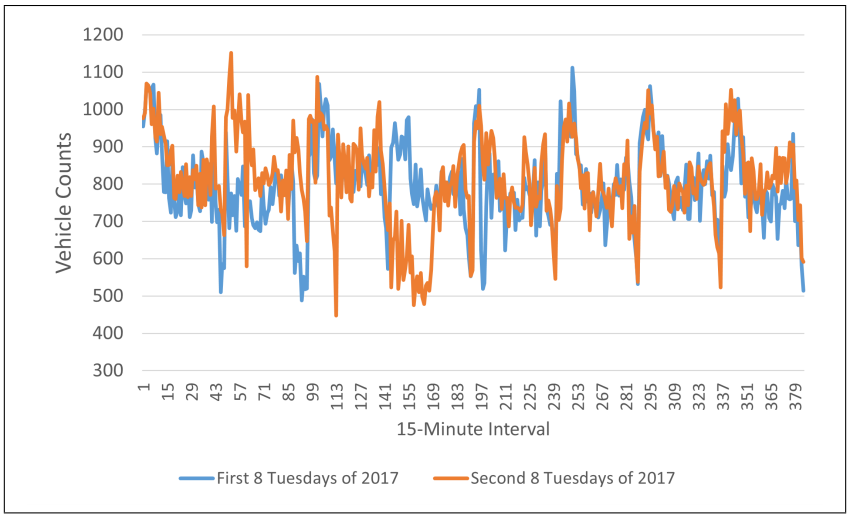


Figure 4.2: Seasonality of Traffic Data Time Series

the exit point. The identification numbers and locations of these traffic sensors are shown in Table 4.1. The traffic network can be seen in Figure 4.3.

Table 4.1: Sensor IDs and Locations

Sensor	ID	GPS (Lat,Long)	Detector Type
Source	409880	(37.097388, -121.60395)	Dual Loop
Sensor 2	402327	(37.103108, -121.60944)	Wireless Magnetometer
Sensor 3	409877	(37.107061, -121.61449)	Dual Loop
Sink	402328	(37.112261, -121.62104)	Wireless Magnetometer

4.3.1 Data

The raw data files come from the Caltrans PeMS system (of Transportation, n.d.), and consist of all recorded vehicle data values for all sensors in a specified zone for each five-minute interval for a specified month. The files are ordered by time code, starting with 12:00 am to 12:05 am on the first day of that month, and ending with 11:55 pm to 12:00 am on the last day of the month. There are 12 zones throughout the state, with zone 4 representing the Bay area and surroundings, which includes San Martin, CA.

The raw data files were processed by first splitting them according to traffic sensor ID, and then organizing the data into 288 row by 365 column matrices for each sensor (there are 288 five-minute intervals in a day). These matrices were

Figure 4.3: U.S. Highway 101 Network

then compressed into 96 rows by 365 columns by consolidating into 15-minute time intervals following the work of Peng (Peng et al., 2018). Finally, the choice was made to focus on Tuesdays only, to provide more consistency to the data, so the matrices were reduced to 96 rows by 52 columns.

The raw traffic data includes a value for the % Observed at a particular station which represents the percentage of data points that were observed versus imputed (where the percent imputed is given by 1 - % Observed) (of Transportation, 2020). PeMS uses four main types of data imputation:

- **Linear regression from neighbors based on local coefficients** - Data gaps are filled using information from the detectors in neighboring lanes at the same location and from detectors in locations immediately upstream and downstream.
- **Linear regression from neighbors based on global coefficients** - When PeMS determines that some detectors never report reasonable data, the system looks at general relationships in the detector data throughout the district to fill in gaps.
- **Temporal medians** - PeMS looks at data values at similar times and days of the week over a long period. The medians of those data values are used to fill gaps.

- **Cluster medians** - PeMS examines data from detectors with similar traffic patterns over a typical week to fill data gaps.

Figure 4.4 shows the vehicle counts for the four sensors in the traffic network on January 2nd, 2018. Sensors 409880 and 409877 seem to agree with each other in the general sense of traffic flow, and sensors 402327 and 402328 also seem to agree, but the two groups do not agree. At times there is a difference of nearly 100 vehicles, which is not simply explained by differences in vehicle movements. And in fact, the two groups are, in a sense, interlaced, meaning the first group is the first and third sensor, while the second group is the second and fourth sensor. The question is which of the two groups to trust? This is where the % Observed value can be used. On this particular day, the % Observed for sensors 409880 and 409877 was 100.0 for the entire day, meaning *all* of the data was recorded as observed and none of the data was imputed. However, for sensors 402327 and 402328, the % Observed value was 0.0 for the entire day, meaning *all* of the data was imputed and none was observed. So the trustworthy numbers come from sensors 409880 and 409877. According to the PeMS manual (of Transportation, 2020), the imputed values for the other sensors were calculated using one of the methods listed above. It seems clear though that these imputations were not accurate, as the difference in vehicle counts at consecutive sensors should not be this large.

It turns out that the % Observed value for sensors 402327 and 402328 is actually 0.0 for all times and all days throughout 2017, 2018, and 2019. The decision was made to include these sensors in the simulated traffic network in a *physical* sense, meaning the simulation would identify those locations as where sensors are located, but their data is not used for comparison purposes. It was also decided to not pursue a self-designed imputation scheme, so these sensors are, as far as the data is concerned, not being used.

A typical highway traffic system will require a dynamic arrival process because most days will see dramatic shifts in vehicle counts throughout the day. Figure 4.5 shows the average traffic flow for all Tuesdays from 2017 through 2019 using 15-minute time intervals, as well as the spread of the data by showing lines dividing the quartiles. The complex nature of the traffic can be seen in the multiple busy periods evident in the shape of the graph.

4.3.2 Arrivals

The simulation system needs a way to convert vehicle count data into individual vehicle arrival times so that the simulation source can realistically generate the vehicles. This process first requires a forecasting technique to generate counts

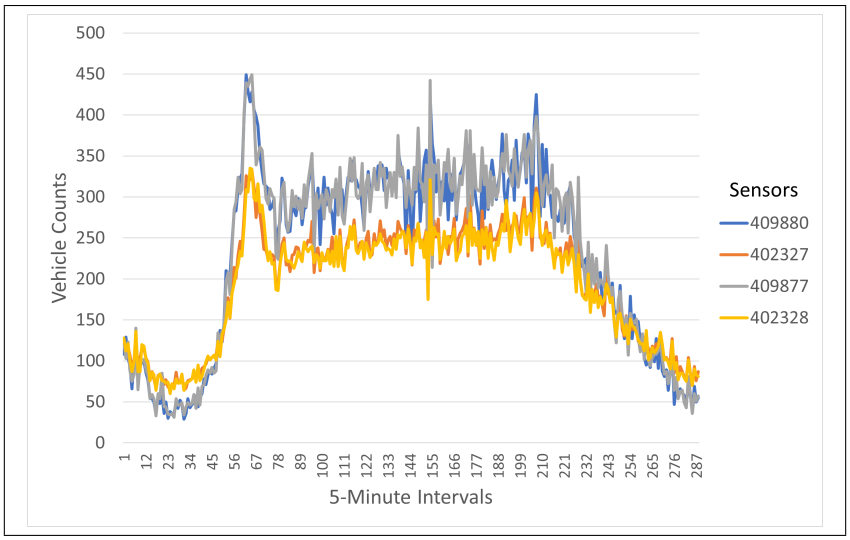


Figure 4.4: Sensor Data for January 2nd, 2018

for the time interval of interest because only past data can be used, and then vehicle arrival times are generated from these forecasts. One might suggest that these forecasting techniques could be used to estimate the vehicle counts at all sensors of the system, and forego the simulation entirely. However, the accurate estimation of vehicle counts is not the only reason to create the simulation system. A microscopic traffic simulation system models individual vehicles and attempts to capture fine details of their behavior, which gives information about traffic flow well beyond simple statistics like vehicle counts, vehicle speeds, etc. Indeed, the simulation system allows answering what-if questions such as how a road might be affected by the addition of a lane, or if an intersection is changed from a four-way stop to a roundabout. The simulation system can only be built, and these types of questions answered, if a suitable arrival model is created.

In this project, the forecast counts are used to generate the arrival times using the process of Leemis (L. M. Leemis, 1991) described above. Once the arrival times have been computed the inter-arrival times are calculated and used by the simulation model as a time headway value at the simulation source. The source creates a vehicle, then waits for the duration of the next inter-arrival time on the list, and then creates the next vehicle. As the vehicles are created they enter the highway, and at that point, their movement is controlled by the car-following model until they exit the system.

There is some question as to whether or not the vehicle counts generated by an NHPP really belong to the same distribution as the real vehicle counts. This

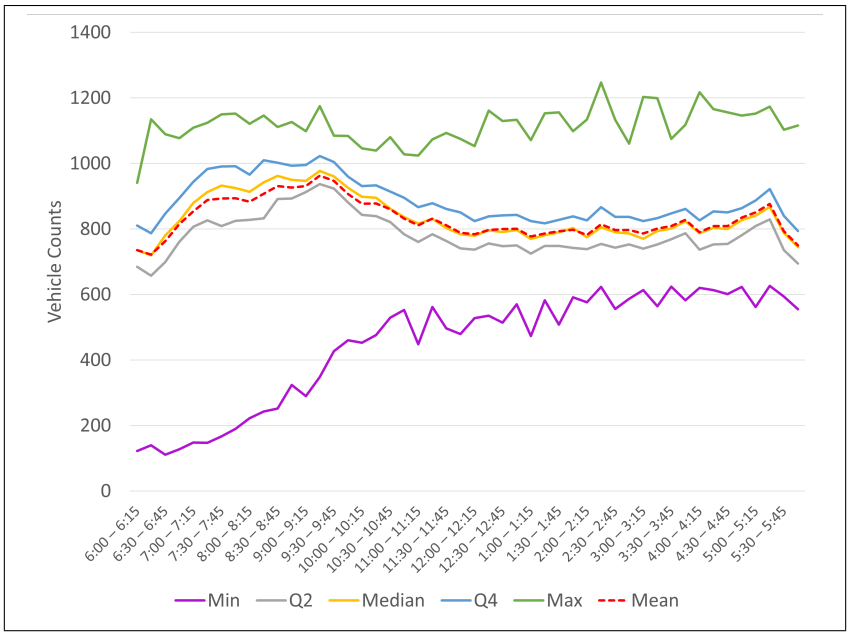


Figure 4.5: Spread of vehicle count data

question can generally be answered by using a two-sided Kolmogorov-Smirnov Test (Kolmogorov, 1933), (Smirnov, 1948). The test calculates the maximum distance between the two cumulative distributions and then calculates a p-value from the sample sizes and the distance. Figure 4.6 shows a sample comparison of a distribution of counts generated by an NHPP with a distribution of actual vehicle counts. The p-value suggests that the NHPP method is capable of representing the actual vehicle counts for the data in this project.

Five methods of forecasting the counts are presented below. Two of the methods use a seasonal autoregressive integrated moving average model (SARIMA) (G. E. Box et al., 1967) which is a standard time series forecasting technique that extends the idea of ARIMA models. ARIMA methods (Whittle, 1951), (G. E. Box & Jenkins, 1962) have a long history of use for time series forecasting and remain an extremely popular forecasting method. The other three forecasting methods, which are defined below, are much simpler and require very little training time. The five forecasting methods are defined below.

Once the forecasts are made, the arrival times are generated using an NHPP, which is a form of Poisson process where there is an arrival rate function $\lambda(t)$ instead of a single arrival rate λ. In this project, the rate function is defined

$$\lambda(t) = \hat{c}_t \tag{4.4}$$

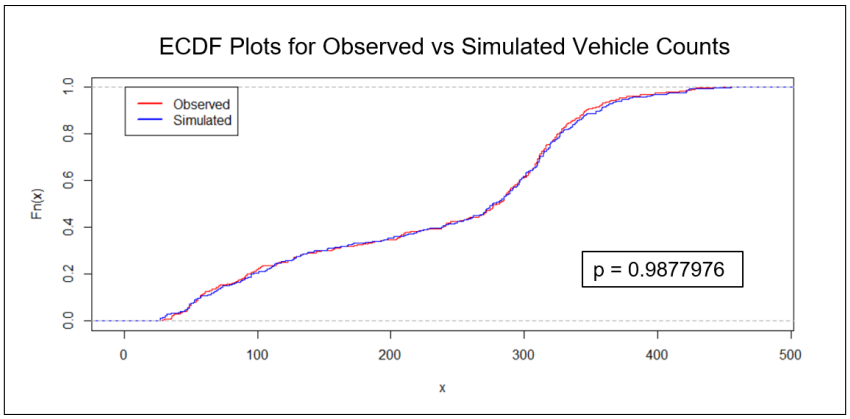


Figure 4.6: KS Test between the observed vehicle count distribution and the distribution of counts generated by an NHPP

where $\hat{c}_t$ is the forecast vehicle count for the time interval represented by t. The NHPP is created using the procedure described by Leemis (1991).

4.4 Offline Methods

Offline forecasting methods only use data from past days and are trained ahead of the day being forecasted. The first offline method discussed uses a SARIMA time series model to create the forecasts.

SARIMA models require order and differencing parameters for both the non-seasonal and seasonal components of the model. In this project, these parameters were chosen using a brute-force optimization technique. The parameter set with the best average accuracy value was chosen for the model, which yielded a SARIMA(1,0,0)(1,1,2) model. The time series consists of some number n of past days' data over the entire 12-hour period and is used to create a forecast for the next 12 hours, where the count at each time point is the value $\hat{c}_t$ used to generate the NHPP. A value of $n = 6$ was determined to be optimal by the optimization procedure.

The second offline method, which could be called a Historical Average Method (HAM), calculates a simple average of n previous Tuesdays and uses that value as the predicted vehicle count $\hat{c}_t$ for that time interval on the day being forecasted. In Equation 4.5, and for Equations 4.6, 4.7, 4.8, and 4.9 below, j represents the day index within the data matrix. This calculation is made for all time intervals to give a vector of vehicle counts for the day, and this is then used as the basis for the rate function $\lambda(t)$.

$$\hat{c}_{t,j} = \frac{1}{n} \sum_{i=j-n}^{j-1} c_{t,i} \tag{4.5}$$

4.5 Online Methods

Online forecasting techniques assume that new data is available as time passes, which can be incorporated into the model to improve accuracy. The methods below all assume that for each 15-minute interval, the count of vehicles for the *previous* 15-minute interval is already known in the system.

4.5.1 Online SARIMA Model

The first online method is a SARIMA model where each time series consists of $96n$ time points where n again represents the number of days in the past to use. Once again the order and differencing parameters were determined using a brute-force optimization technique. These parameters are the same throughout the 12-hour period being examined here, but the model is retrained for each 15-minute interval with the new data value that is available. The data used for the project came from two different Tuesdays so the optimization of the SARIMA model was conducted twice. The SARIMA model found for the first day of data is defined as a SARIMA(2,0,2)(1,1,1) model, and the model for the second day is defined as a SARIMA(1,0,0)(2,1,1) model.

The SARIMA models are constructed with the time series data and then used to predict the estimated vehicle count one step of 15 minutes ahead. The prediction is then used as the rate value prediction $\hat{c}_{t,j}$ for the construction of a Nonhomogeneous Poisson Process.

4.5.2 Rate of Change Models

Both of the other methods use a simple average rate of change of past data for each time interval that can then be applied to the new data as it arrives.

The first of these methods uses a ratio-based calculation, presented in Equations 4.6 and 4.7. This will yield a percentage of vehicles either gained or lost in that time interval for each of the previous n weeks. Take the average of these percentages, and apply them, starting at the last real data value available for earlier that same day to construct a new forecasted count of vehicles on which to base the arrival rate function.

$$r_t = \frac{1}{n} \sum_{i=j-n}^{j-1} \frac{c_{t,i}}{c_{t-1,i}} \tag{4.6}$$

$$\hat{c}_{t,j} = c_{t-1,j} \cdot r_t \tag{4.7}$$

The difference-based approach is similar to the ratio-based approach, however, instead of taking the ratio of the counts, differences are used. Equations 4.8 and 4.9 show the calculations.

$$d_t = \frac{1}{n} \sum_{i=j-n}^{j-1} [c_{t,i} - c_{t-1,i}] \tag{4.8}$$

$$\hat{c}_{t,j} = c_{t-1,j} + d_t \tag{4.9}$$

To increase the flexibility of these models, a weighted average of the new methods with the offline method can be utilized.

4.6 Arrival Process Comparisons

Each of the methods uses some number n of past days' data. Once again using a brute-force optimization procedure, it was determined that all of the models optimized their accuracy when $n = 6$.

The accuracy metric used throughout this work is the symmetric mean absolute percent error (sMAPE) and is defined in the Equation 4.10.

$$\text{sMAPE} = \frac{200}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{|F_t| + |A_t|} \tag{4.10}$$

where A_t is the actual value and F_t is the forecast value. sMAPE is a commonly used accuracy metric that was chosen as one of the accuracy metrics in both the M3 and M4 Competitions (Makridakis et al., 2020) because it is "scale-independent and intuitive to understand". With traffic flow changing throughout the day, it is important to have a relative error metric. An absolute error metric can be difficult to interpret when the target values change so much over the course of the day.

The performance of the methods was tested for 150 Tuesdays from 2017 through 2019. For each day the arrival forecasts were produced using each of the procedures described above for each fifteen-minute interval from 6:00 am until 6:00 pm. This time period encompasses most of the heavy traffic parts

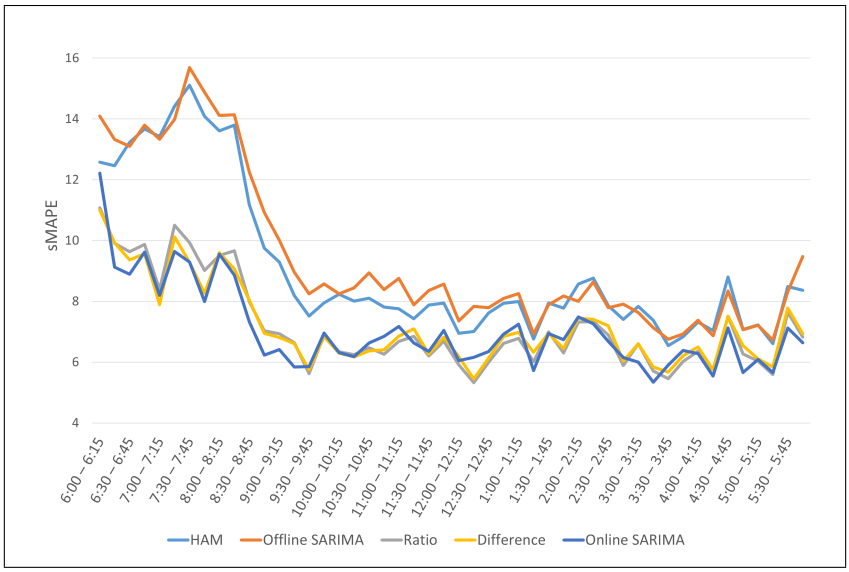


Figure 4.7: Comparison of Arrival Methods Before Application of NHPPs

of the day. Each of the forecasts was compared to the actual data and sMAPE values were calculated.

Figure 4.7 shows a comparison of the accuracy of the arrival techniques in terms of forecasting the actual counts. As can be seen, the offline approaches are inferior to the online methods. Figure 4.8 compares the various methods once their forecasts have been used to generate arrival times for the simulation using an NHPP. Figures 4.7 and 4.8 look very similar but there is a slight accuracy loss through the process of turning the forecasts into arrival times, and then back into vehicle counts.

A comparison of the average performance of the different arrival process models is given in Table 4.2. Across several statistics, it can be seen that the online models perform better than the offline models. The worst performer was the offline SARIMA model, while the best performer was the online SARIMA model, but with only a slight improvement over the other two online techniques. With the amount of time it can take to optimize the various parameters of a SARIMA model, and recognizing the desire for time-efficient forecasting systems, the fact that the two proposed online methods are actually quite competitive with the SARIMA model, and execute very quickly due to their simplicity, it is reasonable to investigate these methods further.

The process of using online data implies a form of ongoing calibration where new data is used to keep the simulation model informed. An event, such

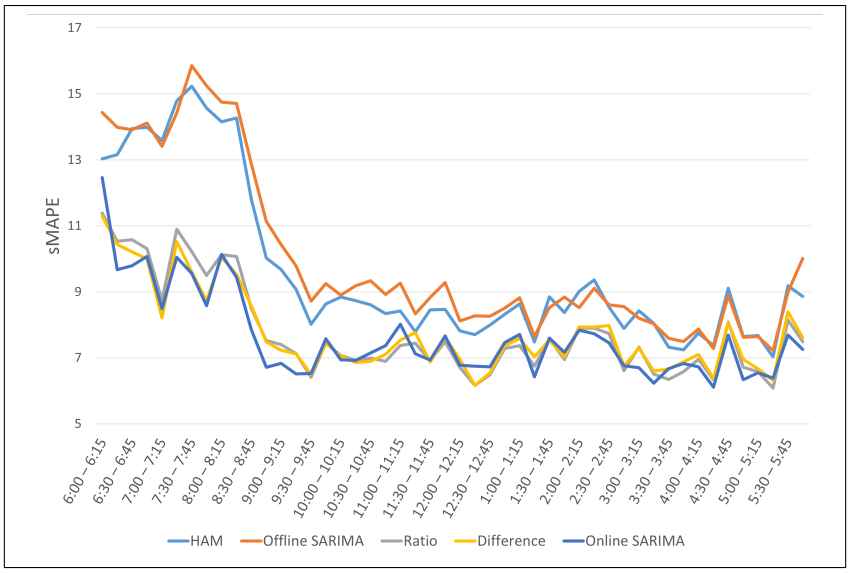


Figure 4.8: Comparison of Arrival Methods After Application of NHPPs

Table 4.2: Arrival Model Comparisons

sMAPE	HAM	Offline SARIMA	Ratio	Difference	Online SARIMA
Mean	9.60948	9.95241	7.75968	7.75724	7.62440
Median	8.6207	8.89527	7.37368	7.40042	7.21305
Min	7.02541	7.22076	6.08701	6.16568	6.11235
Max	15.23084	15.85398	11.38149	11.29758	12.46072

as an accident, can greatly change traffic from what drivers are accustomed to, and methods that incorporate the latest data will necessarily have a better chance of capturing these changes and maintaining the ability to produce accurate results in these instances.

4.7 Conclusions

A traffic simulation needs a time headway model to introduce vehicles to the roads at realistic times that closely approximate the true traffic flow. Vehicle counts from traffic sensors can be used to create an arrival model by assuming the counts are sampled from a nonhomogeneous Poisson distribution and then using inversion to design an exponential arrival time model. This process requires time series forecasting since the goal of the simulation is to predict future traffic flows.

Several different techniques for generating these vehicle count forecasts were compared including three methods that utilize fresh data to create an online arrival model. Each method uses a different process to create a predicted rate function for the NHPP, which is then constructed in the technique provided by Leemis (L. M. Leemis, 1991). Two of the online methods are quite simple in that they construct the rate function $\lambda(t)$ using very simple calculations. However, these methods show comparable results to a more complex and time-intensive SARIMA model, which in an online system might require more time to construct than is available. The amount of time needed to construct the ratio-based and difference-based rate function predictions is negligible, whereas the SARIMA model would potentially take a time-consuming process of optimizing the internal parameters of its construction.

A worthwhile direction for future work is to explore distributions besides the exponential and Poisson distributions, or to use a different process altogether, such as a neural network. Other forecasting methods could also be utilized to generate more accurate forecasts before the application of the NHPP.

Chapter 5

Car-Following Models

5.1 Introduction

There have been many car-following models suggested in the literature such as (Gazis et al., 1959), (Gazis et al., 1961), (Chandler et al., 1958), (Herman et al., 1959), (Kometani & Sasaki, 1961), (Pipes, 1953), (Newell, 1961) to highlight some of the early contributions. Car-following models that are in wide use today are Gipps' Model (Gipps, 1981) and the Intelligent Driver Model (IDM) (Treiber et al., 2000). A comprehensive genealogy of microscopic models is presented in (van Wageningen-Kessels et al., 2015). A detailed comparison of the IDM and Gipps' Model was also presented (Matcha et al., 2021) whereby the two models are compared across different formations of traffic flows. They determined that the IDM is more appropriate for homogeneous types of traffic, but that Gipps' Model has better performance in more heavily mixed types of traffic. Traffic on a limited access highway is likely to be fairly homogeneous with very little variation in the parameters of individual vehicles.

The purpose of a car-following model is to update a vehicle's position, velocity, and possibly acceleration based on the same values of the vehicle immediately in front of the vehicle. A key trait of any car-following model is that the velocity of a vehicle should asymptotically approach its desired speed in free traffic. How quickly it approaches its desired speed results from the intricacies of the model itself.

Figure 5.1 shows the concept of distance headway between two vehicles. This distance is very important in many car-following models. The concept of space headway is defined in Section 5.6.5.

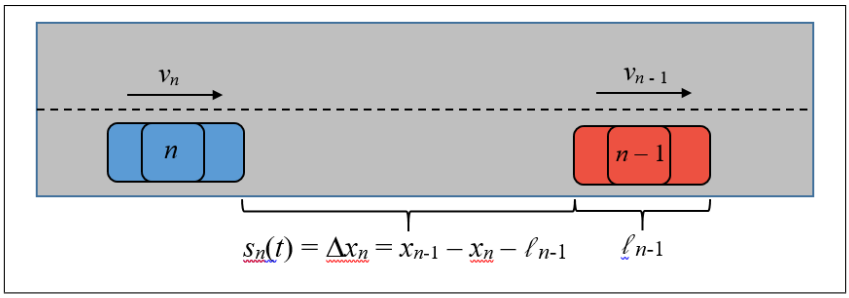


Figure 5.1: Distance Headway

5.2 GHR Models

One of the earliest models is due to Gazis et al. (1961) and is known as a stimulus-response model. It defined the new acceleration of the nth vehicle as

$$a_n = \lambda[v_{n-1} - v_n] \tag{5.1}$$

where λ is a *sensitivity* parameter that better controls the resulting value. The idea is straightforward in that a vehicle's acceleration/deceleration should depend on the difference between its velocity and the velocity of the vehicle in front of it. If a vehicle is driving faster than the vehicle in front of it, then the velocity difference will be negative, resulting in deceleration. If the vehicle is driving slower than its predecessor, then the difference will be positive and the formula will output an acceleration.

The sensitivity value need not be a constant but may be determined by a more complex function. Early on it was proposed that the sensitivity should depend on the distance between the vehicles. Indeed, if traffic is quite sparse, and the distance between vehicles is very large, it would make sense that the car-following idea can almost be disregarded in favor of free-driving behavior. Two early changes to the sensitivity are given below in Equations 5.2 (Gazis et al., 1959) and 5.3 (Edie & Foote, 1961)

$$\lambda = \lambda_1/[x_{n-1} - x_n] \tag{5.2}$$

$$\lambda = \lambda_2 v_n/[x_{n-1} - x_n]^2 \tag{5.3}$$

The general form of the model is now represented as

$$a_n = \lambda v_n^m \frac{(v_{n-1} - v_n)}{(x_{n-1} - x_n)^l} \tag{5.4}$$

for parameter λ and exponent parameters m and l which would be used to fit the general model to the data.

In Equation 5.2 the sensitivity is inversely proportional to the distance between the vehicles. This distance can only get small if the following vehicle is driving *faster* than the vehicle in front, which, as discussed earlier, would result in a deceleration. Combining this with the inverse proportionality of the sensitivity relationship, this situation would result in "slamming on the brakes" as it's known. If the distance between the vehicles is large, then the sensitivity value is small, and its effect on the acceleration computation is therefore small, which makes sense at great distances.

Equation 5.3 represents the sensitivity as a mixed direct and inverse relationship, where the sensitivity is directly proportional to the vehicle's velocity and inversely proportional to the distance between the vehicles. This means that the impact of the relative distance between the vehicles is scaled with the velocity of the follower vehicle so that the braking force will increase polynomially as the speed increases.

5.3 Pipes' Model

Another early model, known as a safe-distance model, was proposed by Pipes (1953) and was defined as in Equation 5.5.

$$x_{n-1} = x_n + d + Tv_n + l_{n-1}^{veh} \tag{5.5}$$

where x_{n-1} is the position of the front bumper of the lead vehicle, x_n is the position of the front bumper of the following vehicle, T is the time headway between the vehicles, d is the distance between the vehicles, and l_{n-1}^{veh} is the *length* of the lead vehicle. This formula forms a relationship between the position of the lead vehicle and its follower, based on the velocity of the follower and the distance between the two. At first, it may not make sense because the reality is that the front bumper of the lead vehicle is specifically equal to $x_n + d + l_{n-1}^{veh}$, but the idea here is to introduce a term with the velocity of the follower vehicle that would ensure the distance between the vehicle would always seek to be *safer* than that. Pipes (1953) based his model on a "rule of thumb" given in the state of California Motor Vehicle Code:

> "A good rule for following another vehicle at a safe distance is to allow yourself the length of a car (about fifteen feet) for every ten miles per hour you are traveling."

The term Tv_n is the product of time headway and the current velocity of the vehicle. This is equal to the distance the car will travel in the amount of time that is equal to the time headway.

5.4 Gipps' Model

A very important safe-distance model is Gipps' Model (Gipps, 1981), which is based on the assumption that drivers will try to maximize their velocities as safely and as quickly as possible.

Gipps' Model computes the vehicle's new velocity $v_n(t+\tau)$ instead of the new acceleration. This is one of the major differences between Gipps' Model and the IDM. Another key difference is in the model parameters which are given below in Table 5.1. The equations for computing the new velocity are given in Equations 5.6, 5.7, and 5.8.

$$v_n(t+\tau) = \min\left[v_F(t+\tau), v_C(t+\tau)\right] \tag{5.6}$$

where

$$v_F(t+\tau) = v_n + 2.5a_n\tau(1 - v_n(t)/V_0)(0.025 + v_n(t)/V_0)^{1/2} \tag{5.7}$$

is the *Free driving* velocity and the *Car-following* velocity is given by

$$v_C(t+\tau) = b_n\tau + \sqrt{b_n^2\tau^2 - b_n[2\Delta x_n - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}]} \tag{5.8}$$

where $\Delta x_n = x_{n-1}(t) - s_{n-1} - x_n(t)$, with s_{n-1} representing the length of the $(n-1)$st vehicle plus a safe distance s_0 that the vehicle always wants to allow between then and their leading vehicle, and $\hat{b}$ representing an estimated deceleration value for the $(n-1)$st vehicle (since the nth driver cannot reasonably know this value).

The derivation of the car-following portion of Gipps' Model requires the classic physics equations below.

Table 5.1: Gipps' Model Values

Symbol	Definition
n	Current vehicle
V_0	Desired velocity of nth vehicle
l_n	Length of nth vehicle
$x_n(t)$	Location of nth vehicle
$v_n(t)$	Current velocity of nth vehicle
a_n	Max acceleration of nth vehicle
b_n	Max deceleration of nth vehicle
$\hat{b}$	Estimate of the (n-1)st vehicle's deceleration
τ	Reaction Time of all vehicles

$$d(t) = \frac{1}{2}at^2 + v_0 t \tag{5.9}$$

$$v(t) = at + v_0 \tag{5.10}$$

Assume a lead car ℓ is traveling at constant velocity v_ℓ and begins to apply the constant deceleration b_ℓ at time t. Using Equations 5.9 and 5.10 in t seconds the distance d the lead vehicle will travel while braking is

$$d(t) = v_\ell t - \frac{1}{2}b_\ell t^2 \tag{5.11}$$

Differentiating this gives $v(t) = v_\ell - b_\ell t$, which is the formula for the velocity after t seconds, so setting this equal to zero will yield the amount of time needed for the lead vehicle to come to a stop.

$$\begin{aligned} 0 &= v_\ell - b_\ell t \\ t &= \frac{v_\ell}{b_\ell} \end{aligned} \tag{5.12}$$

Substituting this back into Equation 5.11 gives the *distance* needed for the lead vehicle to come to a complete stop:

$$\begin{aligned} d(v_\ell/b_\ell) &= v_\ell \left(\frac{v_\ell}{b_\ell}\right) - \frac{1}{2}b_\ell \left(\frac{v_\ell}{b_\ell}\right)^2 \\ &= \frac{v_\ell^2}{2b_\ell} \end{aligned} \tag{5.13}$$

Now considering the follower car, there is a reaction time τ that is required once the lead vehicle first applies the brakes before the follower begins to apply their brakes. Assuming the follower vehicle has constant acceleration a_f (which could in fact already be a deceleration), and initial velocity v_0, using Equation 5.10 and dropping the functional notation yields

$$a_f t = v_f - v_0 \tag{5.14}$$

and substituting this back into Equation 5.9 gives that the distance the follower vehicle will travel during the reaction time is

$$\begin{aligned} d_f(t+\tau) &= \frac{1}{2}(v_f - v_0)\tau + v_0\tau \\ &= \frac{(v_0 + v_f)\tau}{2} \end{aligned} \tag{5.15}$$

After the reaction time τ, the follower begins to apply deceleration b_f at time $t + \tau$, and the distance covered while decelerating to a complete stop is

$$d = \frac{v_f^2}{2b_f} \tag{5.16}$$

which gives the total stopping distance of the follower vehicle from the moment the *lead* vehicle began to apply its brakes as

$$d_f = \frac{(v_0 + v_f)\tau}{2} + \frac{v_f^2}{2b_f} \tag{5.17}$$

Figure 5.2 shows the stopping distances for the lead vehicle (red) and follower vehicle (blue). The image is dynamic meaning the vehicles on the left are their positions at time t while the positions on the right are at a later time. The value s_0 is considered the minimum safe gap that the follower wants to allow while braking.

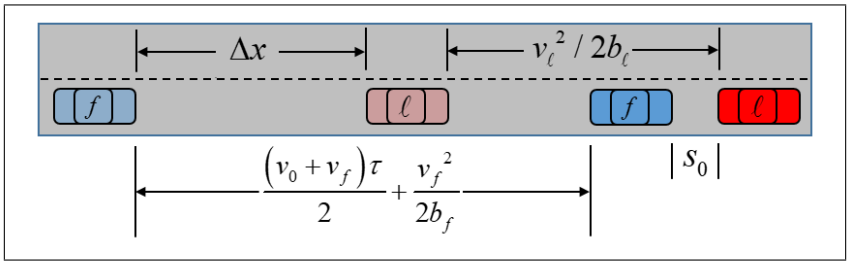


Figure 5.2: Stopping Distances

Gipps' Model is a safe-distance model, so an additional term for the distance traveled during the time needed to actually hit the brake is included. This value is $v_f(\tau/2)$. Assuming that the length of the vehicles is the same (not always reasonable, but one made here), then the following relationship represents the minimum gap s_0

$$s_0 = \Delta x + \frac{v_\ell^2}{2b_\ell} - \left(\frac{(v_0 + v_f)\tau}{2} + \frac{v_f^2}{2b_f} + \frac{v_f \tau}{2} \right) \tag{5.18}$$

Multiplying through by $2b_f$ yields

$$2b_f s_0 = 2b_f \Delta x + v_\ell^2 \left(\frac{b_f}{b_\ell} \right) - b_f v_0 \tau - 2b_f \tau v_f - v_f^2 \tag{5.19}$$

and rearranging gives a quadratic equation in v_f

$$v_f^2 + (2b_f\tau)\, v_f + \left[b_f v_0 \tau - v_\ell^2 \left(\frac{b_f}{b_\ell} \right) - 2b_f \left(\Delta x - s_0 \right) \right] = 0 \tag{5.20}$$

In the general case, v_f is considered the safe velocity for a follower, and solving Equation 5.20 yields the formula in Equation 5.8. Some slight alterations are made, mostly in notation:

- $v_f \rightarrow v_n(t + \tau)$
- $v_\ell \rightarrow v_{n-1}(t)$
- $v_0 \rightarrow v_n(t)$
- $b_\ell \rightarrow \hat{b}$
- $b_f \rightarrow b_n$

and Δx_n being redefined to include s_0 as explained above.

5.5 Intelligent Driver Model

A more recent stimulus-response model is the Intelligent Driver Model (IDM). The details of the IDM are presented in Equations 5.21 and 5.22, and 5.23, and in Table 5.2.

$$\dot{v}_n(t + \tau) = a \left(1 - \left(\frac{v_n(t)}{V_0} \right)^\delta - \left(\frac{s^*(v_n(t), \Delta v_n(t))}{s_n(t)} \right)^2 \right) \tag{5.21}$$

where

$$s^*(v_n(t), \Delta v_n(t)) = s_0 + v_n(t)T + \frac{v_n(t)\Delta v_n(t)}{2\sqrt{ab}} \tag{5.22}$$

and

$$s_n(t) = x_{n-1}(t) - x_n(t) - l_{n-1} \tag{5.23}$$

Table 5.2: IDM Parameters

Parameter	Definition
n	Current vehicle
τ	Reaction time
$v_n(t)$	Current velocity of nth vehicle
$x_n(t)$	Current position of the nth vehicle
V_0	Desired velocity of nth vehicle
s_0	Minimum distance headway
T	Minimum time headway
a	Maximum comfortable acceleration
b	Maximum comfortable deceleration
$s_n(t)$	Current distance headway of nth vehicle
l_n	Length of the nth vehicle
δ	IDM tuning parameter

A visual comparison of the IDM and Gipps' Model is given in Figure 5.3. The vertical axis shows the distance traveled, measured in meters, and the horizontal axis is time, measured in seconds. In each picture, the lead vehicle is shown in red and was artificially given instructions to slow down at specific times, so that the car-following behavior of the models can be seen. The dark bands seen in the graphs result from the upstream traffic slowing down to avoid colliding with their predecessor vehicles. The two scenarios had identical initial conditions, and some interesting conclusions can be drawn from these images. First, it is clear that Gipps' Model increases the lead vehicle's speed toward its desired speed more slowly than the IDM, or in other words, the IDM leads to higher accelerations than Gipps' Model. Second, the IDM responds more severely to braking behavior than does Gipps' Model, which can be seen from the much darker bands evident in the IDM figure. Both of these facts lead to the conclusion that the IDM can result in both more extreme accelerations *and* more extreme decelerations (braking) than does Gipps' Model.

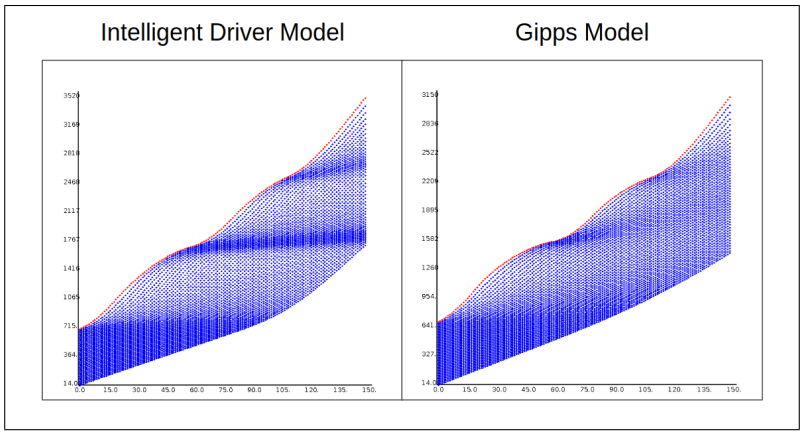


Figure 5.3: IDM vs. Gipps' Model

5.5.1 Velocity

V_n is defined to be the driver's desired velocity, and the car-following model will attempt to safely increase the velocity of the car asymptotically until it equals v_0. This progress will of course be controlled by the presence of a leading vehicle. In such instances, the car-following model will maintain a distance behind the vehicle, but will still factor in the desired velocity to the calculation. If there is no leading vehicle, or if the distance between the vehicles (defined as space headway, which is explained below) is large, then the vehicle's velocity can eventually make it to v_0 and then stay there.

In Gipps' model, if $v_n(t)$ is equal to V_0 then Equation 5.7 will remain $v_n(t)$ and since Equation 5.6 takes the minimum, of Equations 5.7 and 5.8, there's no way for the velocity to continue increasing.

In the IDM, if $v_n(t)$ is equal to V_0 then the term $v_n(t)/V_0$ is equal to one and Equation 5.21 can be at most zero, which would prevent the velocity from going higher than V_0.

5.6 Position

The current position of the nth vehicle is maintained by the value $x_n(t)$ and is mainly used to derive the relative distances between following and leading vehicles. The position values of the two vehicles, along with the length of the leading vehicle, are used to compute the space headway described below in Section 5.6.5.

5.6.1 Acceleration

Acceleration in both the IDM and Gipps' Model is defined as the maximum comfortable acceleration that a driver would prefer to use to speed up their vehicle. The IDM formula can occasionally result in larger acceleration values, but the model is structured to try to keep the new acceleration under the maximum acceleration.

An acceleration of 1 g is equal to 9.8 m/s^2, which might seem like too fast of an acceleration for a regular vehicle, but the Tesla Plaid is reported to have an acceleration up to 1.2 g (Richard, 2021).

5.6.2 Deceleration

Deceleration in the models is defined as the maximum comfortable deceleration that a driver would prefer to use when braking as a follower vehicle. Deceleration is usually considered a negative value, and is reported as such in this paper, but is implicitly used as a positive value in the IDM formulation. Equation 5.22 contains the calculation $\sqrt{ab}$, which can be read mathematically as the geometric mean of the maximum acceleration and maximum deceleration. If interpreted this way, it seems the resulting value is essentially the average amount of acceleration-type force that the driver wants to put their vehicle under.

5.6.3 Reaction Time

Reaction time determines how long it takes for the effects of the car-following model to be assigned but essentially corresponds to how long it takes a driver to respond to the behavior of the car in front of them. Within the workings of the car-following model, it is the amount of time, after calculation of the new acceleration, that the vehicle will take on that value as its new acceleration. In the sense of a discrete event traffic simulator, this is the amount of time increment that would be put on the system schedule for the vehicle under consideration, and the vehicle would next be updated after that amount of time has passed in the simulation system.

5.6.4 Time Headway

Time headway is defined as the amount of time (in seconds) it takes for a following vehicle to reach the same physical location that the leading vehicle just was. In fact, many driving handbooks mention the "three-second-rule" that says a safe following distance, 45 mph and under, is to allow three seconds of time

between the following car and the lead car (Georgia-DDS, 2022), (Missouri-DOR, 2022), (California-DMV, 2022). In the case of the IDM, this parameter is the *preferred* time headway value. Its function is to help control the braking of a vehicle as it approaches the car in front of it. If the preferred time headway is large then the follower vehicle will begin to brake sooner than a car with a smaller time headway value. Time headway is also related to arrival modeling and in fact, some arrival models explicitly define the arrival rate in terms of time headway.

5.6.5 Space Headway

Space headway is defined as the physical distance between the front bumpers of the lead car and the following car, and so includes the length of the leading car. The parameter in the IDM is defined as a vehicle's *preferred* space headway value when following closely. Space headway also helps control the braking behavior of a follower vehicle, and is a similar idea to time headway, however, space headway is more of an absolute value in that it does not depend on how fast the vehicles are moving, were time headway does depend on the vehicles' velocities. A larger space headway value would lead to a vehicle braking earlier than a vehicle with a smaller space headway value when approaching a lead vehicle.

5.6.6 Parameter δ

The delta parameter is purely a model-tuning parameter and has no inherent units or correspondence to the real world. The authors mention that in their work it is usually set to 4.0 (Treiber et al., 2000).

Chapter 6

Calibration Techniques

The overall problem of building a model for a real-world system is often referred to as model *estimation* and includes processes such as determining the type of model to use, the structure of the chosen model and determining the values of internal parameters. The work in this chapter touches on the first two elements of this process but focuses on the third in the largest part. Two car-following models were chosen to represent the vehicle behavior in the simulation system, and several different arrival models were chosen to represent the time headways of vehicles as they enter the simulation, giving eight total *different* models to compare. However, the majority of this chapter is devoted to determining the parameters for these models using techniques from parameter estimation and calibration.

There are multiple forms of modeling errors when attempting to model a real-world system. One error source is model error, meaning the model chosen, and its construction, are not fully capable of representing the real-world system that it was meant to. Another source of modeling error is parameter error, meaning that the model parameters cannot be determined to a degree that allows the model to fit the real-world system perfectly. This second source of error is the focus of the work below, specifically the task of estimating parameter values for the purpose of calibrating traffic models.

A model of a real-world system will contain some number of parameters. To use the model for any worthwhile purpose these parameters must be determined as best as possible so that the model does come close to the real-world system it is built to represent. Defined below are the two principle paradigms for achieving this. The first, parameter estimation, is an analytically-driven process for finding the best parameters for the model in a kind of complete sense. That is, the idea is that there are either exact parameters that work all the time for the model, or there is some distribution or collection of distributions to which the parameters

belong, and once those have been determined, sampling that distribution is all that is needed. So the parameter estimation problem is being solved for the entire model over all possible data values.

The second paradigm is calibration, which is the idea that the parameters can be set to reasonable values, or sometimes even random values, and with data, the values can be incrementally changed until the model output and the real output are as close as possible. This can only be done with a subset of the data, and so parameters found with this process might not be valid with other subsets of the data.

6.1 Parameter Estimation

Traffic flow networks are real-world dynamic systems that have a large impact on society and are therefore of particular interest to be modeled. Real-world systems can contain any number of measurable characteristics, and traffic is no different. Vehicle speeds, vehicle counts, traffic congestion, and lane occupancy are just a few of the traits traffic modelers are interested in measuring. The data can be thought of as the output of the system and can be represented by a vector of values $\mathbf{x}$. In most real-world systems however the output data $\mathbf{x}$ will be very noisy, and so the measurements are not assumed to be perfect, and therefore the output data of the system is represented as

$$\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta}) + \boldsymbol{\epsilon} \tag{6.1}$$

where $\boldsymbol{\epsilon}$ represents the noise, and $\mathbf{y}$ represents the actual system output, f represents the model, and $\boldsymbol{\theta} = (\theta_1, ..., \theta_m)$ represents the vector of model parameters, and m represents the number of parameters.

The parameter vector $\boldsymbol{\theta}$ needs to be optimized using known input and output system data so that the model can get as close as possible to the real system. There are multiple techniques for optimizing the parameter vector. Two traditional techniques, Maximum Likelihood Estimation (MLE) and Least Squares Estimation (LSE) are shown in the appendix in terms of time series parameter estimation.

6.1.1 Calibration

There have been many efforts made to determine the distribution of traffic flow, but it appears that this is highly dependent on the geographic area, the type of road, and the types of vehicles in the traffic system. The distribution also likely depends on the time of day, the day of the week, and even the month under

consideration. These facts make the traditional analytical methods of parameter estimation very difficult to carry out. For this reason, it was decided that a calibration process using a subset of traffic data be used to determine the model parameters for this traffic simulation. The methodology of that process and its results are discussed below in section 6.2. This technique requires choosing a loss function and the choice of an optimization procedure.

We can define the process as in Equation 6.2 below.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \mathcal{L}(f(\mathbf{x}; \boldsymbol{\theta}), \mathbf{y}) \tag{6.2}$$

where $\hat{\boldsymbol{\theta}}$ is the vector of optimized parameters, f represents the model itself, $\mathbf{x}$ is the vector of input data values for the model, and $\mathbf{y}$ is the vector of output data values. $\mathcal{L}$ represents the loss function, and there are many to choose from. Common examples of loss functions are the Mean Squared Error (MSE), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), Mean Absolute Percent Error (MAPE), and Symmetric Mean Absolute Percent Error (sMAPE).

6.2 Calibration of the Traffic Model

The simulation system built for this project can utilize any car-following model. The two models selected for this work are the Intelligent Driver Model (IDM) and Gipps' Model. The model parameters that have been chosen for the calibration of the IDM are given in Table 6.1 along with their domains. These domains have been based partially using previous calibration efforts (Kurtc & Treiber, 2016), (Rahman et al., 2020), and partially on the modeler's own experience.

Table 6.1: IDM Model Parameters and Domains

Definition	Symbol	Domains	Units
Max Acceleration	a	$[0.5, 10.0]$	m/sec^2
Max Deceleration	b	$[-10.0, -0.5]$	m/sec^2
Reaction Time	τ	$[1.0, 3.0]$	sec
Time Headway	T	$[1.0, 5.0]$	sec
Distance Headway	s	$[3.0, 12.0]$	m
IDM Parameter	δ	$[3.0, 8.0]$	$-$

It should be noted that parameter δ is a tuning parameter that, according to the authors, is used to control how the acceleration decreases as the vehicle approaches its desired velocity v_0. Therefore it has no units on its own and is used simply to aid in calibrating the model to a specific data set.

The domains are important to the process of calibration because the optimization algorithms must stay within them when searching for the optimum. To create random values for the initial solution pool, the GA is given a uniform random variable based on the feasible domain for each parameter. This ensures that every solution makes sense in the real world. NM and SPSA must also stay within the feasible domains of the parameters, and this is achieved using penalties in the objective function if the algorithms veer outside the acceptable region.

The model parameters chosen for the calibration of Gipps' Model are shown in Table 6.2, along with their domains.

Table 6.2: Gipps' Model Parameters and Domains

Definition	Symbol	Domains	Units
Acceleration	a	$[0.5, 10.0]$	m/sec^2
Deceleration	b	$[-10.0, -0.5]$	m/sec^2
Reaction Time	τ	$[1.0, 3.0]$	sec

Gipps' model has fewer parameters than the IDM. The parameters chosen in here match the first three parameters chosen for the IDM, but Gipps' model does not explicitly use the time and distance headways in the calculations.

For both car-following models, vehicle velocity was not chosen as a model parameter. The reasoning is that the road network under consideration is a section of a limited-access U.S. highway, with a high-speed limit, and it is felt that the majority of cars will try to maximize their speed. The speed limit on this highway is 70 miles per hour, however, according to Mannering (2007), the variance in drivers' chosen speed for a 70 mph speed limit is around 5.24. Therefore a constant maximum velocity of $v = 34.0$ meters per second (around 76 miles per hour) was chosen for all vehicles. Keep in mind that this does not mean all of the vehicles will only ever be traveling 34 meters per second. Rather, it means that 34 meters per second is their *desired* velocity. It is felt that this is both a reasonable choice to make, and will also reduce the complexity of the calibration by reducing the dimensionality of the parameter space.

6.3 Optimization Algorithms

The optimization algorithms that have been chosen for the calibration procedure were Genetic Algorithms (GA), the Nelder-Mead Simplex algorithm (NM), and the Simultaneous Perturbation Stochastic Approximation algorithm (SPSA). These three methods were included in a comparison of opti-

mization techniques for calibrating a car-following model in (L. Li et al., 2016). These authors found similar results to the work done here. Namely that the Nelder-Mead algorithm and Genetic Algorithm perform comparably and perform well in the context of parameter accuracy, but SPSA does not work as well in this context. Details of the three algorithms are presented below in Sections 6.3.1, 6.3.2, and 6.3.3.

6.3.1 Genetic Algorithm

Genetic Algorithms are a class of optimization procedures based on the concepts of evolution and natural selection. They use biologically inspired operators such as mutation, crossover, and selection. They maintain a pool of candidate solutions and then proceed to create new generations of the pool using these operators. Details of the genetic algorithm used in this project are described in Section A.3. An overview of the process is shown below.

Step 1 Create an initial pool of candidate solutions and order them according to function value.

Step 2 Select the top n candidates and use them to create the next generation of the pool by cross-breeding their values.

Step 3 Mutate the new candidate solutions.

Step 4 Reorder the pool and if a maximum number of generations have occurred, or if some stopping criterion has been satisfied, end the algorithm and output the top candidate as the solution. Otherwise, return to Step 2.

Genetic Algorithms have both advantages and disadvantages when compared to other optimization algorithms. The largest advantage they have is the ability to easily overcome getting stuck in local optimum zones. The randomization of the selection of points from the domains of each dimension means that the algorithm always has a direct method to cover new ground in the overall input space of the problem. A second advantage that GAs have over many other algorithms is the lack of reliance on gradients to move to new points.

The genetic algorithm used in this project was designed and implemented by the author using the basic principles of GAs. Various elements of the implementation are found in the Appendix.

Since these simulations take a long time to execute, the GA caps the number of generations at 40. The initial generation is created using Uniform random

variables for each parameter, where each random variable is created using the domain for the specific parameter. The solution pool has a constant size of $n = 15$ so the initialization of the pool creates all 15 candidate solutions as a collection of parameter vectors. The method then executes the simulation with each vector and sorts them based on the value of the loss function.

Subsequent generations are produced using the following process:

1. Take all 6 combinations of the top 4 candidates and produce 6 new candidates through crossover.
2. Execute a mutation on each of the 6 new candidates.
3. Create 5 completely new candidates by using the random variables to round out the 15 candidates in the pool.
4. Run the simulation for each candidate, sort the pool based on the results, and repeat the process.

The cross-breeding phase generates a random index k in the range of parameter vector indices and another random integer j. The value of j determines which of the two parameter vectors will be considered x_1 and which will be considered x_2. The value of k determines where the split in the vector occurs. Assuming there are m indices in the parameter space, a new vector is produced by taking the first k values from x_1 and the last $m - k$ values from x_2.

The mutation phase iterates through the vector, and for each value generates a random value in the range $[-0.2, 0.2]$ and adds this value to 1.0. This creates a multiplicative factor that at most will apply a 20% change to the value in either direction. This strategy was used to accommodate the different relative sizes and domains of the parameters, and to prevent the mutations from being too large. A percentage-based mutation will be applied to all parameters in a more equal way than an additive or simple multiplicative one.

6.3.2 Nelder-Mead Simplex

The Nelder-Mead Simplex algorithm (Nelder & Mead, 1965) is a direct-search algorithm and requires no derivatives or gradients to determine its next search direction. A *simplex* is a convex hull of $n + 1$ vertices (where n is the dimensionality of the parameter space) where the vertices cannot lie on the same hyperplane. A simplex is essentially a generalized tetrahedron in $\mathbb{R}^n$, which would be a triangle in $\mathbb{R}^2$.

Table 6.3: Chosen Parameter Values for the Nelder-Mead Simplex Algorithm

Parameter	Transformation	Value
α	Reflection	1.0
β	Contraction	0.5
γ	Expansion	2.0
δ	Shrinkage	0.5

The algorithm has several defined parameters, one each for the simplex transformations defined below. The parameters and their chosen values are given in Table 6.3

The parameter values essentially define the lengths used in computing new points in each of the transformations defined below.

The optimization algorithm begins by constructing an initial simplex from an initial guess $\mathbf{x}_0$ and the objective function $f(\mathbf{x})$, and then uses simplex transformations to gradually augment and reduce the size of the simplex. Once the simplex has shrunk to a sufficiently small size the vertex with the best function value is returned as the optimal $\mathbf{x}$.

The algorithm maintains a list of the best vertex $\mathbf{x}_b$, the worst vertex $\mathbf{x}_w$, and the second-worst vertex $\mathbf{x}_s$. In each iteration of the algorithm, the *centroid* c of the best "side" is computed, which is the arithmetic mean of the vertices, not including the worst vertex.

The transformations that are used to reduce the simplex are:

1. Reflection - Reflect $\mathbf{x}_w$ around the centroid to generate the point $\mathbf{x}_r$ using Equation 6.3

$$\mathbf{x}_r = c + \alpha(c - \mathbf{x}_w) \tag{6.3}$$

If $f(\mathbf{x}_b) \leq f(\mathbf{x}_r) < f(\mathbf{x}_s)$ then $\mathbf{x}_r$ replaces $\mathbf{x}_w$ in the simplex. Otherwise, the algorithm moves on to the next step.

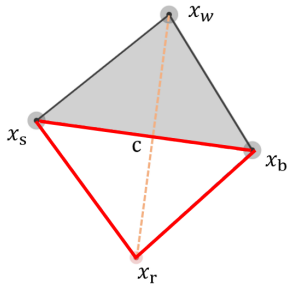


Figure 6.1: Reflection transformation. The new simplex is shown in red.

2. Expansion - Expansion proceeds if $f(\mathbf{x}_r) < f(\mathbf{x}_b)$. If this is the case then expansion point $\mathbf{x}_e$ is produced using Equation 6.4

$$\mathbf{x}_e = c + \gamma(\mathbf{x}_r - c) \tag{6.4}$$

Then there are two cases:

1. If $f(\mathbf{x}_e) < f(\mathbf{x}_r)$ then $\mathbf{x}_e$ is accepted as the new vertex of the simplex, replacing $\mathbf{x}_w$.

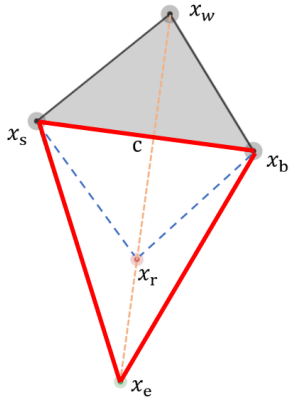


Figure 6.2: Expansion transformation.

2. If $f(\mathbf{x}_e) \geq f(\mathbf{x}_r)$ then $\mathbf{x}_r$ is accepted as the new vertex of the simplex, replacing $\mathbf{x}_w$.

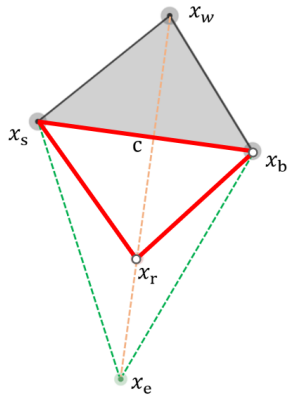


Figure 6.3: Reverting to reflection transformation.

If expansion did not proceed then the algorithm moves on to step 3.

3. Contraction - Contraction occurs if $f(\mathbf{x}_r) \geq f(\mathbf{x}_s)$. There are two cases:

1. If $f(\mathbf{x}_s) \leq f(\mathbf{x}_r) < f(\mathbf{x}_w)$ then compute contraction point $\mathbf{x}_c$ *outside* the centroid using Equation 6.5.

$$\mathbf{x}_c = c + \beta(\mathbf{x}_r - c) \tag{6.5}$$

If $f(\mathbf{x}_c) \leq f(\mathbf{x}_r)$, accept $\mathbf{x}_c$ as the vertex to replace $\mathbf{x}_w$. Otherwise, go on to the shrink operation.

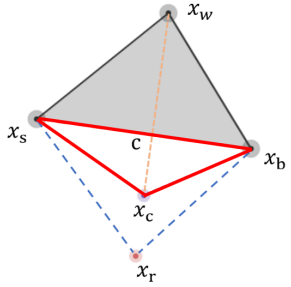


Figure 6.4: Outer contraction transformation.

2. If $f(\mathbf{x}_r) \geq f(\mathbf{x}_w)$ then compute contraction point $\mathbf{x}_c$ *inside* the centroid using Equation 6.6.

$$\mathbf{x}_c = c + \beta(\mathbf{x}_w - c) \tag{6.6}$$

If $f(\mathbf{x}_c) < f(\mathbf{x}_w)$, accept $\mathbf{x}_c$ as the vertex to replace $\mathbf{x}_w$. Otherwise, go on to the shrink operation.

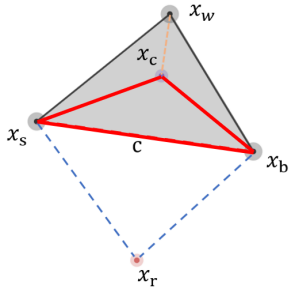


Figure 6.5: Inner contraction transformation.

4. Shrinkage - Compute new vertices for the entire simplex except for the best vertex $\mathbf{x}_b$ using the defined scaling factor δ. In two dimensions this would be done with Equations 6.7 and 6.8.

$$\mathbf{x}_{n1} = \mathbf{x}_b + \delta(\mathbf{x}_w - \mathbf{x}_b) \tag{6.7}$$

$$\mathbf{x}_{n2} = \mathbf{x}_b + \delta(\mathbf{x}_s - \mathbf{x}_b) \tag{6.8}$$

This will contract the simplex down closer to the best point.

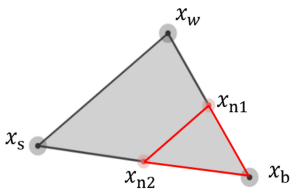


Figure 6.6: Shrinkage transformation.

Each iteration can end after any one of these steps depending on the outcome of the transformation involved.

6.3.3 Simultaneous Perturbation Stochastic Approximation

Gradient-based optimization algorithms work extremely well under certain assumptions, which are unfortunately not present in this context. Since the function being minimized relies on executing a microscopic discrete-event traffic simulation every time it is evaluated, there is no expectation of requirements for gradient techniques like continuity. However, the main reason gradient-based optimization techniques work poorly for this context is that they rely on executing the function twice for each dimension of the parameter space for each iteration of the algorithm. In many instances, this would lead to far too many executions of the simulation and would result in computation times that are far too long. SPSA (Spall et al., 1992) solves much of the second problem by stochastically generating a perturbation vector (typically from a Bernoulli random variable) and calculating a gradient approximation using only two function evaluations, regardless of the size of the parameter space.

The formulas to compute the gradient approximation in each iteration are shown below in equations 6.9 and 6.10. Equation 6.9 requires the calculation of vector Δ_k which is here determined by a Bernoulli random variable set to output either -1 or 1, making each value Δ_{kn} is ± 1. However, in the general case, the values of vector Δ_k can be determined by any zero-mean probability distribution that satisfies the conditions found in Spall (1992), which could yield values for Δ_{kn} which are not ± 1. The exponent "-1" in Equation 6.9 for each value Δ_{kn} is a convenient notation for representing that these values have been reciprocated.

$$\hat{g}(\hat{\theta}_k) = \frac{f(\hat{\theta}_k + c_k\Delta_k) - f(\hat{\theta}_k - c_k\Delta_k)}{2c_k} \begin{bmatrix} \Delta_{k1}^{-1} \\ \Delta_{k2}^{-1} \\ \vdots \\ \Delta_{kp}^{-1} \end{bmatrix} \tag{6.9}$$

$$\hat{\theta}_{k+1} = \hat{\theta}_k - a_k \hat{g}_k(\hat{\theta}_k) \tag{6.10}$$

The algorithm parameters define *gain sequences* a_k and c_k, which guide how the step size shrinks as the process progresses. These parameters are a, c, A, α, and γ, which are used in the gain sequences shown in equations 6.11 and 6.12. The value k is the iteration value.

$$a_k = \frac{a}{(A + k + 1)^\alpha} \tag{6.11}$$

$$c_k = \frac{c}{(k + 1)^\gamma} \tag{6.12}$$

The original authors (Spall, 1998a) gives the following recommendations for the parameter values:

1. $\alpha = 0.602$, though they say the value $\alpha = 1.0$ may also be used.
2. $\gamma = 0.101$, though they say the value $\gamma = 1/6$ may also be used.
3. c should be approximately equal to the standard deviation in the measurement noise of the function and should be chosen so that the perturbations do not get excessively large.
4. A should be much less than the maximum number of iterations expected.
5. a should be a value that makes a_k keep the changes in θ_k small in the early iterations.

Table 6.4 shows the parameters that were chosen for SPSA in this context.

Table 6.4: Chosen SPSA Parameter Values

Parameter	Value
α	1.0
γ	0.167
c	0.5
A	2.0
a	10.0

The gain sequences should decrease as the value of k increases. This results in gradually smaller perturbations and smaller movement in the direction of the approximate gradient $\hat{g}(\hat{\theta}_k)$ defined below in Equation 6.9. In the context of the simulation parameters given in Table 6.1 if a_k and c_k are too large, then

the algorithm could send a parameter value outside of its domain. However, if the values of a_k and c_k are too small then there is likely to be no appreciable difference in the simulation outputs. Below is the reasoning for choosing the values of each parameter.

- The values of α and γ were chosen directly from the advice of the authors (Spall, 1998a).
- The perturbation in each iteration is given by $c_k\Delta_k$. With c set to 0.5, the perturbations will never be too large in the context of the traffic parameters.
- The number of iterations is desired to be small in this context since the function evaluations are so expensive. The value of $A = 2.0$ should be sufficiently small to satisfy the advice of the authors.
- The distance the algorithm moves in the direction of the approximate gradient is given by $a_k\hat{g}_k(\hat{\theta}_k)$. This distance should be large enough in the early iterations to not just search in a small space, but not so large that it is moving outside of the parameter domains. The value $a = 10.0$ gives values for a_k that put the movement distance in a desirable zone throughout the process.

The algorithm begins with an initial guess $\mathbf{x}_0$ and then proceeds into the gradient approximation process, which, just as a normal gradient-based procedure would do, yields the direction of movement for the next guess $\mathbf{x}_k$.

The algorithm ends either once a maximum number of iterations has occurred or based on some stopping criteria.

6.4 Calibration Methodology

GAs have one advantage over the other two types of search methods which is that the creation of random solutions and random mutations will more robustly cover the surface. The NM and SPSA approaches have a higher chance of finding local optima rather than global optima on a very bumpy surface. Both NM and SPSA require a starting point $\mathbf{x}_0$, which on a very noisy surface can have a great effect on the optimization results. The two algorithms were tested on various starting points, which are shown in Table 6.5. For both algorithms, the first four starting points differ only in the acceleration and deceleration parameters, where these values are chosen to divide their respective domains into thirds.

The choice of values in x_1 through x_4 is summed up here:

- Parameter δ is usually set to 4.0 according to the original authors.
- Reaction time was chosen as a small value just inside the bottom boundary of its domain.
- Time headway and distance headway were both set at values halfway through their respective domains.

Table 6.5: Initial Starting Points for IDM Calibration

Point	a	b	τ	T	s	δ
$\mathbf{x}_1$	7.0	-3.5	1.5	3.0	7.5	4.0
$\mathbf{x}_2$	3.5	-7.0	1.5	3.0	7.5	4.0
$\mathbf{x}_3$	7.0	-7.0	1.5	3.0	7.5	4.0
$\mathbf{x}_4$	3.5	-3.5	1.5	3.0	7.5	4.0

Table 6.6: Initial Starting Points for Gipps Calibration

Point	a	b	τ
$\mathbf{x}_1$	7.0	-3.5	1.5
$\mathbf{x}_2$	3.5	-7.0	1.5
$\mathbf{x}_3$	7.0	-7.0	1.5
$\mathbf{x}_4$	3.5	-3.5	1.5

To test the three different calibration approaches data from four Tuesdays in 2018 was used to generate arrivals for the simulation using the HAM, Ratio, Difference, and Online SARIMA arrival models defined in Section 4.3. The Offline SARIMA model was left out due to its having the worst performance for modeling simulation arrivals. The simulations were then calibrated over each fifteen-minute period from 6:00 am until 6:00 pm on the four chosen Tuesdays with each of the chosen algorithms, with NM and SPSA also being calibrated across the four given starting points. Only data from *previous* time periods are used to create the simulation, and the results are compared to the data from the current time period, which is an out-of-sample calibration procedure.

For each time period under consideration, a cold start using the previous 15-minute interval is used to get the proper number of simulation entities into the model. Otherwise, the simulation would begin without enough vehicles already on the road for the time period of focus. On this fairly short length of highway, fifteen minutes is enough of a cold start to have the right number of vehicles on the road. Forecasts for arrival counts are generated and the NHPP for the arrival times is constructed as discussed in Chapter 4. The simulation continues to the

end of the fifteen-minute period just after the time period under consideration to allow the vehicles to exit smoothly. The statistical comparisons using the loss function are limited to just the time period of interest.

Fifteen-minute intervals were chosen both to avoid long simulation execution times and because intervals that are too long will not have the flexibility to adapt to changes in traffic patterns. Each simulation execution counts the number of vehicles that pass through the sensor locations and then compares those counts to the actual data for that day using the sMAPE calculation in Equation 4.10. The calibrated parameters, full list of optimal values over epochs, and execution times were saved for each calibration run.

6.5 Calibration Results

The calibrations are analyzed below in a few different ways, which are shown below. Within each paradigm of analysis, the calibrations were compared according to several different characteristics. Each calibration can be analyzed according to the car-following model, optimization algorithm, initial points/streams, and arrival model.

- Calibration Accuracy - How accurate, in terms of sMAPE, were the calibrations?
- Calibration Efficiency - How efficient, in terms of total execution time, were they?
- Calibration Improvement - How did each optimization algorithm improve as they operated?

After these comparisons of the calibration process are done, an analysis of the calibrated parameters is presented, which is then followed by a review of other researchers' calibration efforts in the realm of traffic model calibration.

6.5.1 Calibration Accuracy

It turns out that these four starting points yield the same results in the uncalibrated model, which suggests that acceleration and deceleration are not the most consequential parameters for this case study. The average results of calibrating the model with each initial point can be seen in Table 6.7 and show that in general the best results came from the GA. Note that the GA does not

Table 6.7: Average Calibration Results Across Initial Points.

Point	NM	GA	SPSA	Uncalibrated
$\mathbf{x}_1$	6.93887	6.30663	7.11991	9.61395
$\mathbf{x}_2$	6.64965	6.25586	7.27894	9.61395
$\mathbf{x}_3$	6.65474	6.31105	7.02135	9.61395
$\mathbf{x}_4$	6.52576	6.2821	7.10472	9.61395

depend on an initial starting point so the differences in this table are the random number streams used in the evolutionary process to create new candidate solutions.

The model was also compared across the different types of arrival models. These results are presented in Table 6.8. It is again clear that the GA produced better calibration results than the Nelder-Mead Simplex and SPSA. It is also clear that the online arrival modeling techniques are an improvement over the HAM arrival model, though it that is not as clear for the SARIMA process. It is unclear why the online SARIMA arrival method has performed more poorly under calibration than the other online methods, and more investigation is warranted.

Table 6.8: Average Calibration Results Across Arrival Models.

Method	HAM	Ratio	Difference	SARIMA
NM	6.94606	6.42806	6.37851	7.01639
GA	6.58620	5.95635	6.04709	6.56600
SPSA	7.38289	6.79691	6.80331	7.54180

Figure 6.7 shows the average calibration results in terms of sMAPE values for the three methods throughout the 12-hour period across all other characteristics.

Figure 6.8 shows the average calibration results across the initial starting points for the Nelder-Mead Simplex and SPSA, while Figure 6.9 shows the results across the random number streams for the GA.

It is fairly clear from the figure that there is little difference in terms of starting points in the accuracy of the calibrations. The figure includes both the Nelder-Mead Simplex and SPSA in the average, and the overall averages for each point are shown in Table 6.9.

From both the table and the figure it appears that point x_4 is *slightly* better in terms of accuracy than the other points.

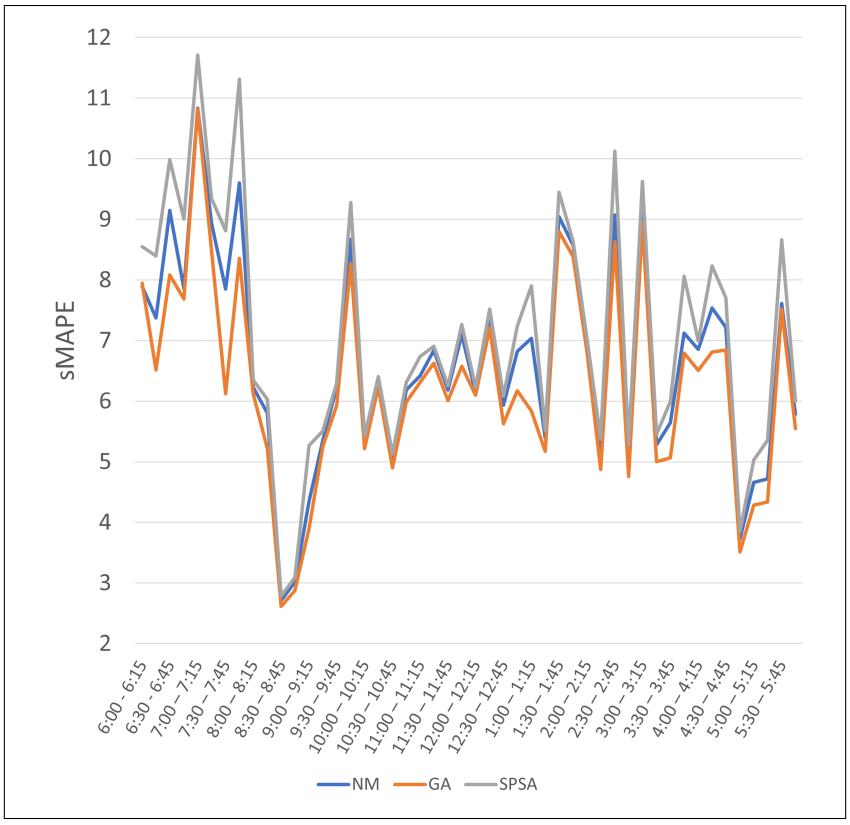


Figure 6.7: Results of calibration in terms of algorithm

Table 6.9: Average calibration results across initial points for NM and SPSA.

x_1	x_2	x_3	x_4
7.0294	6.9643	6.838	6.8152

The difference between random number streams for the genetic algorithm is unlikely to have a large effect on the calibration results, and Figure 6.9 and Table 6.10 show that this is indeed the case.

The calibrations were also compared across the four arrival methods. Figure 6.10 shows the average results of these calibrations.

It appears that the SARIMA model performed the worst in terms of calibration accuracy, which is surprising. Table 6.11 shows the comparison of the calibrations' overall average sMAPE values. Within this analysis, the ratio-based arrival model performed slightly better than the difference-based arrival model, and the HAM arrival model came in third, with the SARIMA model finishing

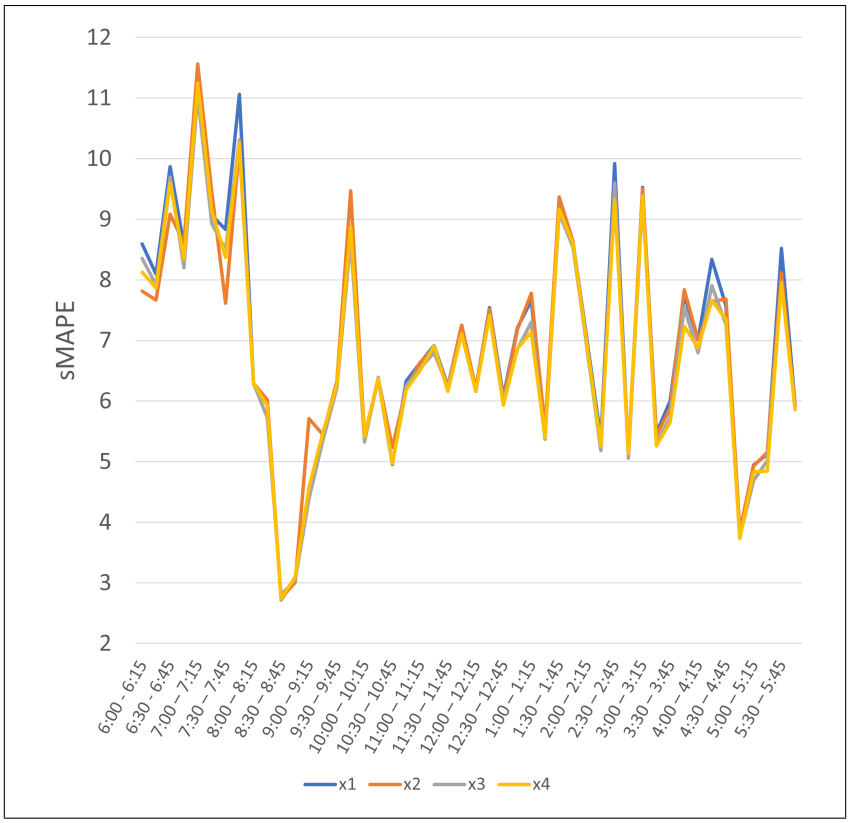


Figure 6.8: Results of calibration in terms of initial starting points

last. This relatively poor performance of the SARIMA model requires further study.

Figure 6.11 shows the calibration results across the two car-following models, and once again there doesn't appear to be a huge difference in the calibration results in this comparison. It can be seen in the figure that Gipps' Model appears to outperform the IDM somewhat significantly in the very earliest part of the day, and then only slightly during most of the rest of the 12-hour period. Table 6.12 shows the overall average for both car-following models, and Gipps' Model has a slightly better result.

Looking at each of Figures 6.7, 6.8, 6.9, 6.10, and 6.11 it is evident that the same overall shape to the calibrations comes out in each one, and the largest difference amongst all of the comparisons occurs between the optimization algorithms, where the GA pretty clearly outperforms the other two methods, with the Nelder-Mead Simplex coming in second place. Another comparison that shows quite a bit of difference is the comparison across arrival models. This

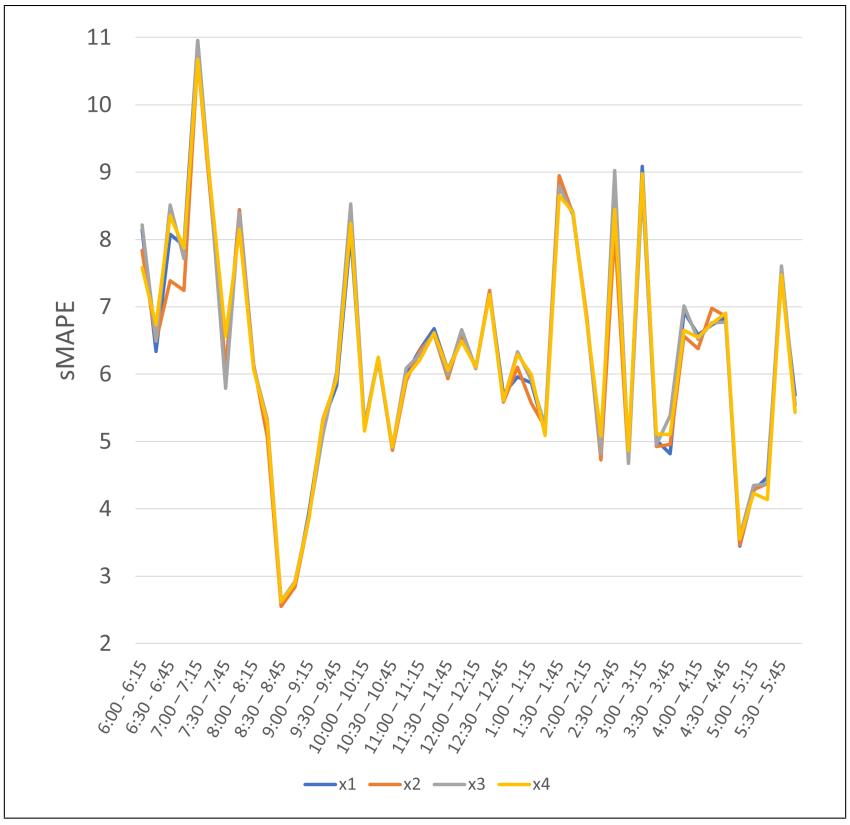


Figure 6.9: Results of calibration in terms of random number streams

is not very surprising since the choice of arrival model determines the initial space headways of the vehicles in the system, and the more accurate this process is, it stands to reason that the rest of the simulation stands a greater chance of being more accurate as well. What was very surprising was the performance of the SARIMA arrival model in this regard. It is difficult to explain and warrants further research.

Table 6.10: Average calibration results across random number streams for the GA.

x_1	x_2	x_3	x_4
6.3066	6.2559	6.311	6.2821

Table 6.11: Average calibration results across arrival models

HAM	Ratio	Difference	SARIMA
6.9717	6.3938	6.4096	7.0414

Table 6.12: Average calibration results across car-following models

IDM	Gipps
6.802	6.6062

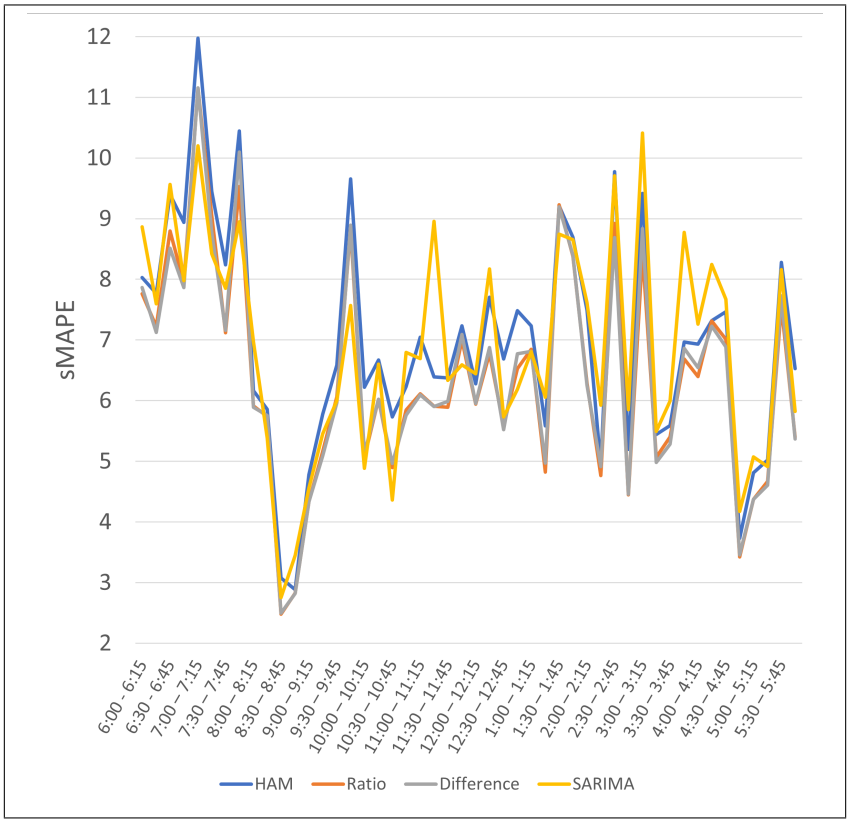


Figure 6.10: Results of calibration in terms of arrival method

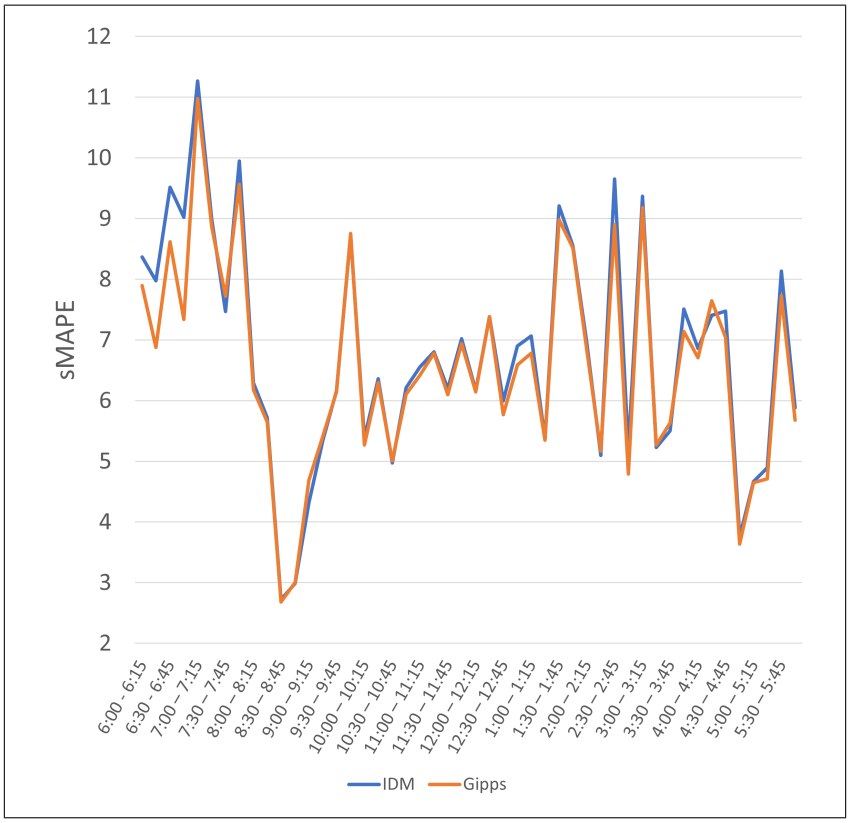


Figure 6.11: Results of calibration in terms of car-following model

6.5.2 Calibration Efficiency

The efficiency of the calibrations is analyzed in two ways. First, the overall execution times of each algorithm are compared and contrasted. Second, the improvement of the calibrations over the epochs of the algorithms is compared. Both of these efficiency paradigms are important in a system which is implicitly an online system. In an online system, it is desired that any calibrations can be done in a reasonable amount of time. Execution times are clearly a reasonable metric for how efficient an optimization algorithm is, but how quickly an algorithm *improves* is also important. If one algorithm improves more quickly than the others, then it might be desirable in certain situations.

Figures 6.12 and 6.13 show the spread of the execution time data. It is evident that there were many outliers in the execution times. These figures were prepared using Microsoft Excel and the exact formula for calculating outliers has not officially been published but after some testing, it appears that they use the common rule that any value outside of 1.5 times the inter-quartile range is an outlier.

Table 6.13: Analysis of outliers in execution time data

Algorithm	Total Calibrations	Outliers	Percent
NM	3072	340	11.07%
GA	3072	178	5.79%
SPSA	3072	212	6.90%

It is evident that in terms of execution times the Nelder-Mead Simplex has a much higher chance of requiring more time to find an optimum point. The number of iterations of each of the algorithms was controlled so as not to result in extremely long calibration times, and it is likely that the Nelder-Mead Simplex had to run simulations that had some sort of pathological traffic behavior. For example, if accelerations are too low then it can lead to vehicles going very slowly. Another example would be if the desired space between vehicles is too large, then it could result in traffic jams at the simulation source. Still, even though there were many more outlier execution times for the Nelder-Mead Simplex, the Genetic Algorithm took, on average, longer to reach an optimum point.

Figure 6.14 shows the overall execution times of the algorithms. The figure clearly shows that the GA takes much longer to reach optimum values than the other methods. It is not surprising that SPSA performs more efficiently than the other methods as it only requires two function executions in each iteration of the algorithm. Combining the two perspectives reveals that neither the Nelder-Mead nor SPSA should be overlooked for the task of calibrating

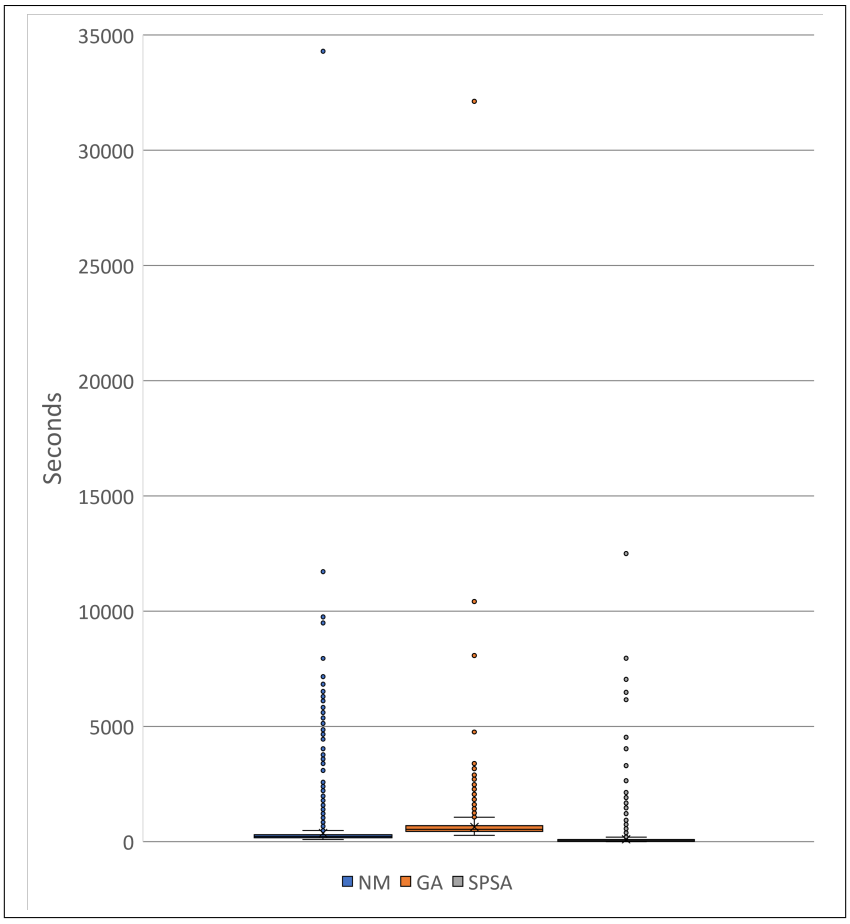


Figure 6.12: All execution times including outliers.

traffic models. Indeed, for an online simulation system, where recalibrations of the model will have to be done from time to time, a time-efficient algorithm that can still produce "pretty good" results might be the method best suited for the purpose.

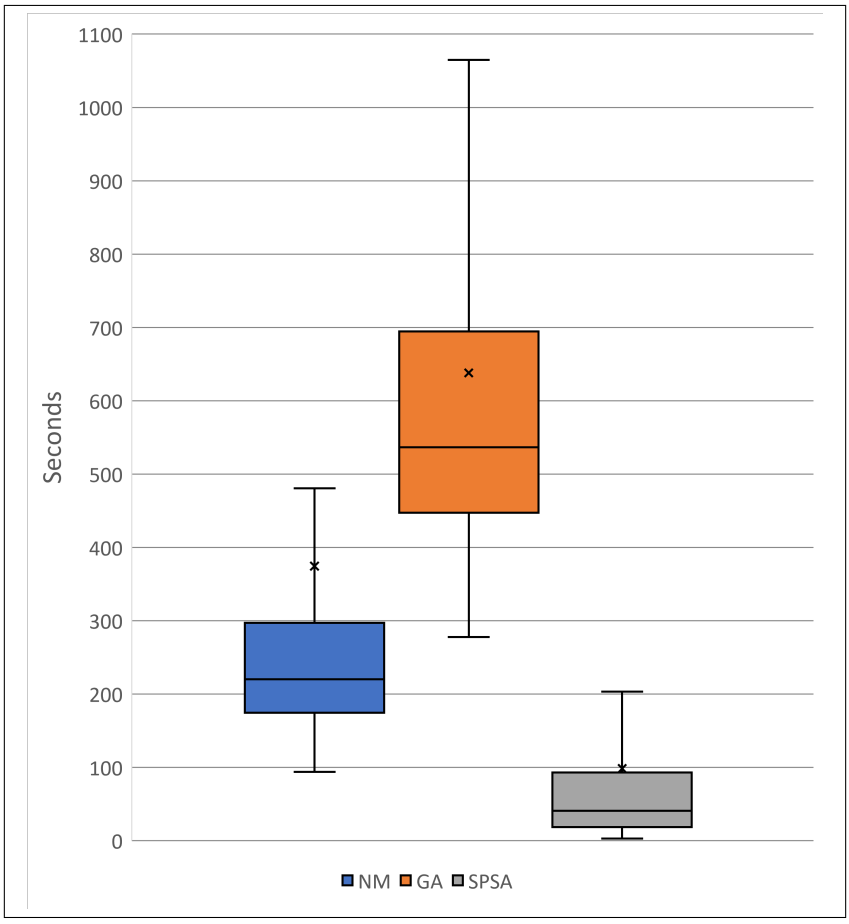


Figure 6.13: All execution times not including outliers.

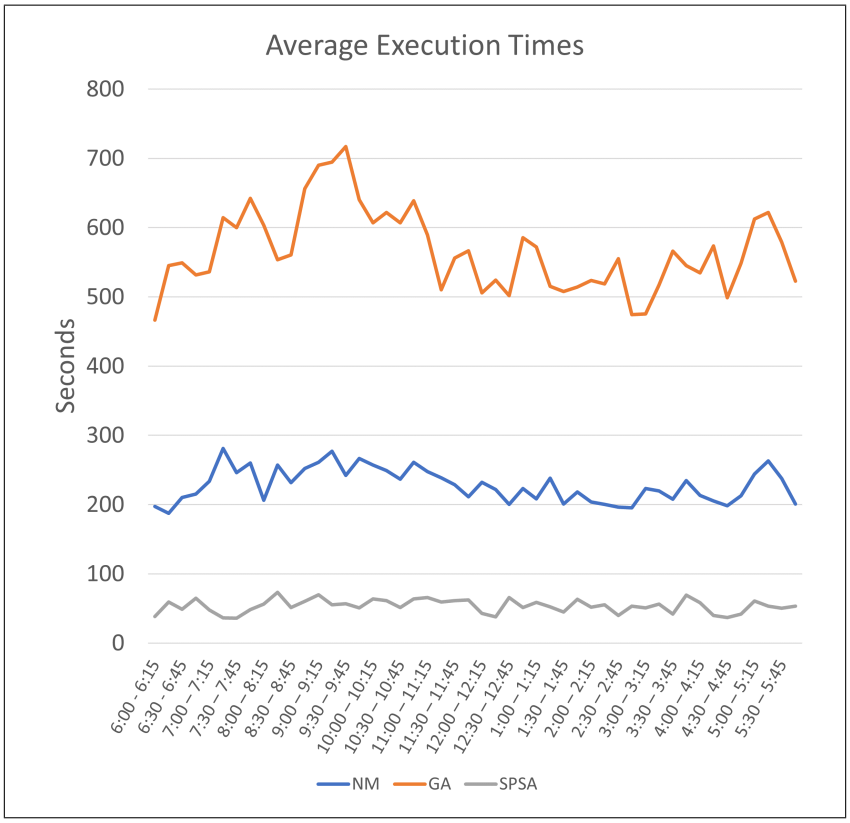


Figure 6.14: Efficiency Comparison Using Means With Outliers Removed

The average results when comparing the improvement over epochs of the algorithms can be seen in Figure 6.15. This figure shows that the GA overall had the most dramatic improvement of the three methods throughout the entirety of the optimization procedure. SPSA does not improve much throughout the optimization procedures. The Nelder-Mead algorithm shows faster improvement in the early stages than the other two methods, which could suggest it is a good technique when the number of iterations of the calibration needs to be kept very low.

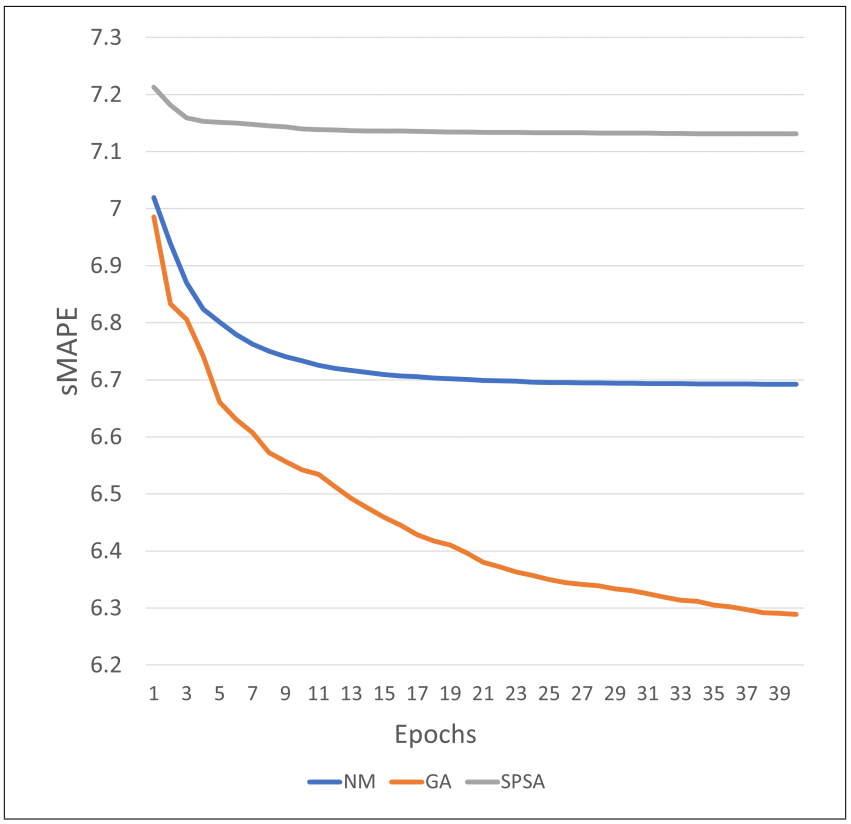


Figure 6.15: Change in Objective Function Value by Epoch

Figure 6.16 shows the calibration improvement by epochs across the four initial points. Note that since the GA doesn't depend on initial points it has been left out of this figure. It is clear from the graph that points x_3 and x_4 show the best calibration and even show a slight bit more improvement in the early iterations of the algorithms compared to the other two initial points. One common trait between points x_3 and x_4 is that their acceleration and deceleration values are either both aggressive (as in x_3) or more cautious (as in x_4). Points x_1 and x_2 have a mixture of either aggressive acceleration and cautious deceler-

ation or vice versa. This picture would seem to indicate that consistency in the severity of accelerating and braking is more important.

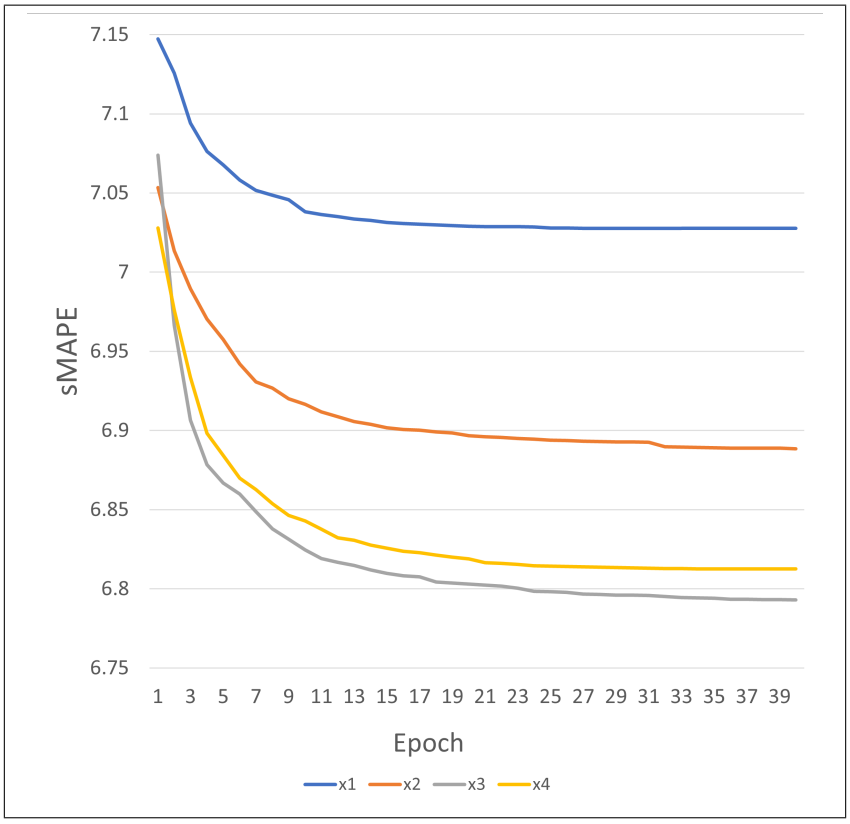


Figure 6.16: Change in Objective Function Value by Epoch Across Initial Points

6.5.3 Calibrated Parameters

The car-following models being calibrated are the Intelligent Driver Model and Gipps' Model, which are repeated below in Equations 6.13, 6.14, and 6.15 for the IDM, and Equations 6.16, 6.17, and 6.18 for Gipps' Model.

$$\dot{v}_n(t+\tau) = a\left(1 - \left(\frac{v_n(t)}{V_0}\right)^{\delta} - \left(\frac{s^*(v_n(t), \Delta v_n(t))}{s_n(t)}\right)^2\right) \tag{6.13}$$

where

$$s^*(v_n(t), \Delta v_n(t)) = s_0 + v_n(t)T + \frac{v_n(t)\Delta v_n(t)}{2\sqrt{ab}} \tag{6.14}$$

and

$$s_n(t) = x_{n-1}(t) - x_n(t) - l_{n-1} \tag{6.15}$$

$$v_n(t+\tau) = \min\left[v_F(t+\tau), v_C(t+\tau)\right] \tag{6.16}$$

where

$$v_F(t+\tau) = v_n + 2.5a_n\tau(1 - v_n(t)/V_0)(0.025 + v_n(t)/V_0)^{1/2} \tag{6.17}$$

is the *Free driving* velocity and the *Car-following* velocity is given by

$$v_C(t+\tau) = b_n\tau + \sqrt{b_n^2\tau^2 - b_n[2\Delta x_n - v_n(t)\tau - v_{n-1}(t)^2/\hat{b}]} \tag{6.18}$$

The parameters calibrated for the Intelligent Driver Model are

- Acceleration a
- Deceleration b
- Reaction Time τ
- Desired Time Headway T
- Desired Space Headway s
- Model Parameter δ

while the parameters calibrated for Gipps' Model are

- Acceleration a
- Deceleration b
- Reaction Time τ

Gipps' Model does not incorporate the driver's desired time or space headway into the formulas, nor does it have a model-fitting parameter such as δ.

Average values for each parameter of the IDM are shown in Table 6.14 and are given for each of the optimization methods. Table 6.15 shows the values for Gipps' Model. Tables 6.14 and 6.15 also break down the calibrations by initial point/random stream. For Nelder-Mead and SPSA, it can be seen in the tables in the acceleration and deceleration that the choice of initial point does have a dramatic effect on the final parameter value. This is likely a result of a very noisy and bumpy response surface which the two algorithms found difficult to maneuver within. The initial points for the Nelder-Mead Simplex and SPSA, given in Tables 6.5 and 6.6, differ only in the acceleration and deceleration parameters. This resulted, for those two optimization algorithms, in essentially a bimodal distribution of parameter values for acceleration and deceleration. The genetic algorithm was run with four different random streams, but in each case ended up in the same general region of values for parameters. Of course, GAs are more resilient to bumpy surfaces and these results certainly support that idea. This then is another good argument in favor of GAs if calibration accuracy is the primary desire.

It is important to recall here that the calibrations were made for each 15-minute interval from 6:00 am to 6:00 pm, with the reasoning being that the behavior of drivers is likely to change throughout the day depending on the volume of traffic on the highway. Therefore it is important to get an idea of the changing parameter values throughout the day.

It is also important to note that each parameter was not calibrated in isolation. All of the parameters were calibrated simultaneously, and the values of some parameters during the optimization process could certainly have an effect on the ultimate results of the other parameters.

Table 6.14: IDM Calibrated Parameters

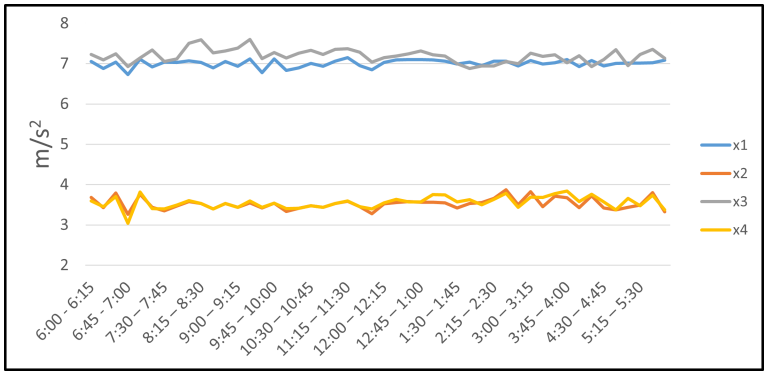
		Parameters					
Initial Point	Method	a	b	τ	s	T	δ
x_1	NM	7.01	-3.45	1.78	7.62	3.10	4.47
	GA	5.60	-5.54	2.01	6.63	2.25	5.62
	SPSA	7.00	-3.49	1.50	7.53	3.02	3.99
x_2	NM	3.52	-6.94	1.76	7.62	3.08	4.51
	GA	5.34	-5.87	1.89	6.27	2.21	5.3
	SPSA	4.82	-5.89	1.83	7.83	3.93	3.66
x_3	NM	7.19	-7.20	1.59	7.56	3.15	4.44
	GA	5.89	-5.32	2.01	6.75	2.28	5.52
	SPSA	6.75	-6.77	1.51	7.73	3.13	4.03
x_4	NM	3.55	-3.41	1.65	7.64	3.11	4.45
	GA	5.10	-6.19	1.84	6.14	2.15	5.19
	SPSA	4.10	-3.65	1.75	7.40	3.07	4.42

Table 6.15: Gipps Calibrated Parameters

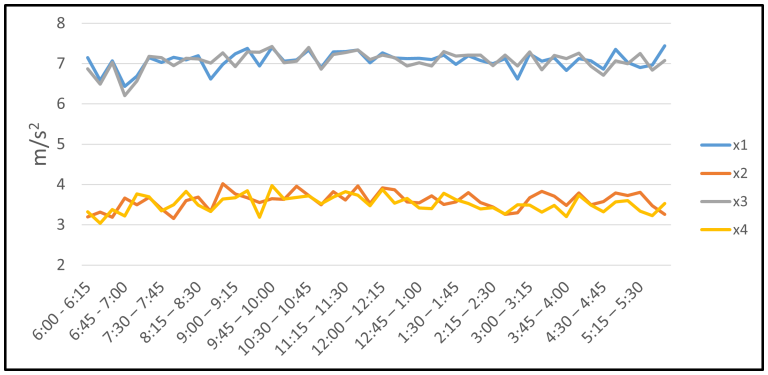
		Parameters		
Initial Point	Method	a	b	τ
x_1	NM	7.07	-3.21	1.77
	GA	6.18	-4.14	2.57
	SPSA	6.98	-3.55	1.49
x_2	NM	3.60	-6.71	1.58
	GA	5.90	-4.80	2.52
	SPSA	3.83	-7.10	1.43
x_3	NM	7.06	-6.67	1.73
	GA	6.30	-4.05	2.58
	SPSA	6.97	-7.05	1.5
x_4	NM	3.53	-3.31	2.12
	GA	5.96	-4.64	2.53
	SPSA	3.50	-3.54	1.46

6.5.4 Acceleration

Consider the parameter values shown in Figure 6.17 which shows, via line graphs, the change in the calibrated acceleration parameter value throughout the 12-hour period for both car-following models, using the Nelder-Mead algorithm. First, the bimodal nature of the calibrated parameters, mentioned above, can be clearly seen in the figures. And though the parameter values are fairly consistent throughout the day, there are some fluctuations that are important to capture.



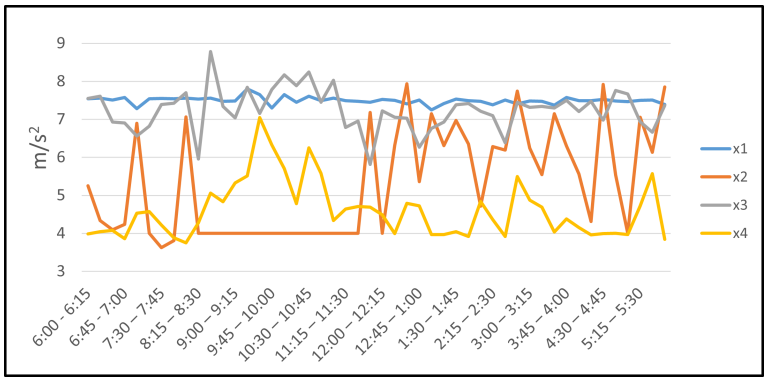
(a)



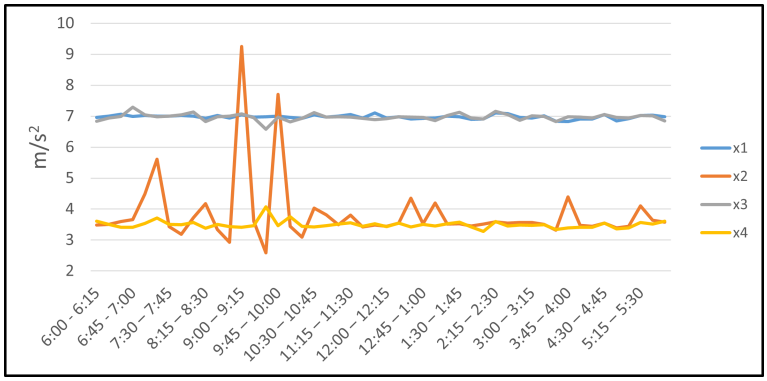
(b)

Figure 6.17: a. Calibrated acceleration value using the IDM and Nelder-Mead. b. Calibrated acceleration value using Gipps' Model and Nelder-Mead.

This bimodal distribution of parameter values is seen as well with SPSA as the optimization algorithm, in Figure 6.18, though SPSA shows somewhat more of an ability to jump out of local optima. Random restarts could improve the ability to escape local optima, but would likely suffer from decreased efficiency performance. The internal parameters of the optimization algorithms could also be massaged for the context, however, this would also come with the price of decreased efficiency considering the time it can take to find the optimal parameters of an optimization algorithm.



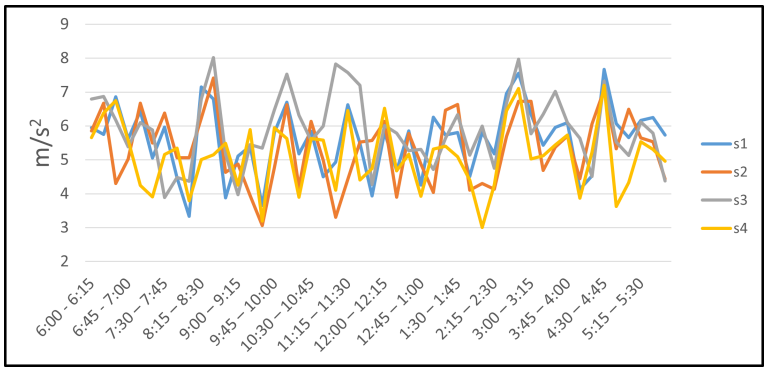
(a)



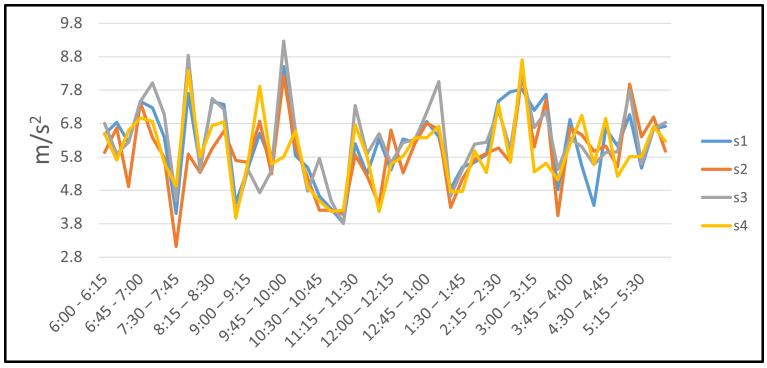
(b)

Figure 6.18: a. Calibrated acceleration value using the IDM and SPSA. b. Calibrated acceleration value using Gipps' Model and SPSA.

The picture is quite different when examining the results for the genetic algorithm, as can be seen in Figure 6.19. The genetic algorithm was calibrated using four different random number streams but does not use initial points in the algorithm. As a result, there is a sense that the optimization process doesn't get stuck in local optima. Still, there appear to be a large number of inconsistencies in the actual parameter values throughout the day.



(a)

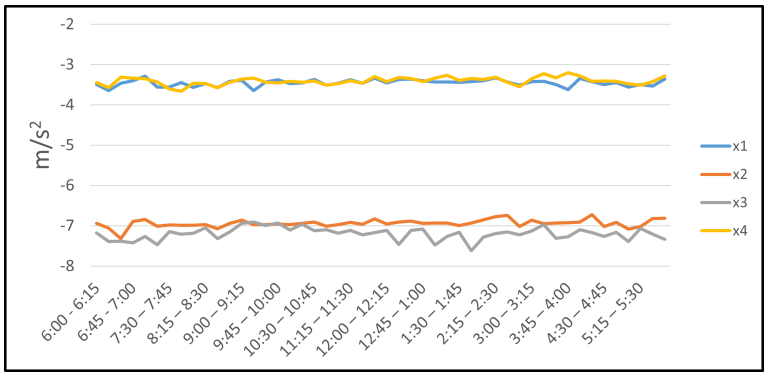


(b)

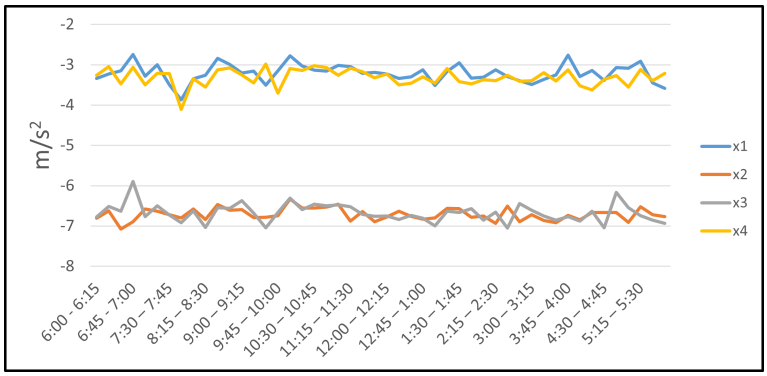
Figure 6.19: a. Calibrated acceleration value using the IDM and the GA. b. Calibrated acceleration value using Gipps' Model and the GA.

6.5.5 Deceleration

The deceleration parameter, like the acceleration parameter, was given two different initial values in the initial points given in Tables 6.5 and 6.6, and in Figures 6.20 and 6.21 below we can see the same type of bimodal distribution of calibrated parameter values that was seen above for the acceleration parameter. This is not unexpected, once again due to the likelihood that the Nelder-Mead Simplex and SPSA algorithms can have a difficult time escaping local minima on noisy surfaces.

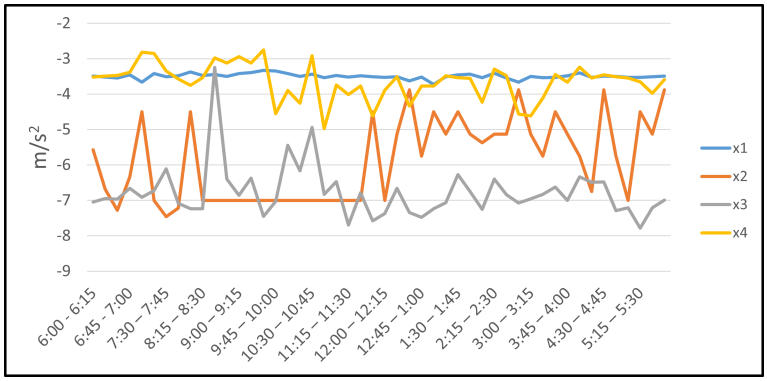


(a)

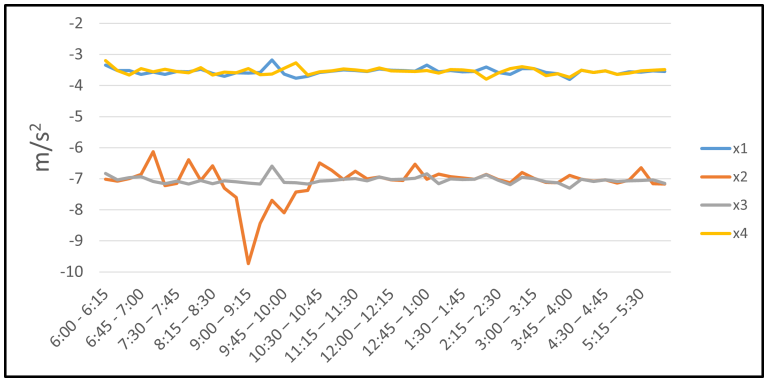


(b)

Figure 6.20: a. Calibrated deceleration value using the IDM and Nelder-Mead. b. Calibrated deceleration value using Gipps' Model and Nelder-Mead.

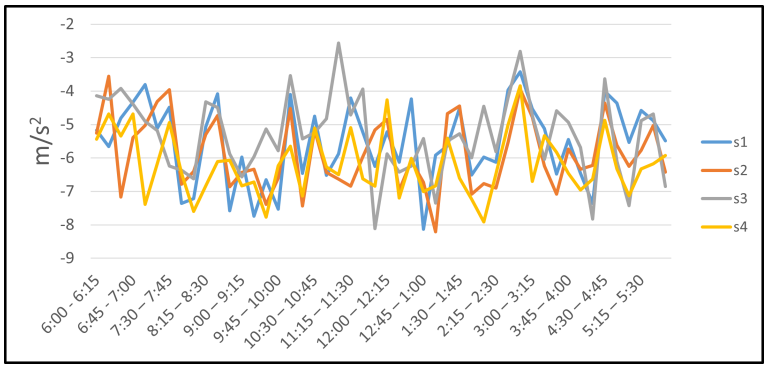


(a)

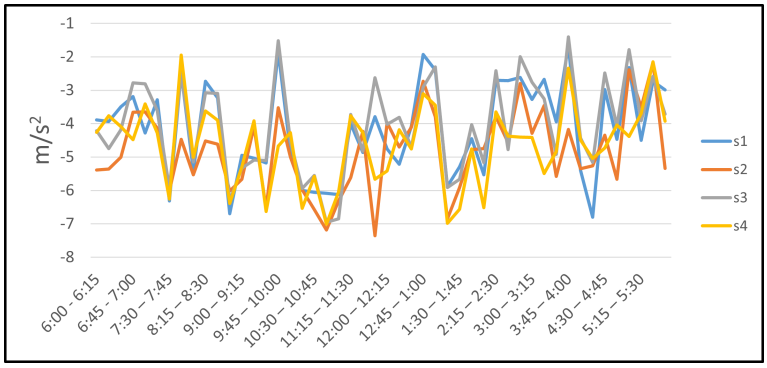


(b)

Figure 6.21: a. Calibrated deceleration value using the IDM and SPSA. b. Calibrated deceleration value using Gipps' Model and SPSA.



(a)

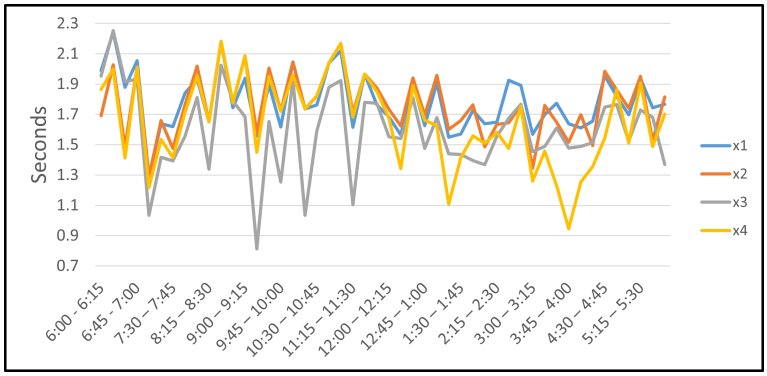


(b)

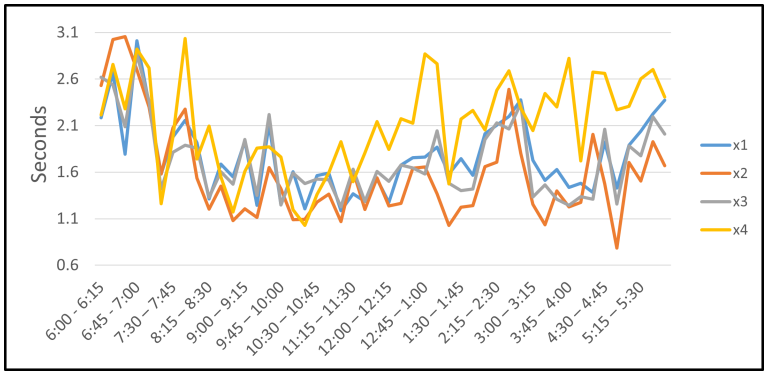
Figure 6.22: a. Calibrated deceleration value using the IDM and the GA. b. Calibrated deceleration value using Gipps' Model and the GA.

6.5.6 Reaction Time

All four initial points are identical in the initial reaction time value. Figures 6.23, 6.24, and 6.25 show the calibrated reaction time values for both car-following models, across all three optimization techniques, and across all four initial points/random number streams. The majority of calibrated values are between 1 and 2 seconds, which seems reasonable. Gipps' Model, calibrated with the genetic algorithm, actually shows calibrated reaction time values mostly between 2 and 3 seconds, which seems a little high, but there was a large amount of consistency in these results across the four random number streams. The calibration of Gipps' Model with the Nelder-Mead Simplex algorithm also shows slightly higher results than is seen in other situations. It appears there was a greater amount of agreement between the three methods, generally speaking, with the IDM.



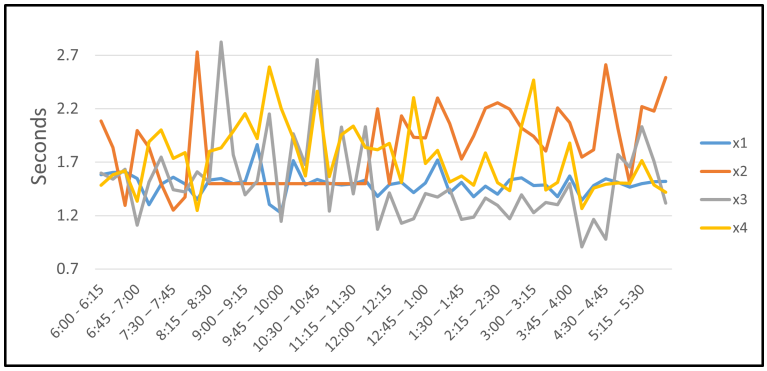
(a)



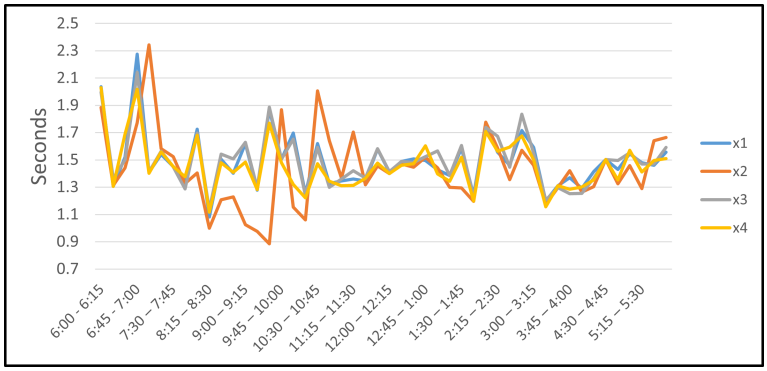
(b)

Figure 6.23: a. Calibrated reaction time value using the IDM and Nelder-Mead. b. Calibrated reaction time value using Gipps' Model and Nelder-Mead.

The final three parameters are only used in the Intelligent Driver Model. Gipps' Model does not incorporate a driver's *desired* time or space headway

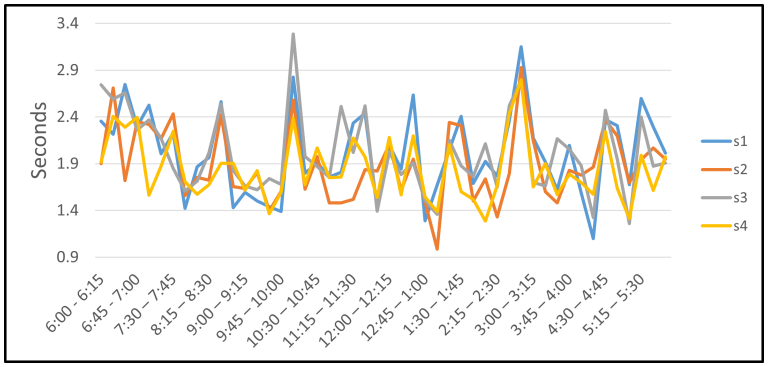


(a)

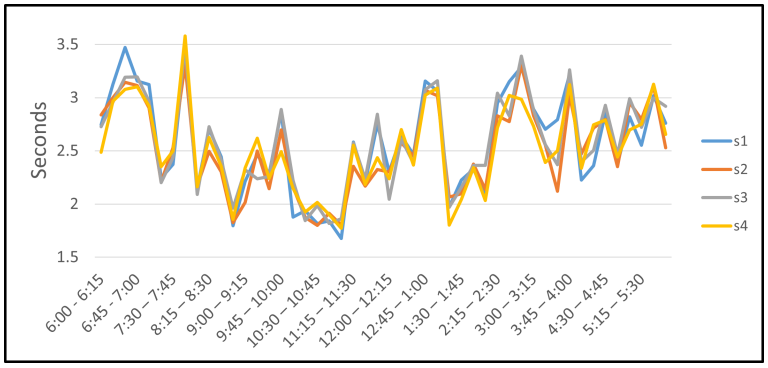


(b)

Figure 6.24: a. Calibrated reaction time value using the IDM and SPSA. b. Calibrated reaction time value using Gipps' Model and SPSA.



(a)



(b)

Figure 6.25: a. Calibrated reaction time value using the IDM and the GA. b. Calibrated reaction time value using Gipps' Model and the GA.

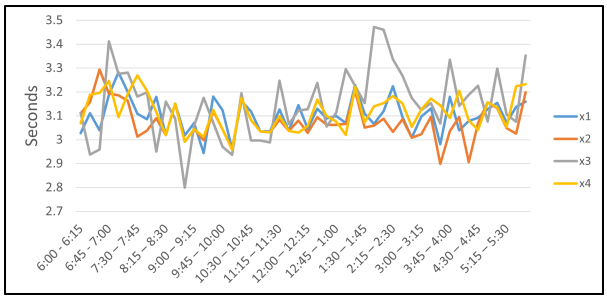
into its formulas, while the IDM does. The IDM also has the parameter δ that acts as an exponential model-fitting parameter. Gipps' Model has no such parameter.

6.5.7 Time Headway

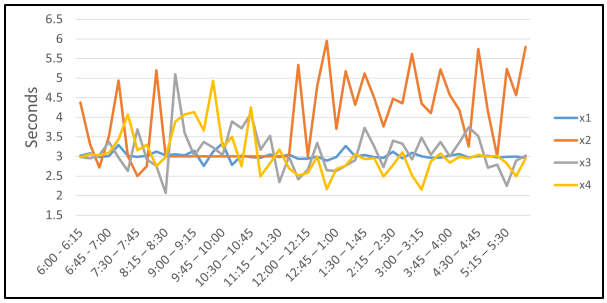
The calibrated values for time headway are shown in Figure 6.26, and show that the genetic algorithm generally settled on lower values than the other two methods. The range of time headway values for Nelder-Mead is mostly between 3 and 3.5 seconds, which given the long tradition of the "three-second rule" seems like a reasonable set of values.

The calibrations of time headway using SPSA show a large amount of inconsistency across both initial points, and throughout the time period. The calibrations are all fairly chaotic in the early part of the day, but smooth out for initial points x_1, x_3, and x_4. The calibration for initial point x_2 continues to be quite chaotic, and fairly far removed from the other three initial points, in the second half of the day. Still, it seems that a majority of the calibrated values fall into the same general range of 3 to 3.5 seconds.

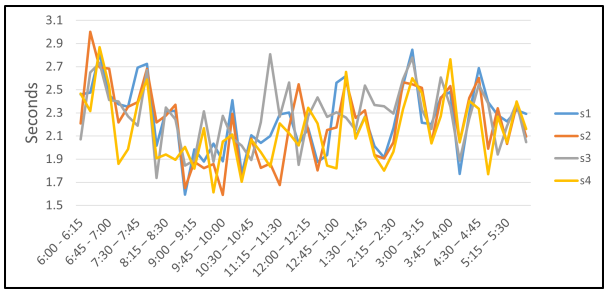
The calibrated values from the genetic algorithm are fairly consistent across random number streams but are generally lower than the calibrated values from the other two optimization techniques. The range for these values seems to generally be from around 1.7 seconds to 2.7 seconds, which is more than a second less than the other two techniques. The maximum allowable velocity for vehicles in these simulations is 34 meters per second, so this generally translates to possibly allowing vehicles to be 34 meters closer than the other two techniques would allow. Though, if the goal of calibration is to fit the real-world data, this might be more accurate, and the calibration accuracy results shown above certainly indicate that the genetic algorithm was the most accurate technique.



(a)



(b)

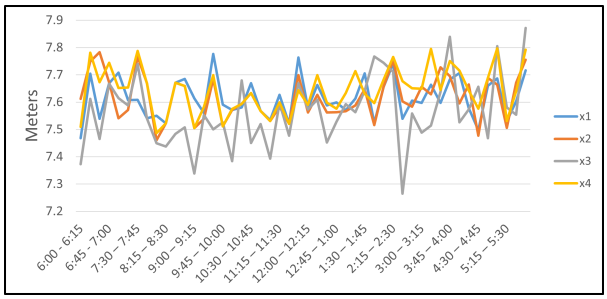


(c)

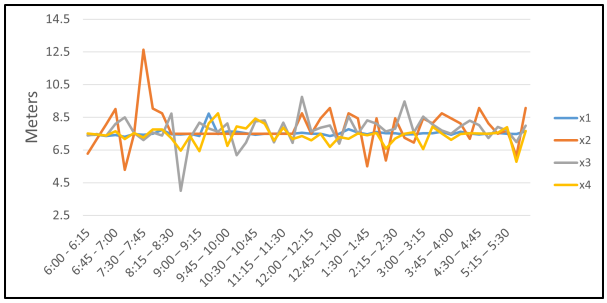
Figure 6.26: a. Calibrated time headway value using Nelder-Mead. b. Calibrated time headway value using SPSA. c. Calibrated time headway value using the GA.

6.5.8 Space Headway

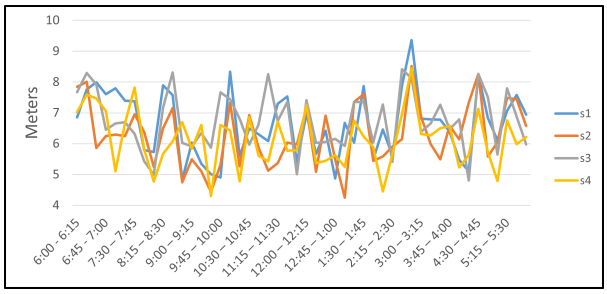
The calibrated parameters for space headway are shown below in Figure 6.27. The Nelder-Mean Simplex found values in a relatively much tighter range than did SPSA or the genetic algorithm, though SPSA was also quite consistent with the calibrated values.



(a)



(b)

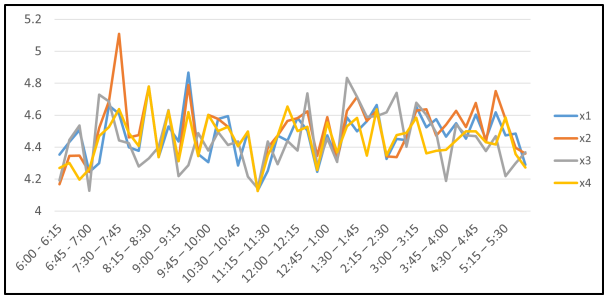


(c)

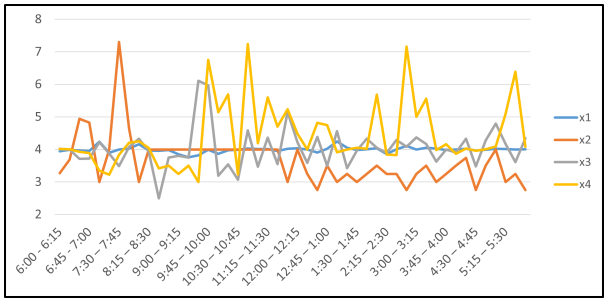
Figure 6.27: a. Calibrated space headway value using Nelder-Mead. b. Calibrated space headway value using SPSA. c. Calibrated space headway value using the GA.

6.5.9 Parameter δ

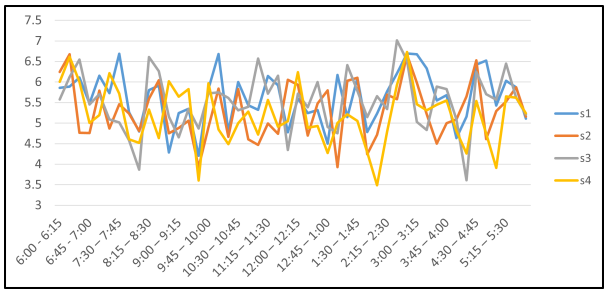
The calibrated values of the δ parameter are shown in Figure 6.28. This parameter is an exponent in the IDM and is included as a tuning parameter. Once again there are large differences between the three algorithms in the values found.



(a)



(b)



(c)

Figure 6.28: a. Calibrated δ value using Nelder-Mead. b. Calibrated δ value using SPSA. c. Calibrated δ value using the GA.

6.6 Comparison with Other Calibration Efforts

There have been many efforts made to calibrate microscopic car-following models. With the many varied car-following models out there and the large number of optimization techniques available, it is rare to find multiple calibration methodologies that are alike. Table 6.16 shows calibration projects along with their choices of performance metric, and the choice of loss function or goodness-of-fit measure.

For comparison to the work in this project, it is appropriate to compare to another model if there is some shared aspect to the method, such as optimization procedure, or car-following model. Table 6.17 shows several other calibration efforts that share at least one aspect of the calibration techniques used in this project.

As can be seen in the table, GAs were by far the most popular optimization algorithm for calibration. This is not surprising given the long history GAs have for robust global optimization. SPSA was also used in several papers and the majority of the authors agree that the main appeal is their fast execution times. Among the studies that compared multiple optimization algorithms, there was large disagreement on which algorithms performed better. Rahman et al. (Rahman et al., 2020) found that in addition to executing much more quickly than a GA and a GA/SQP hybrid model, SPSA was also more accurate in the calibration. Abdalhaq and Baker (Abdalhaq & Baker, 2014) however found that SPSA was very inconsistent, and did not perform as well as a GA or a Particle Swarm. They did not test execution times but did collect data on the total number of executions by each algorithm, and the numbers for SPSA are consistent with it executing more quickly than other methods. They also tested the Nelder-Mead Simplex algorithm but found it to be inaccurate. Paz et al. (Paz et al., 2015) calibrated with both a GA and SPSA, and again found that actual execution times of the SPSA were much quicker than the GA. In fact, when just comparing the average execution times the disparity was profound. They conducted two experiments, one in Reno, NV, and one using synthetic data, and used both algorithms for each one. For the Reno experiment, the GA took 20.2 hours to return its optimal value, while SPSA took only 25.5 minutes. which is a speed gain of around 48 times. For the synthetic data experiment, the GA took 12.8 hours versus just 10 minutes for SPSA, resulting in a speed gain of around 77 times. They do mention however that SPSA took 20 hours to calibrate, due to the high sensitivity of the gain sequence parameters in the algorithm, while the GA took only 1 hour to calibrate. In terms of accuracy, they found that SPSA was slightly more accurate than the GA. For the Reno

Table 6.16: Measure-of-Performance and Goodness-of-Fit Choices From Other Calibration Efforts

Publication	Perforance Metric	Loss Function/GoF
This Project	Vehicle Counts	sMAPE
Pourabdollah et al., 2017	Estimated Power Consumption	NMSE
Rahman et al., 2020	Positions and Velocities	SSE
Treiber and Kesting, 2013	Model Parameters	SSE
Kurtc and Treiber, 2016	Velocities and Gaps	MPSE
Vasconcelos et al., 2014	Gaps	RMSE
Chiappone et al., 2016	Speed-Density	WSSE
Montanino et al., 2012	Counts and Velocities	RMSE, IMSE
Chen et al., 2010	Gaps	Mixed
Van der Horst, 2011	Counts and Velocities	RMSE
Paz et al., 2012	Counts	GEH
Schultz and Rilett, 2004	Counts and Travel Time	MAPE
Yu and Fan, 2017	Flow and Velocity	MANE, GRE, PMAE, PMRE
Ma et al., 2007	Road Capacity and Critical Occupancy	GEH
Balakrishna et al., 2007	Origin-Destination Flows	RMSN, RMSPE, MPE, Theil's U, GEH
Abdalhaq and Baker, 2014	Travel Time	ME
Paz et al., 2015	Counts and Velocities	NRMS, GEH
Hale et al., 2015	Velocity and Density	PE
Menneni et al., 2008	Speed-Flow	Graph Comparison
Park and Qi, 2005	Travel Time	Travel Time Error
Kim and Rilett, 2003	Counts	MAER

experiment, they found the GA and SPSA to have an average accuracy of around 10.8% and 10%, respectively.

Another paper that compared GAs and SPSA, as well as a trial-and-error method, is that of Ma et al. (Ma et al., 2007). Once again it is seen that SPSA performs exceptionally faster than GAs, but in this case, the authors find that SPSA was not as accurate as the GA. They also found that the optimal parameters found by the two methods were very different. They explain that the response surface has a high number of local optima and that SPSA is more likely to fall into one of these than the GA is.

Table 6.17: Car-Following Model Calibration Efforts

	Car-Following			Optimization			
Publication	Gipps	IDM	Other	GA	NM	SPSA	Other
This Project	x	x		x	x	x	
Pourabdollah et al., 2017		x	x	x			
Rahman et al., 2020		x		x		x	x
Treiber and Kesting, 2013		x	x				x
Kurtc and Treiber, 2016		x	x				x
Vasconcelos et al., 2014	x			x			
Chiappone et al., 2016	x			x			
Markou et al., 2019	x		x			x	
Punzo and Tripodi, 2007	x						x
Montanino et al., 2012		x					x
Chen et al., 2010		x	x	x			
Van der Horst, 2011		x	x				x
Fard and Mohaymany, 2019		x		x			x
Ciuffo and Punzo, 2013	x			x		x	
Paz et al., 2012			x			x	
Schultz and Rilett, 2004			x	x			
Yu and Fan, 2017			x	x			x
Ma et al., 2007			x	x		x	x
Balakrishna et al., 2007			x			x	
Abdalhaq and Baker, 2014			x	x	x	x	x
Paz et al., 2015			x	x		x	
Hale et al., 2015			x			x	x
Menneni et al., 2008			x	x			
Park and Qi, 2005			x	x			
Kim and Rilett, 2003			x		x		

Hale et al. (Hale et al., 2015) calibrated with SPSA as well as a Directed Brute Force (DBF) algorithm and found both methods to have reasonable accuracy. They suggest that a hybrid model using the extreme speed of SPSA to find a good neighborhood, and then another algorithm, such as DBF to search in more detail, could result in better accuracy than the two alone can return.

Vasconcelos et al. (Vasconcelos et al., 2014) calibrate Gipps' Model with a GA and used the gap spacing between vehicles as the metric for the objective function, which was RMSE. The parameters they calibrate are the maximum velocity v, maximum acceleration a, maximum deceleration b of the follower,

estimated maximum deceleration d' of the leader, minimum gap s, and driver reaction time τ, which are very similar to the parameters chosen for this project. They report only single values for each parameter which implies that they calibrated the model for their entire time frame, and did not split the time frame into subintervals as was done in this project.

Table 6.18: IDM Calibrated Parameters

			Parameters					
Publication	Method	Road	a	b	V_0	s	T	δ
Rahman et al.	GA	HW	2.78	-2.84	30	4.04	0.71	–
	SQP	–	2.89	-2.77	30	4.71	0.62	–
	SPSA	–	1.01	-1.43	10.2	5.24	0.14	–
Treiber et al.	Local (LM)	Urb	0.51	-1.47	16.9	1.56	1.02	–
	Global (LM)	–	1.46	-0.63	16.1	1.44	1.27	–
Van der Horst	LM	HW	0.67	-0.21	32.51	11.31	1.07	12.23

Table 6.19: Gipps' Model Calibrated Parameters

			Parameters			
Publication	Method	Road	a	b	V_0	τ
Vasconcelos et al.	GA	HW 1	0.82	-2.53	26.0	0.4
		HW 2	9.80	-8.87	24.47	0.5
		Urb 1	1.53	-5.06	11.5	0.75
		Urb 2	1.06	-2.87	13.05	0.79
		Urb 3	1.42	-2.87	18.03	0.82
Markou et al.	SPSA	Urb	0.8	-3.2	14.0	0.4
Punzo et al.	LS	HW (fast)	–	-1.62	23.25	0.2
		HW (slow)	–	-1.1	17.58	0.2

The calibrations presented in Tables 6.18 and 6.19 suggest parameter values over long stretches of time. An assumption made in this paper is that the parameters will change throughout the day as the overall traffic flow changes. While the desire by drivers to move at a certain speed may remain constant throughout the day, their comfortable deceleration value, or their desired time headway, for example, will likely change as the traffic density changes. To this end, the calibrations were made over each 15-minute interval of the period from 6:00 am to 6:00 pm, and so the optimized parameters form more of a distribution or function of values, rather than a single parameter value. Still, it is worth comparing the parameter values from these efforts to those found in this work.

6.7 Conclusions

Two car-following models, the Intelligent Driver Model (Treiber et al., 2000) and Gipps' Model (Gipps, 1981) were calibrated. Three different optimization algorithms, the Nelder-Mead Simplex Algorithm (Nelder & Mead, 1965), the Simultaneous Perturbation Stochastic Approximation Algorithm (Spall et al., 1992), and a genetic algorithm designed and implemented by the author.

The calibrations revealed that in terms of accuracy, the GA outperforms the other two algorithms, with the Nelder-Mead Simplex performing second best, and SPSA finishing in last place. As well, when looking at the improvement of the algorithms over epochs, the GA shows faster improvement in the early stages of the optimization. In terms of calibration efficiency, this could allow the GA to compete with SPSA and the Nelder-Mead Simplex, both of which were much faster in overall computation time. SPSA was by far the fastest algorithm, but also the least accurate. Still, SPSA saw a significant improvement over the uncalibrated model, so with the desire for fast predictions, has a case for being the desired optimization algorithm.

In terms of the car-following models, there was only a slight improvement by Gipps' Model over the IDM, but in a practical sense, they performed very similarly.

It should be noted that tuning the internal parameters of these algorithms was not considered here but could result in large improvements in the performance of each algorithm. Such optimization of the parameters of the optimization algorithms is quite a difficult problem and one that could be investigated in future work. There are also many other calibration techniques that could be applied to this problem, as shown by comparison with the calibration efforts of other researchers.

Chapter 7

Conclusions

Traffic congestion is a ubiquitous problem in the world today and one that most citizens have a strong desire to solve. Vast amounts of traffic flow data have been collected from road networks that allow researchers to develop models for this traffic flow, however, many of these techniques deal only with the macroscopic elements of the traffic dynamics. Microscopic traffic simulation models seek to create behavioral relationships between the vehicles themselves, allowing for greater flexibility in the modeling capabilities, and opening doorways to problem-solving that macroscopic approaches cannot.

This dissertation details much of the work done to create a simulation system and modeling approach for microscopic traffic simulation using Scala-Tion, from the early stages of development, through to the calibration of model parameters and some analysis of the methodologies. Below is a brief summary of each of the chapters, followed by a look at directions for future research.

7.1 Traffic Flow Modeling (Chapter 2)

The earliest work was done in an attempt to optimize the timing of traffic lights at a series of urban traffic intersections in Kenmore, Washington, U.S.A. A general framework for an arrival model was designed and code to generate random arrivals was produced. A car-following model was also introduced to the system which was a mixture of the Intelligent Driver Model for car-following, and a basic physics model for free-driving. A basic framework for modeling a connected road network was presented, including the internal design elements of the traffic signals themselves. An approach to turning behavior based on the traffic data was given. Optimization procedures were utilized to attempt to optimize the signal timings, and some key results emerged. Namely that gradient-based

optimization procedures can have problems in the traffic optimization field. In the case of traffic light timings, this results from the fact that the signal timings should probably be modeled by a discrete space instead of a continuous one.

Still, much was learned from this early work, and a promising basic framework for creating a microscopic traffic simulation system in SCALATION resulted from it. But much more analysis and research of the problem needed to be done.

7.2 Microscopic Traffic Simulation (Chapter 3)

After the initial work of attempting to optimize traffic light timings, it was clear that a deeper dive into car-following models and traffic simulation was needed. A comparison of the three main paradigms of traffic simulation was performed. These are macroscopic, mesoscopic, and microscopic traffic simulations. The various traffic flow models and car-following models that were discovered gave a much greater understanding of the problem as a whole, and where the direction of the research should be focused.

As a point of comparison, a treatment of alternative methods was produced by colleague Hao Peng (2018)(2019), which shows the capabilities of other techniques. Several approaches from deep learning and time series analysis were given.

7.3 Arrival Modeling (Chapter 4)

Time headway is a crucial element of traffic flow modeling, and is directly related to arrival modeling. The horizon of the traffic simulation creates a boundary between what is seen and what is not seen from real-world traffic networks. Sources within the simulation are designed to generate cars for the model, but in the real world, these vehicles would have already existed, so the source must introduce them to the simulation in a realistic way. This includes both time headway, or the amount of time between the front bumpers of vehicles on a road, and the speeds of the vehicles. The first issue is dealt with in this chapter.

Many distributions for time headway have been proposed. One classically used distribution for time headway is the exponential distribution. And if the distribution of arrival times is exponential, then the distribution of arrival counts in a given period of time is Poisson, and the counts are a Poisson counting process. This can be loosened a bit for a nonhomogeneous Poisson process, which allows for a varying arrival count rate. The inversion process of Leemis (1991) was shown which allows arrival count data to be used to recover a realistic

random variable for arrival *times*. This random arrival time generator can then be used to recover the interarrival times which are essentially the time headway values.

That is only half of the story though. Since the purpose of the system is traffic flow forecasting, the generated arrival times must be for *future* time periods, and so the arrival count data at the sources must be forecast by some other means before the simulation can be executed. Several methods for performing the arrival count forecasts were proposed and tested. With an eye toward a continuously updating system in the future, several of the methods were designed to use the most recent possible data to improve the forecasts. The results show clearly that using these data can result in a drastic improvement to the forecasts.

Among these *online* forecasting methods, two were designed with a very simple rate-of-change type calculation, and one used a much more sophisticated SARIMA model. The SARIMA model was clearly the most accurate forecasting method but took significantly longer in its derivation than the other methods, and with efficiency a key concern in any online forecasting system, this implies a potential for the other methods used.

7.4 Car-Following Models (Chapter 5)

The earliest work covered in Chapter 2 partially utilized the Intelligent Driver Model (Treiber et al., 2000) for its car-following capabilities, but there was not much of an analysis done at the time comparing car-following models. In this chapter, a bit more of the background, and in the case of Gipps' Model (1981) a partial derivation, is given.

Some of the early work from the late 1940s and 1950s is introduced. The work of Pipes (1953), Gazis, Herman, and Rothery (1959), (1959) is highlighted as representing foundational work in the field. Other car-following models are mentioned as well.

The focus of the chapter is the Intelligent Driver Model (IDM) and Gipps' Model. The formulas are given, as well as a thorough rundown of their parameters and the meanings of those parameters. Details of the implementations of these models are given in the appendix.

7.5 Calibration (Chapter 6)

The traffic simulation model, built from the arrival model, car-following model, and traffic data, requires a process through which to choose the parameters of the model to best reflect the given data. The general problem of parameter

estimation theoretically employs probability and statistics to analytically arrive at an optimal set or distribution of parameters for a model. However, in the case of an extremely complex system like traffic, it is often the case that analytical solutions are hard to come by, and the process needs to be executed numerically. This approach is a calibration process, and Chapter 6 presents several calibration methodologies for the traffic simulation model, and then compares them.

This type of calibration approach requires the use of a numerical optimization procedure, and three were chosen for this project, with their internal parameters and general algorithms detailed. The three procedures are the Nelder-Mead Simplex Algorithm (Nelder & Mead, 1965), Simultaneous Perturbation Stochastic Approximation Algorithm (SPSA) (Spall et al., 1992), and a self-written Genetic Algorithm, the implementation details of which are presented in the appendix. Nelder-Mead and SPSA require the use of initial points to begin their algorithms, while the GA requires a random number stream for the initialization of the first generation of candidates. Four different initial starting points were used for the first two methods, and four different random number streams were used for the GA.

The calibration approach was to independently calibrate each 15-minute interval of the time period starting at 6:00 am and ending at 6:00 pm for two different Tuesdays in 2018. These were June 26, and July 31. With forty-eight 15-minute intervals in the 12-hour time period, two car-following models, four initial points (or random number streams), four arrival models, three optimization procedures, and two days' worth of data to calibrate, there were 9,216 total calibration executions performed. Results across the various points of comparison were performed.

As a final point of comparison and contrast, some details of other calibration efforts were given at the end of the chapter.

7.6 Future Work

There are many avenues for future research with this starting point. One crucial effort would be to attempt to validate the simulation model using the calibrations performed here. A validation methodology and framework can be implemented and the optimal parameters found in this project can be used to forecast future traffic data. Since the data is a time series, rolling validation can be done on the time period currently calibrated. The calibrations were done for each 15-minute time interval, so the calibrated values could be used for the next 15-minute time interval, or for some other time window just succeeding the cali-

bration window, for short-term validation. Each 15-minute interval calibration can be validated by creating the rolling validation framework.

Since the time series is seasonal, with a period of 168 hours (7 days), validations can also be performed for subsequent corresponding seasons, forecasting a longer time window. There is also the idea that for weekdays there could be a seasonal period of 24 hours so that Tuesday parameters might be able to forecast Wednesday traffic. This idea could definitely be explored. In either situation, an *online* traffic simulation system could use the rolling validation to continuously check the accuracy of the parameters, and if a threshold is passed, re-calibrate the model.

The fact that the parameters were calibrated for each 15-minute interval of the given time period lends itself to a possible parameter function or distribution, instead of just one set of parameter values. Regression could be used to generate essentially a model of the parameters' behavior as the day moves forward. It would be interesting to see if such a general function could be determined, possibly with its own parameters, that would allow for a general parameter model to be developed for the overall simulation model. This would be some form of meta-model that would determine the parameters for the simulation model for each time period.

Another important area of work is to expand the simulated network beyond the fairly small window used in this project. Incorporating on-ramps and off-ramps, and even highway interchanges would be very important additions to the capabilities of the system.

One aspect of microscopic traffic simulation that acts as a major selling point is that it can be used prescriptively to solve traffic design problems. An avenue of research in this direction would be to find a road network with good data that experienced a major change in road design, and see if the simulation system was able to capture the real-world results. In other words, model the system before the design change, implement the change in the simulation, forecast the results of the change, and then compare these to what is actually seen in the post-change data.

There are also many elements of the existing model that can be fine-tuned and elements that can be implemented. For example, currently, no lane-changing model is being utilized, and this could impact the accuracy of the overall simulation model.

Bibliography

Abdalhaq, B. K., & Baker, M. I. A. (2014). Using meta heuristic algorithms to improve traffic simulation. *Journal of Algorithms*, *2*(4), 110–128.

Ahmed, M. S., & Cook, A. R. (1979). *Analysis of freeway traffic time-series data by using box-jenkins techniques*. Transportation Research Board.

Aldrich, J. (1997). Ra fisher and the making of maximum likelihood 1912-1922. *Statistical science*, *12*(3), 162–176.

Al-Ghamdi, A. S. (2001). Analysis of time headways on urban roads: Case study from riyadh. *Journal of Transportation Engineering*, *127*(4), 289–294.

Balakrishna, R., Antoniou, C., Ben-Akiva, M., Koutsopoulos, H. N., & Wen, Y. (2007). Calibration of microscopic traffic simulation models: Methods and application. *Transportation Research Record*, *1999*(1), 198–207.

Bando, M., Hasebe, K., Nakanishi, K., & Nakayama, A. (1998). Analysis of optimal velocity model with explicit delay. *Physical Review E*, *58*(5), 5429.

Bando, M., Hasebe, K., Nakayama, A., Shibata, A., & Sugiyama, Y. (1995). Dynamical model of traffic congestion and numerical simulation. *Physical review E*, *51*(2), 1035.

Barton, R. R., & Ivey Jr, J. S. (1996). Nelder-mead simplex modifications for simulation optimization. *Management Science*, *42*(7), 954–973.

Barua, S., Das, A., & Roy, K. C. (2015). Estimation of traffic arrival pattern at signalized intersection using arima model. *International Journal of Computer Applications*, *128*(1), 1–6.

Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011). Sumo–simulation of urban mobility: An overview. *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*.

Bhaskar, A., & Chung, E. (2013). Fundamental understanding on the use of bluetooth scanner as a complementary transport data. *Transportation Research Part C: Emerging Technologies*, *37*, 42–72.

Box, G. E., & Jenkins, G. M. (1962). Some statistical aspects of adaptive optimization and control. *Journal of the Royal Statistical Society: Series B (Methodological)*, *24*(2), 297–331.

Box, G. E., & Jenkins, G. M. (1970). *Time series analysis forecasting and control* (tech. rep.). DTIC Document.

Box, G. E., Jenkins, G. M., & Bacon, D. W. (1967). *Models for forecasting seasonal and non-seasonal time series.* (tech. rep.). WISCONSIN UNIV MADISON DEPT OF STATISTICS.

Box, G., & Wilson, K. (1951). On the experimental attainment of optimum conditions. *Journal of the Royal Statistical Society. Series B (Methodological)*, *13*(1), 1–45.

Brackstone, M., & McDonald, M. (1999). Car-following: A historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, *2*(4), 181–196.

Branston, D. (1976). Models of single lane time headway distributions. *Transportation Science*, *10*(2), 125–148.

Bratley, P., Fox, B. L., & Schrage, L. E. (2011). *A guide to simulation*. Springer Science & Business Media.

Broyden, C. G. (1970). The convergence of a class of double-rank minimization algorithms 1. general considerations. *IMA Journal of Applied Mathematics*, *6*(1), 76–90.

Buckley, D. (1968). A semi-poisson model of traffic flow. *Transportation Science*, *2*(2), 107–133.

Burghout, W., Koutsopoulos, H. N., & Andreasson, I. (2006). A discrete-event mesoscopic traffic simulation model for hybrid traffic simulation. *Intelligent Transportation Systems Conference, 2006. ITSC'06. IEEE*, 1102–1107.

Buss, A., & Al Rowaei, A. (2010). A comparison of the accuracy of discrete event and discrete time. In B. Johansson, S. Jain, J. Montoya-Torres, J. Hugan, & E. Yücesan (Eds.), *Proceedings of the 2010 winter simulation conference* (pp. 1468–1477). Institute of Electrical; Electronics Engineers, Inc.

California-DMV. (2022). *California driver's handbook*. California Department of Motor Vehicles. Sacremento, CA. https://driving-tests.org/wp-content/uploads/2022/06/CA_DL-600-R7-2022.pdf

Cervero, R. (2013). Linking urban transport and land use in developing countries. *Journal of transport and land use*, *6*(1), 7–24.

Chandler, R. E., Herman, R., & Montroll, E. W. (1958). Traffic dynamics: Studies in car following. *Operations research*, *6*(2), 165–184.

Chen, C., Li, L., Hu, J., & Geng, C. (2010). Calibration of mitsim and idm car-following model based on ngsim trajectory datasets. *Proceedings of 2010 IEEE International Conference on Vehicular Electronics and Safety*, 48–53.

Chiappone, S., Giuffrè, O., Granà, A., Mauro, R., & Sferlazza, A. (2016). Traffic simulation models calibration using speed–density relationship: An automated procedure based on genetic algorithm. *Expert Systems with Applications*, *44*, 147–155.

Chien, S. I.-J., & Kuchipudi, C. M. (2003). Dynamic travel time prediction with real-time and historic data. *Journal of transportation engineering*, *129*(6), 608–616.

Chiou, J.-M. et al. (2012). Dynamical functional prediction and classification, with application to traffic flow prediction. *The Annals of Applied Statistics*, *6*(4), 1588–1614.

Cinlar, E. (1975). Introduction to stochastic processes.

Ciuffo, B., & Punzo, V. (2013). No free lunch" theorems applied to the calibration of traffic simulation models. *IEEE Transactions on Intelligent Transportation Systems*, *15*(2), 553–562.

Ciuffo, B., Punzo, V., & Montanino, M. (2012). Thirty years of gipps' car-following model: Applications, developments, and new features. *Transportation Research Record: Journal of the Transportation Research Board*, *2315*, 89–99.

Council, D. C. (2016). *Journey times across dublin city, from dublin city council traffic department's trips system*. Retrieved August 16, 2016, from https://data.dublinked.ie/dataset/journey-times-across-dublin-city-from-dublin-city-council-traffic-departments-trips-system

Cremer, M., & Ludwig, J. (1986). A fast simulation model for traffic flow on the basis of boolean operations. *Mathematics and Computers in Simulation*, *28*(4), 297–303.

Daganzo, C. F. (1994). The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory. *Transportation Research Part B: Methodological*, *28*(4), 269–287.

Daganzo, C. F. (1995). Requiem for second-order fluid approximations of traffic flow. *Transportation Research Part B: Methodological*, *29*(4), 277–286.

Daganzo, C. F. (1997). A continuum theory of traffic dynamics for freeways with special lanes. *Transportation Research Part B: Methodological*, *31*(2), 83–102.

Daganzo, C. F. (2002). A behavioral theory of multi-lane traffic flow. part i: Long homogeneous freeway sections. *Transportation Research Part B: Methodological*, *36*(2), 131–158.

Dengiz, B., & Alabas, C. (2000). Simulation optimization using tabu search. *2000 Winter Simulation Conference Proceedings (Cat. No. 00CH37165)*, *1*, 805–810.

Dey, P. P., & Chandra, S. (2009). Desired time gap and time headway in steady-state car-following on two-lane roads. *Journal of transportation engineering*, *135*(10), 687–693.

Dougherty, M. (1995). A review of neural networks applied to transport. *Transportation Research Part C: Emerging Technologies*, *3*(4), 247–260.

Edie, L. C., & Foote, R. S. (1961). Experiments on single lane flow in tunnels. *Theory of Traffic Flow Proceedings of the Theory of Traffic Flow*, 175–192.

El Faouzi, N.-E., Klein, L., & De Mouzon, O. (2009). Improving travel time estimates from inductive loop and toll collection data with dempster-shafer data fusion. *Transportation Research Record: Journal of the Transportation Research Board*, *2129*, 73–80.

Esser, J., & Schreckenberg, M. (1997). Microscopic simulation of urban traffic based on cellular automata. *International Journal of Modern Physics C*, *8*(05), 1025–1036.

Ezzat, A. A., Farouk, H. A., El-Kilany, K. S., & Moneim, A. F. A. (n.d.). Optimization using simulation of traffic light signal timings.

Fard, M. R., & Mohaymany, A. S. (2019). A copula-based estimation of distribution algorithm for calibration of microscopic traffic models. *Transportation Research Part C: Emerging Technologies*, *98*, 449–470.

Fernandes, P., & Nunes, U. (2010). Platooning of autonomous vehicles with intervehicle communications in sumo traffic simulator. *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 1313–1318.

Fisher, A. (2013). Google's road map to global domination. *New York Times*, *11*.

Fletcher, R. (1970). A new approach to variable metric algorithms. *The computer journal*, *13*(3), 317–322.

Florian, M., Mahut, M., & Tremblay, N. (2008). Application of a simulation-based dynamic traffic assignment model. *European Journal of Operational Research*, *189*(3), 1381–1392.

Flötteröd, G., & Rohde, J. (2011). Operational macroscopic modeling of complex urban road intersections. *Transportation Research Part B: Methodological*, *45*(6), 903–922.

Fu, M. C., Glover, F. W., & April, J. (2005). Simulation optimization: A review, new developments, and applications. *Proceedings of the Winter Simulation Conference, 2005.*, 13–pp.

Fujimoto, R. (2015). Parallel and distributed simulation. In L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, & M. D. Rossetti (Eds.), *Proceedings of the 2015 winter simulation conference* (pp. 45–59). Institute of Electrical; Electronics Engineers, Inc.

Gazis, D. C., Herman, R., & Potts, R. B. (1959). Car-following theory of steady-state traffic flow. *Operations research*, *7*(4), 499–505.

Gazis, D. C., Herman, R., & Rothery, R. W. (1961). Nonlinear follow-the-leader models of traffic flow. *Operations research*, *9*(4), 545–567.

Georgia-DDS. (2022). *2021-2022 40-hour parent/teen driving guide*. Georgia Department of Driver Services. Atlanta, GA. https://dds.georgia.gov/document/publication/40-hour-parent-teen-driver-guide/download

Gerlough, D. L. (1956). Simulation of freeway traffic by an electronic computer. *Highway Research Board Proceedings*, *35*.

Gipps, P. G. (1981). A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, *15*(2), 105–111.

Gipps, P. G. (1986). A model for the structure of lane-changing decisions. *Transportation Research Part B: Methodological*, *20*(5), 403–414.

Glover, F. (1989). Tabu search—part i. *ORSA Journal on computing*, *1*(3), 190–206.

Goldfarb, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of computation*, *24*(109), 23–26.

Guo, J., Huang, W., & Williams, B. M. (2014). Adaptive kalman filter approach for stochastic short-term traffic flow rate prediction and uncertainty quantification. *Transportation Research Part C: Emerging Technologies*, *43*, 50–64.

Hale, D. K., Antoniou, C., Brackstone, M., Michalaka, D., Moreno, A. T., & Parikh, K. (2015). Optimization-based assisted calibration of traffic simulation models. *Transportation Research Part C: Emerging Technologies*, *55*, 100–115.

Hanai, M., Suzumura, T., Theodoropoulos, G., & Perumalla, K. S. (2015). Towards large-scale what-if traffic simulation with exact-differential simulation. In L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, & M. D. Rossetti (Eds.), *Proceedings of the 2015 winter simulation conference* (pp. 748–756). Institute of Electrical; Electronics Engineers, Inc.

Hasebe, K., Nakayama, A., & Sugiyama, Y. (2003). Dynamical model of a cooperative driving system for freeway traffic. *Physical review E*, *68*(2), 026102.

Helbing, D. (1997). Modeling multi-lane traffic flow with queuing effects. *Physica A: Statistical Mechanics and its Applications*, *242*(1-2), 175–194.

Helbing, D., & Schreckenberg, M. (1999). Cellular automata simulating experimental properties of traffic flow. *Physical review E*, *59*(3), R2505.

Helbing, D., & Tilch, B. (1998). Generalized force model of traffic dynamics. *Physical Review E*, *58*(1), 133.

Hellinga, B. R. (1998). Requirements for the calibration of traffic simulation models. *Proceedings of the Canadian Society for Civil Engineering*, *4*, 211–222.

Helly, W. (1959). Simulation of bottlenecks in single-lane traffic flow.

Henclewood, D., Suh, W., Rodgers, M., Hunter, M., & Fujimoto, R. (2012). A case for real-time calibration of data-driven microscopic traffic simulation tools. In C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, & A. M. Uhrmacher (Eds.), *Proceedings of the 2012 winter simulation conference* (pp. 1670–1681). Institute of Electrical; Electronics Engineers, Inc.

Herman, R., Montroll, E. W., Potts, R. B., & Rothery, R. W. (1959). Traffic dynamics: Analysis of stability in car following. *Operations research*, *7*(1), 86–106.

Herrera, J. C., Work, D. B., Herring, R., Ban, X. J., Jacobson, Q., & Bayen, A. M. (2010). Evaluation of traffic data obtained via gps-enabled mobile phones: The mobile century field experiment. *Transportation Research Part C: Emerging Technologies*, *18*(4), 568–583.

Herring, R., Hofleitner, A., Amin, S., Nasr, T., Khalek, A., Abbeel, P., & Bayen, A. (2010). Using mobile phones to forecast arterial traffic through statistical learning. *89th Transportation Research Board Annual Meeting*, 10–2493.

Holland, J. H. (1992). *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*. MIT press.

Hoogendoorn, S., Van Zuylen, H., Schreuder, M., Gorte, B., & Vosselman, G. (2003). Microscopic traffic data collection by remote sensing. *Transportation Research Record: Journal of the Transportation Research Board*, *1855*, 121–128.

Hoogendoorn, S. P., & Bovy, P. H. (2001). Generic gas-kinetic traffic systems modeling with applications to vehicular traffic flow. *Transportation Research Part B: Methodological*, *35*(4), 317–336.

Hunt, P., Robertson, D., Bretherton, R., & Royle, M. C. (1982). The scoot on-line traffic signal optimisation technique. *Traffic Engineering & Control*, *23*(4), 190–192.

Hunter, M. P., Fujimoto, R. M., Suh, W., & Kim, H. K. (2006). An investigation of real-time dynamic data driven transportation simulation. *Proceedings of the 2006 Winter Simulation Conference*, 1414–1421.

Hyndman, R. J., & Athanasopoulos, G. (2014). *Forecasting: Principles and practice*. OTexts.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice*. OTexts.

Ishikawa, S., & Arai, S. (2015). Evaluating advantage of sharing information among vehicles toward avoiding phantom traffic jam. In L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, & M. D. Rossetti (Eds.), *Proceedings of the 2015 winter simulation conference* (pp. 300–311). Institute of Electrical; Electronics Engineers, Inc.

Jiang, R., Wu, Q., & Zhu, Z. (2001). Full velocity difference model for a car-following theory. *Physical Review E*, *64*(1), 017101.

Kalman, R. E. et al. (1960). A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, *82*(1), 35–45.

Kanezashi, H., & Suzumura, T. (2015). Performance optimization for agent-based traffic simulation by dynamic agent assignment. In L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, & M. D. Rossetti (Eds.), *Proceedings of the 2015 winter simulation conference* (pp. 757–766). Institute of Electrical; Electronics Engineers, Inc.

Kao, E. P., & Chang, S.-L. (1988). Modeling time-dependent arrivals to service systems: A case in using a piecewise-polynomial rate function in a nonhomogeneous poisson process. *Management Science*, *34*(11), 1367–1379.

Karlaftis, M. G., & Vlahogianni, E. I. (2011). Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, *19*(3), 387–399.

Kesting, A., Treiber, M., & Helbing, D. (2010). Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *368*(1928), 4585–4605.

Kim, K.-O., & Rilett, L. (2003). Simplex-based calibration of traffic microsimulation models with intelligent transportation systems data. *Transportation Research Record*, *1855*(1), 80–89.

Klein, R. W., & Roberts, S. D. (1984). A time-varying poisson arrival process generator. *Simulation*, *43*(4), 193–195.

Kolmogorov, A. (1933). Sulla determinazione empirica di una lgge di distribuzione. *Inst. Ital. Attuari, Giorn.*, *4*, 83–91.

Kometani, E., & Sasaki, T. (1961). Dynamic behaviour of traffic with a nonlinear spacing–speed relationship. *Proceedings of Symposium on Theory of Traffic Flow*, 105–119.

Krajzewicz, D., Erdmann, J., Behrisch, M., & Bieker, L. (2012). Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, *5*(3&4), 128–138.

Kumar, S. V., & Vanajakshi, L. (2015). Short-term traffic flow prediction using seasonal arima model with limited input data. *European Transport Research Review*, *7*(3), 1–9.

Kurtc, V., & Treiber, M. (2016). Calibrating the local and platoon dynamics of car-following models on the reconstructed ngsim data. *Traffic and granular flow'15* (pp. 515–522). Springer.

Law, A. M., Kelton, W. D., & Kelton, W. D. (2007). *Simulation modeling and analysis* (Vol. 3). Mcgraw-hill New York.

Leclercq, L. (2007). Hybrid approaches to the solutions of the "lighthill–whitham–richards" model. *Transportation Research Part B: Methodological*, *41*(7), 701–709.

Leduc, G. (2008). Road traffic data: Collection methods and applications. *Working Papers on Energy, Transport and Climate Change*, *1*(55).

Leemis, L. (2003). Estimating and simulating nonhomogeneous poisson processes. *Williamsburg, VA: William and Mary Mathematics Department*.

Leemis, L. M. (1991). Nonparametric estimation of the cumulative intensity function for a nonhomogeneous poisson process. *Management Science*, *37*(7), 886–900.

Leemis, L. M. (2004). Nonparametric estimation and variate generation for a nonhomogeneous poisson process from event count data. *IIE Transactions*, *36*(12), 1155–1160.

Leland, W. E., Taqqu, M. S., Willinger, W., & Wilson, D. V. (1994). On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on networking*, *2*(1), 1–15.

Levin, D. (2015). Googles self-driving cars are coming sooner than you think.

Levin, M., & Tsao, Y.-D. (1980). On forecasting freeway occupancies and volumes (abridgment). *Transportation Research Record*, (773).

Li, L., & Chen, X. M. (2017). Vehicle headway modeling and its inferences in macroscopic/microscopic traffic flow theory: A survey. *Transportation Research Part C: Emerging Technologies*, *76*, 170–188.

Li, L., Chen, X. M., & Zhang, L. (2016). A global optimization algorithm for trajectory data based car-following model calibration. *Transportation Research Part C: Emerging Technologies*, *68*, 311–332.

Li, L., Fa, W., Rui, J., Jian-Ming, H., & Yan, J. (2010). A new car-following model yielding log-normal type headways distributions. *Chinese Physics B*, *19*(2), 020513.

Li, Y., Sun, D., Liu, W., Zhang, M., Zhao, M., Liao, X., & Tang, L. (2011). Modeling and simulation for microscopic traffic flow based on multiple headway, velocity and acceleration difference. *Nonlinear Dynamics*, *66*(1), 15–28.

Lieberman, E., & Rathi, A. K. (1997). Traffic simulation. *Traffic flow theory*.

Lighthill, M. J., & Whitham, G. B. (1955). On kinematic waves. ii. a theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, *229*(1178), 317–345.

Lippi, M., Bertini, M., & Frasconi, P. (2013). Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning. *IEEE Transactions on Intelligent Transportation Systems*, *14*(2), 871–882.

Lo, C.-H., Peng, W.-C., Chen, C.-W., Lin, T.-Y., & Lin, C.-S. (2008). Carweb: A traffic data collection platform. *Mobile Data Management, 2008. MDM'08. 9th International Conference on*, 221–222.

Ma, J., Dong, H., & Zhang, H. M. (2007). Calibration of microsimulation with heuristic optimization methods. *Transportation Research Record*, *1999*(1), 208–217.

Mahnke, R., & Kühne, R. (2007). Probabilistic description of traffic breakdown. *Traffic and Granular Flow'05*, 527–536.

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*(1), 54–74.

Mannering, F. (2007). Effects of interstate speed limits on driving speeds: Some new evidence. *Compendium of papers CD-ROM, Transportation Research Board 86th Annual Meeting*.

Markou, I., Papathanasopoulou, V., & Antoniou, C. (2019). Dynamic car–following model calibration using spsa and isres algorithms. *Periodica Polytechnica Transportation Engineering*, *47*(2), 146–156.

Matcha, B., Namasivayam, S., Ng, K., Sivanesan, S., & EhNoum, S. (2021). Mixed-traffic vehicle dynamics: Analysis and comparison of microscopic gipp's car-following model and intelligent driver model. *Advances in transportation studies*, *55*.

Meng, D., Song, G., Wu, Y., Zhai, Z., Yu, L., & Zhang, J. (2021). Modification of newell's car-following model incorporating multidimensional stochastic parameters for emission estimation. *Transportation Research Part D: Transport and Environment*, *91*, 102692.

Meng, Q., & Khoo, H. L. (2009). Self-similar characteristics of vehicle arrival pattern on highways. *Journal of Transportation Engineering*, *135*(11), 864–872.

Menneni, S., Sun, C., & Vortisch, P. (2008). Microsimulation calibration using speed-flow relationships. *Transportation Research Record*, *2088*(1), 1–9.

Miller, J. A., Han, J., & Hybinette, M. (2010). Using domain specific language for modeling and simulation: Scalation as a case study. *Proceedings of the 2010 Winter Simulation Conference*, 741–752.

Missouri-DOR. (2022). *2022 missouri driver guide*. Missouri Department of Revenue. Jefferson City, MO. https://dor.mo.gov/forms/Driver%20Guide.pdf

Montanino, M., Ciuffo, B., & Punzo, V. (2012). Calibration of microscopic traffic flow models against time-series data. *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 108–114.

Mori, U., Mendiburu, A., Álvarez, M., & Lozano, J. A. (2015). A review of travel time estimation and forecasting for advanced traveller information systems. *Transportmetrica A: Transport Science*, *11*(2), 119–157.

Nagel, K., & Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de physique I*, *2*(12), 2221–2229.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308–313.

Newell, G. F. (1961). Nonlinear effects in the dynamics of car following. *Operations research*, *9*(2), 209–229.

Nikovski, D., Nishiuma, N., Goto, Y., & Kumazawa, H. (2005). Univariate short-term prediction of road travel times. *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, 1074–1079.

of Transportation, C. D. (2020). Pems user guide.

of Transportation, C. D. (n.d.). Pems.

Okutani, I., & Stephanedes, Y. J. (1984). Dynamic prediction of traffic volume through kalman filtering theory. *Transportation Research Part B: Methodological*, *18*(1), 1–11.

Osorio, C., & Chong, L. (2012). An efficient simulation-based optimization algorithm for large-scale transportation problems. In C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, & A. M. Uhrmacher (Eds.), *Proceedings of the 2012 winter simulation conference* (pp. 3916–3926). Institute of Electrical; Electronics Engineers, Inc.

Park, B., & Qi, H. (2005). Development and evaluation of a procedure for the calibration of simulation models. *Transportation Research Record*, *1934*(1), 208–217.

Paveri-Fontana, S. (1975). On boltzmann-like treatments for traffic flow: A critical review of the basic model and an alternative proposal for dilute traffic analysis. *Transportation research*, *9*(4), 225–235.

Paz, A., Molano, V., & Gaviria, C. (2012). Calibration of corsim models considering all model parameters simultaneously. *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 1417–1422.

Paz, A., Molano, V., Martinez, E., Gaviria, C., & Arteaga, C. (2015). Calibration of traffic flow models using a memetic algorithm. *Transportation Research Part C: Emerging Technologies*, *55*, 432–443.

Peng, H., Bobade, S. U., Cotterell, M. E., & Miller, J. A. (2018). Forecasting traffic flow: Short term, long term, and when it rains. *International Conference on Big Data*, 57–71.

Peng, H., Klepp, N., Toutiaee, M., Arpinar, I. B., & Miller, J. A. (2019). Knowledge and situation-aware vehicle traffic forecasting. *2019 IEEE International Conference on Big Data (Big Data)*, 3803–3812.

Pereira, J. L., & Rossetti, R. J. (2012). An integrated architecture for autonomous vehicles simulation. *Proceedings of the 27th annual ACM symposium on applied computing*, 286–292.

Petersen, K. B., Pedersen, M. S. et al. (2008). The matrix cookbook. *Technical University of Denmark*, *7*(15), 510.

Pipes, L. A. (1953). An operational analysis of traffic dynamics. *Journal of applied physics*, *24*(3), 274–281.

Pourabdollah, M., Bjärkvik, E., Fürer, F., Lindenberg, B., & Burgdorf, K. (2017). Calibration and evaluation of car following models using real-world driving data. *2017 IEEE 20th International conference on intelligent transportation systems (ITSC)*, 1–6.

Prigogine, I., & Andrews, F. C. (1960). A boltzmann-like approach for traffic flow. *Operations Research*, *8*(6), 789–797.

Punzo, V., & Tripodi, A. (2007). Steady-state solutions and multiclass calibration of gipps microscopic traffic flow model. *Transportation Research Record*, *1999*(1), 104–114.

Radwan, E., Elahi, M., & Goul, M. (1990). Knowledge based systems applications for traffic signal control. *OECD Workshop on Knowledge-Based Expert Systems in Transportation Part 1 (of 2)*.

Rahman, M. M., Ismail, M. T., & Ali, M. K. M. (2020). Comparing the calibration methods for intelligent driver model using beijing data. *International Journal of Vehicle Systems Modelling and Testing*, *14*(4), 215–231.

Ran, B., Jin, P. J., Boyce, D., Qiu, T. Z., & Cheng, Y. (2012). Perspectives on future transportation research: Impact of intelligent transportation system technologies on next-generation transportation modeling. *Journal of Intelligent Transportation Systems*, *16*(4), 226–242.

Richard, I. (2021). Tesla model s plaid: Peak acceleration is at 1.2 g's, elon musk says its 'faster than failing'.

Richards, P. I. (1956). Shock waves on the highway. *Operations research*, *4*(1), 42–51.

Salimifard, K., & Ansari, M. (2013). Modeling and simulation of urban traffic signals. *International Journal of Modeling and Optimization*, *3*(2), 172.

Schimbinschi, F., Moreira-Matias, L., Nguyen, V. X., & Bailey, J. (2017). Topology-regularized universal vector autoregression for traffic forecasting in large urban areas. *Expert Systems with Applications*.

Schimbinschi, F., Nguyen, X. V., Bailey, J., Leckie, C., Vu, H., & Kotagiri, R. (2015). Traffic forecasting in complex urban networks: Leveraging big data and machine learning. *Big Data (Big Data), 2015 IEEE International Conference on*, 1019–1024.

Schultz, G. G., & Rilett, L. (2004). Analysis of distribution and calibration of car-following sensitivity parameters in microscopic traffic simulation models. *Transportation Research Record*, *1876*(1), 41–51.

Schulze, T., & Fliess, T. (1997). Urban traffic simulation with psycho-physical vehicle-following models. *Proceedings of the 29th conference on Winter simulation*, 1222–1229.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2010). Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *40*(1), 185–197.

Semarak, J. (1996). Intelligent traffic lights control by fuzzy logic. *Malaysian Journal of Computer Science*, *9*(2), 29–35.

Shanno, D. F. (1970). Conditioning of quasi-newton methods for function minimization. *Mathematics of computation*, *24*(111), 647–656.

Sichitiu, M. L., & Kihl, M. (2008). Inter-vehicle communication systems: A survey. *IEEE Communications Surveys & Tutorials*, *10*(2).

Smirnov, N. (1948). Table for estimating the goodness of fit of empirical distributions. *The annals of mathematical statistics*, *19*(2), 279–281.

Smith, B. L., Williams, B. M., & Oswald, R. K. (2002). Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C: Emerging Technologies*, *10*(4), 303–321.

Spall, J. C. et al. (1992). Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, *37*(3), 332–341.

Spall, J. C. (1998a). Implementation of the simultaneous perturbation algorithm for stochastic optimization. *IEEE Transactions on aerospace and electronic systems*, *34*(3), 817–823.

Spall, J. C. (1998b). An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins apl technical digest*, *19*(4), 482–492.

Spall, J. C., & Chin, D. C. (1997). Traffic-responsive signal timing for system-wide traffic control. *Transportation Research Part C: Emerging Technologies*, *5*(3-4), 153–163.

Stathopoulos, A., & Karlaftis, M. G. (2003). A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C: Emerging Technologies*, *11*(2), 121–135.

Stigler, S. M. (1981). Gauss and the invention of least squares. *the Annals of Statistics*, 465–474.

Suh, W., Henclewood, D., Guin, A., Guensler, R., Hunter, M., & Fujimoto, R. (2017). Dynamic data driven transportation systems. *Multimedia Tools and Applications*, 1–17.

Sumaryo, S., Halim, A., & Ramli, K. (2013). Improved discrete event simulation model of traffic light control on a single intersection. *QiR (Quality in Research), 2013 International Conference on*, 116–120.

Sun, S., Zhang, C., & Yu, G. (2006). A bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems*, *7*(1), 124–132.

Sussman, J. S. (2008). *Perspectives on intelligent transportation systems (its)*. Springer Science & Business Media.

Suzumura, T., & Kanezashi, H. (2013). A holistic architecture for super real-time multiagent simulation platforms. In R. P. S.-H. K. A. T. R. Hill &

M. E. Kuhl (Eds.), *Proceedings of the 2013 winter simulation conference* (pp. 1604–1612). Institute of Electrical; Electronics Engineers, Inc.

Tampère, C. M., Corthout, R., Cattrysse, D., & Immers, L. H. (2011). A generic class of first order node models for dynamic macroscopic simulation of traffic flows. *Transportation Research Part B: Methodological*, *45*(1), 289–309.

Thulasidasan, S., & Eidenbenz, S. (2009). Accelerating traffic microsimulations: A parallel discrete-event queue-based approach for speed and scale. In M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, & R. G. Ingalls (Eds.), *Proceedings of the 2009 winter simulation conference* (pp. 2457–2466). Institute of Electrical; Electronics Engineers, Inc.

Thulasidasan, S., Kasiviswanathan, S., Eidenbenz, S., Galli, E., Mniszewski, S., & Romero, P. (2009). Designing systems for large-scale, discrete-event simulations: Experiences with the fasttrans parallel microsimulator. *High Performance Computing (HiPC), 2009 International Conference on*, 428–437.

Touhbi, S., Babram, M. A., Nguyen-Huu, T., Marilleau, N., Hbid, M. L., Cambier, C., & Stinckwich, S. (2018). Time headway analysis on urban roads of the city of marrakesh. *Procedia computer science*, *130*, 111–118.

Treiber, M., Hennecke, A., & Helbing, D. (2000). Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, *62*(2), 1805.

Treiber, M., & Kesting, A. (2013). Microscopic calibration and validation of car-following models–a systematic approach. *Procedia-Social and Behavioral Sciences*, *80*, 922–939.

Van Aerde, M., Hellinga, B., Baker, M., & Rakha, H. (1996). Integration: An overview of traffic simulation features. *Transportation Research Records*.

Van Aerde, M., & Yagar, S. (1988). Dynamic integrated freeway/traffic signal networks: A routing-based modelling approach. *Transportation Research Part A: General*, *22*(6), 445–453.

Van der Horst, A. (2011). Calibration of the idm and metanet traffic flow models. *Delft University*.

van Wageningen-Kessels, F., Van Lint, H., Vuik, K., & Hoogendoorn, S. (2015). Genealogy of traffic flow models. *EURO Journal on Transportation and Logistics*, *4*(4), 445–473.

Vasconcelos, L., Neto, L., Santos, S., Silva, A. B., & Seco, Á. (2014). Calibration of the gipps car-following model using trajectory data. *Transportation Research Procedia*, *3*, 952–961.

Vlahogianni, E. I. (2015). Computational intelligence and optimization for transportation big data: Challenges and opportunities. *Engineering and applied sciences optimization* (pp. 107–128). Springer.

Vlahogianni, E. I., Golias, J. C., & Karlaftis, M. G. (2004). Short-term traffic forecasting: Overview of objectives and methods. *Transport reviews*, *24*(5), 533–557.

Vlahogianni, E. I., Karlaftis, M. G., & Golias, J. C. (2014). Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, *43*, 3–19.

Wang, L. (2005). A hybrid genetic algorithm–neural network strategy for simulation optimization. *Applied Mathematics and Computation*, *170*(2), 1329–1343.

Wang, X. (2005). Integrating gis, simulation models, and visualization in traffic impact analysis. *Computers, Environment and Urban Systems*, *29*(4), 471–496.

Werbos, P. (1974). Beyond regression:" new tools for prediction and analysis in the behavioral sciences. *Ph. D. dissertation, Harvard University*.

Whittle, P. (1951). *Hypothesis testing in time series analysis* (Vol. 4). Almqvist & Wiksells boktr.

Wiedemann, R. (1974). Simulation des strassenverkehrsflusses. *Schriftenreihe des Instituts fuer Verkehrswesen der Universitaet Karlsruhe (*, (8).

Wiedemann, R. (1991). Modelling of rti-elements on multi-lane roads. *Drive Conference (1991: Brussels, Belgium)*, *2*.

Wilkie, D., Sewall, J., & Lin, M. C. (2012). Transforming gis data into functional road models for large-scale traffic simulation. *IEEE transactions on visualization and computer graphics*, *18*(6), 890–901.

Wilkinson, R. I. (1956). Theories for toll traffic engineering in the usa. *Bell System Technical Journal*, *35*(2), 421–514.

Williams, B., Durvasula, P., & Brown, D. (1998). Urban freeway traffic flow prediction: Application of seasonal autoregressive integrated moving average and exponential smoothing models. *Transportation Research Record: Journal of the Transportation Research Board*, (1644), 132–141.

Williams, B. M., & Hoel, L. A. (2003). Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, *129*(6), 664–672.

Wong, G., & Wong, S. (2002). A multi-class traffic flow model–an extension of lwr model with heterogeneous drivers. *Transportation Research Part A: Policy and Practice*, *36*(9), 827–841.

Yang, H., Dillon, T. S., & Chen, Y.-P. P. (2015). Evaluation of recent computational approaches in short-term traffic forecasting. *IFIP International Conference on Artificial Intelligence in Theory and Practice*, 108–116.

Ye, F., & Zhang, Y. (2009). Vehicle type–specific headway analysis using freeway traffic data. *Transportation research record*, *2124*(1), 222–230.

Yu, M., & Fan, W. D. (2017). Calibration of microscopic traffic simulation models using metaheuristic algorithms. *International Journal of Transportation Science and Technology*, *6*(1), 63–77.

Zehe, D., Cai, W., Knoll, A., & Aydt, H. (2015). Tutorial on a modeling and simulation cloud service. In L. Yilmaz, W. K. V. Chan, I. Moon, T. M. K. Roeder, C. Macal, & M. D. Rossetti (Eds.), *Proceedings of the 2015 winter simulation conference* (pp. 103–114). Institute of Electrical; Electronics Engineers, Inc.

Zhang, X., & Rice, J. A. (2003). Short-term travel time prediction. *Transportation Research Part C: Emerging Technologies*, *11*(3), 187–210.

Zhou, X., & Taylor, J. (2014). Dtalite: A queue-based mesoscopic traffic simulator for fast model evaluation and calibration. *Cogent Engineering*, *1*(1), 961345.

Zozaya-Gorostiza, C., & Hendrickson, C. (1987). Expert system for traffic signal setting assistance. *Journal of Transportation Engineering*, *113*(2), 108–126.

Appendix A

Appendix

A.1 Parameter Estimation

A.1.1 Maximum Likelihood Estimation

One technique for parameter estimation is Maximum Likelihood estimation, which is credited to Fisher (Aldrich, 1997). This assumes the data fit some joint probability distribution. Since the data from the system is assumed to rely on the value of the parameters, the goal is to estimate the values of the parameters such that the probability of observing this data sample is maximized. Let $f(\mathbf{y};\boldsymbol{\theta})$ be the probability density function for the distribution, and let $\mathbf{y} = (y_1, y_2, ..., y_n)$. Then the likelihood function is defined

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{y}) = f(\mathbf{y}|\boldsymbol{\theta}) \tag{A.1}$$

and if $\mathbf{y}$ is an i.i.d. random variable, the likelihood function can be defined as

$$\mathcal{L}(\boldsymbol{\theta}|\mathbf{y}) = \prod_{k=1}^{n} f(y_k|\boldsymbol{\theta}) \tag{A.2}$$

where a logarithm is usually taken to produce the log-likelihood function

$$\mathcal{LL}(\boldsymbol{\theta}|\mathbf{y}) = \sum_{k=1}^{n} \log(f(y_k;\boldsymbol{\theta})) \tag{A.3}$$

and the maximum likelihood estimate (MLE) is defined as

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}}(\mathcal{LL}(\boldsymbol{\theta};\mathbf{y})) \tag{A.4}$$

In the case where the distribution is a $\mathcal{N}(\mu, \sigma^2)$ normal distribution, the pdf is

$$f_k(y_k|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{1}{2}\left(\frac{y_k-\mu}{\sigma}\right)^2} \tag{A.5}$$

where y_k are the $k = 1...n$ data values, μ, the mean of the distribution, and σ^2, the variance of the distribution, are the parameters of the model. Then the likelihood function is given by

$$\mathcal{L}(\boldsymbol{\mu}, \boldsymbol{\sigma^2}|\mathbf{y}) = \prod_{k=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{y_k-\mu}{\sigma}\right)^2} \tag{A.6}$$

and the log-likelihood becomes

$$-\mathcal{LL}(\boldsymbol{\mu}, \boldsymbol{\sigma^2}|\mathbf{y}) = \frac{n}{2}\log(2\pi\sigma^2) + \frac{1}{2\sigma^2}\sum_{k=1}^{n}(y_k - \mu)^2 \tag{A.7}$$

where the negative sign has been moved to the other side of the equation.

The picture is not so straightforward for time series, because the assumption of an i.i.d. random variable is not usually true since the next data value is related to the previous value in a typical time series.

Maximum likelihood estimation for a time series can be done by redefining the problem as a *conditional* MLE problem. To begin to understand the process, consider a simple $AR(2)$ autoregressive time series model with a single lag (Hyndman & Athanasopoulos, 2018):

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \varepsilon_t \tag{A.8}$$

where c, ϕ_1, and ϕ_2 are real numbers and $\varepsilon_t \sim \mathcal{N}(0, \sigma^2)$ i.i.d. is the white noise. Then for zero-centered time series data $\mathbf{y} = (y_1, ..., y_n)$ and parameter vector $\boldsymbol{\theta} = (\phi_1, \phi_2, \sigma)$ we can define the conditional PDF $f(y_3, ..., y_n|y_1, y_2, \boldsymbol{\theta})$.

Define $\hat{y}_t$ to be the forecast value from the model as in

$$\hat{y}_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} \tag{A.9}$$

Now let $\varepsilon_t = y_t - \hat{y}_t = y_t - \phi_1 y_{t-1} - \phi_2 y_{t-2}$ represent the noise found in the data for $t = 3, ..., n$. The white noise values are assumed to be independently drawn from a $\mathcal{N}(0, \sigma^2)$ normal random variable, and therefore give the following form to the PDF:

$$
\begin{aligned}
f(y_3, ..., y_n | y_1, y_2, \boldsymbol{\theta}) &= \prod_{t=3}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{\varepsilon^2}{2\sigma^2}} \\
&= \prod_{t=3}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_t - \phi_1 y_{t-1} - \phi_2 y_{t-2})^2}{2\sigma^2}}
\end{aligned}
\tag{A.10}
$$

and the negative log-likelihood function is defined as:

$$
-\mathcal{LL}(\boldsymbol{\theta}|\mathbf{y}) = \frac{n-2}{2} \ln(2\pi\sigma^2) + \frac{1}{2\sigma^2} \sum_{t=3}^{n} (y_t - \phi_1 y_{t-1} - \phi_2\, y_{t-2})^2 \tag{A.11}
$$

This process can be extended to general AR(p) models but is a complex process. A similar process can also be carried out for moving average models which allows for an MLE method for combined ARMA models.

As the number of parameters in the time series models rises, the complexity of the MLE process also rises and is also susceptible to overfitting. To deal with the problem of overfitting, various *information criteria* can be used to choose the right model, rather than just choosing the model with the optimized log-likelihood value. The Akaike information criterion (AIC) is defined

$$
AIC = 2k - 2\mathcal{LL}(\boldsymbol{\theta}|\mathbf{y}) \tag{A.12}
$$

where k is the number of model parameters is the optimized likelihood value. Among candidate models, the one with the lowest AIC will be the preferred one. The first term of the AIC calculation increases directly with the number of parameters, and so the method penalizes larger numbers of parameters. Overfitting commonly occurs when extra parameters are included for the sole purpose of fitting the given data, regardless of whether there is a theoretical reason to include the extra parameters.

Another information criterion is the Bayesian information criterion (BIC) defined

$$
BIC = k \log(n) - 2\mathcal{LL}(\boldsymbol{\theta}|\mathbf{y}) \tag{A.13}
$$

where n is the number of data points, and the other values have the same meaning as the AIC. Like the AIC, there is a penalty assessed for increasing the number of parameters, discouraging overfitting. Note that BIC also depends on the number of data points n, which allows for models to be compared using different sizes of data sets (otherwise the value of n would be constant across all

BIC values and have no power to differentiate models). This implies that the model with the best performance on the smallest (reasonable) amount of data is likely the preferred model.

A.1.2 Least Squares Estimation

The method of Least Squares Estimation (LSE) has existed since at least 1805 when Legendre published a paper on determining the orbits of comets, however, Gauss later claimed to have known of the method since around 1795 (Stigler, 1981). Since this project deals with time series data, the method will be presented in terms of an AR(2) autoregressive model as in the maximum likelihood estimation above.

Consider once again the definitions of Equations A.8 and A.9 for zero-centered time series data $\mathbf{y} = (y_3, ..., y_n)^\mathsf{T}$ and parameter vector $\boldsymbol{\theta} = (\phi_1, \phi_2)^\mathsf{T}$. This time series can be written as

$$y_t = \phi_1 y_{t-1} + \phi_2 y_{t-2} + \varepsilon_t \tag{A.14}$$

for $t = 3, ..., n$. The LSE process is to minimize the sum of squared residuals, which yields the *conditional sum of squares* function:

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \arg\min_{\boldsymbol{\theta}} \sum_{t=3}^{n} \varepsilon_t^2 \\ &= \arg\min_{\boldsymbol{\theta}} \sum_{t=3}^{n} (y_t - \phi_1 y_{t-1} - \phi_2 y_{t-2})^2 \end{aligned} \tag{A.15}$$

This function is conditional on the initial values of the time series. Equation A.14 can be written in matrix form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta} + \boldsymbol{\varepsilon} \tag{A.16}$$

where

$$\mathbf{y} = \begin{bmatrix} y_3 \\ y_4 \\ \vdots \\ y_n \end{bmatrix}, \mathbf{X} = \begin{bmatrix} y_2 & y_1 \\ y_3 & y_2 \\ \vdots & \\ y_{n-1} & y_{n-2} \end{bmatrix}, \boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_3 \\ \varepsilon_4 \\ \vdots \\ \varepsilon_n \end{bmatrix}$$

Using matrix notation Equation A.15 can be rewritten as

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 \tag{A.17}$$

where $\|\cdot\|$ is the ℓ_2-norm. Set $S = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2$. Then expanding S results in

$$\begin{aligned} S = \|\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\|^2 &= (\mathbf{y} - \mathbf{X}\boldsymbol{\theta})^\mathsf{T}(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}) \\ &= \mathbf{y}^\mathsf{T}\mathbf{y} - \mathbf{y}^\mathsf{T}\mathbf{X}\boldsymbol{\theta} - \boldsymbol{\theta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{y} + \boldsymbol{\theta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\theta} \\ &= \mathbf{y}^\mathsf{T}\mathbf{y} - 2\boldsymbol{\theta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{y} + \boldsymbol{\theta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\theta} \end{aligned} \tag{A.18}$$

using the fact that $\mathbf{y}^\mathsf{T}\mathbf{X}\boldsymbol{\theta} = \boldsymbol{\theta}^\mathsf{T}\mathbf{X}^\mathsf{T}\mathbf{y}$.

Differentiating this matrix equation with respect to $\boldsymbol{\theta}$ using matrix differentiation rules (Petersen, Pedersen, et al., 2008) gives

$$\frac{\partial S}{\partial \boldsymbol{\theta}} = -2\mathbf{X}^\mathsf{T}\mathbf{y} + 2\mathbf{X}^\mathsf{T}\mathbf{X}\boldsymbol{\theta} \tag{A.19}$$

and setting this equal to zero gives the normal equation

$$\mathbf{X}^\mathsf{T}\mathbf{X}\hat{\boldsymbol{\theta}} = \mathbf{X}^\mathsf{T}\mathbf{y} \tag{A.20}$$

where $\hat{\boldsymbol{\theta}}$ is substituted as the theoretical minimum of the function. In practice, however, the system in Equation A.16 would typically be solved numerically using a matrix factorization technique such as QR, Cholesky, or SVD.

A.2 Extending SCALATION for Traffic Simulation

A microscopic traffic simulation system requires, at minimum, several essential elements:

1. An arrival model to introduce vehicles to the system
2. A car-following model to control how vehicles move within the system
3. A network of roads on which the vehicles move

A.2.1 System Code

The system is run by the Simulation class which extends SCALATION's Model class. This class sets up the traffic network by creating instances of the classes described in Tables A.1 and A.2. Each Source must be given a random variable for creating arrivals based on the arrival modeling defined in Sections 4.3 and A.2.2. Once the traffic network has been created the simulation is started and

Table A.1: Location-oriented System Classes

Class	Description
Source	Vehicles are generated in the simulation at Sources
Sensor	Sensors keep track of vehicle data such as counts and speeds and correspond in location to the placement of real-world traffic sensors
Sink	Vehicles exit the simulation at Sinks

Table A.2: Movement-oriented System Classes

Class	Description
Lane	A Lane is a single line of traffic flow. This is where the vehicles actually move
Section	Sections are made up of multiple Lanes usually connecting Sources to Sensors, Sensors to Sensors, or Sensors to Sinks
Highway	A Highway object is a multi-lane, multi-Section stretch of a limited access highway that will usually begin with a Source and end with a Sink
Vehicle	Vehicles are the entities that move around the traffic network.

Vehicles are introduced at Sources. Vehicles extend ScalaTion's SimActor class, and so must implement the act method of SimActor.

ScalaTion has an animation tool that was altered to include zooming and shifting capabilities. Figures A.1, A.2, and A.3 show the traffic system animation at three different zoom levels.

In order to place sensor locations and the roads in the right places within the simulation, the latitude and longitude of each sensor were used to create the simulation structure. The simulation uses speeds measured in meters per second so distances must be measured in meters. In this particular traffic network, the direction of travel is northbound, and since the animation system places the origin in the top left corner of the animation canvas, the first point placed, at animation coordinate $(10.0, 10.0)$ was the simulation sink. Then, given the latitudes and longitudes of each sensor, the animation grid point of the sink, and the formula for calculating the real-world distance between GPS coordinates, each other sensor, working back to the source, could be located in the system, where the system distances would mirror the real-world distances in meters. The code for these calculations is given below in Table **??**. The formulas

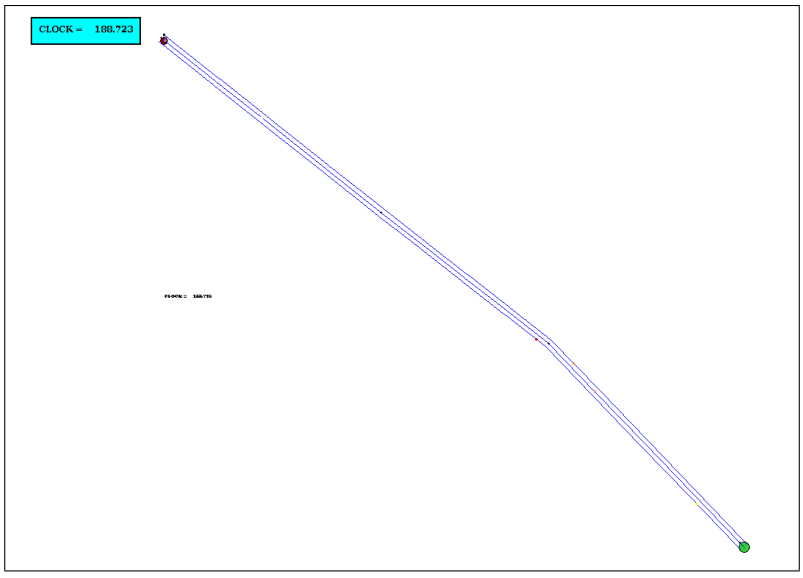


Figure A.1: Screen Grab of Traffic System Animation

for calculating the real-world distance between GPS coordinates are contained in the LatitudeLongitude class in ScalaTion.

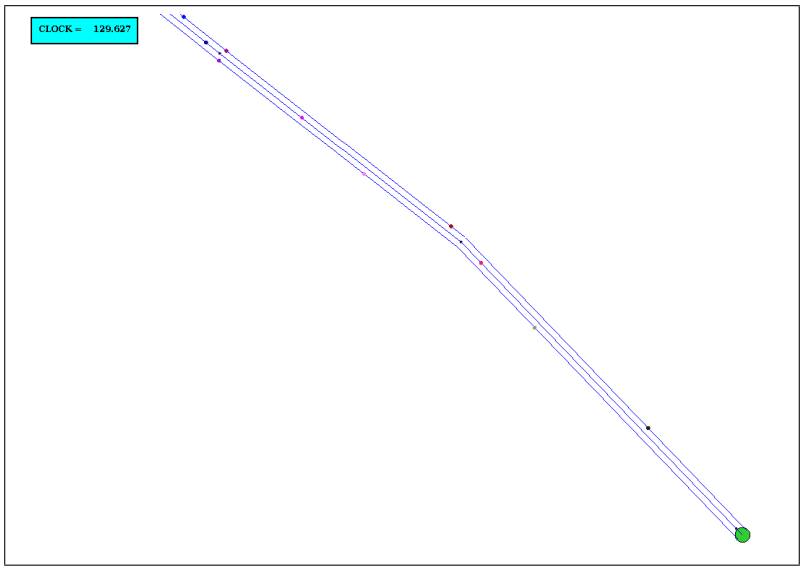


Figure A.2: Screen Grab of Traffic System Animation

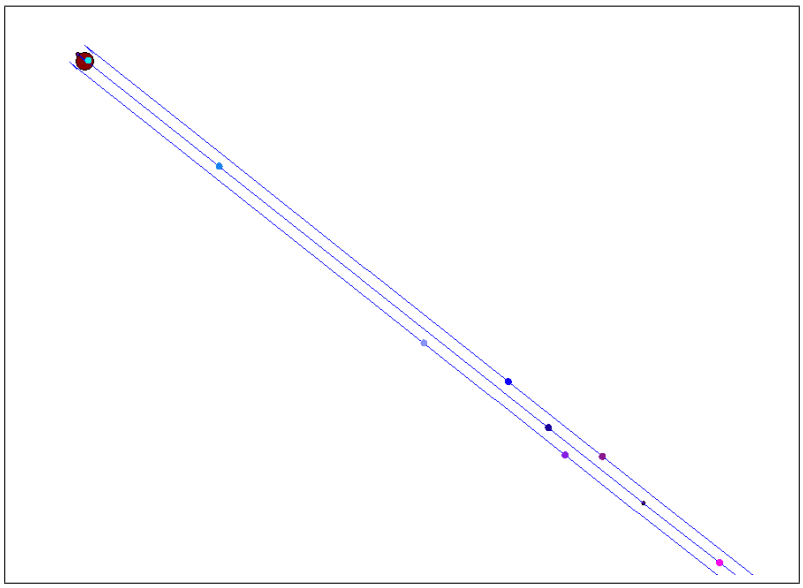


Figure A.3: Screen Grab of Traffic System Animation

Table A.3: Source Code for Determining System Coordinates

```
1  //:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2  /** Code to convert latitude and longitude coordinates to
3   *  animation coordinates. These values reflect distances
4   *  in meters in the real world.
5   */
6  object LatLong
7  {
8      type pt = Tuple2 [Double, Double]
9
10     //:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
11     /** Calculate the new grid point from previous grid point
12      *  and lat-long coordinates
13      *  @param ll1  lat-long coordinates for first point
14      *  @param ll2  lat-long coordinates for next point
15      *  @param xy1  animation grid point for first point
16      */
17     def calcGridPoint (ll1: pt, ll2: pt, xy1: pt): pt =
18     {
19         val lalo1 = new LatitudeLongitude (ll1._1, ll1._2)
20         val lalo2 = new LatitudeLongitude (ll2._1, ll2._2)
21         val rxy   = lalo1.distance (lalo2)
22         val rll   = llDistance (ll1, ll2)
23         val dil   = rxy / rll   // scale factor between lat-long
24         val dlo   = ll2._2 - ll1._2             // and animation
25         val dla   = ll2._1 - ll1._1
26         val nx    = xy1._1 + dlo * dil   // x-coord for new pt
27         val ny    = xy1._2 - dla * dil   // y-coord for new pt
28         (nx, ny)
29     } // calcGridPoint
30
31     //:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
32     /** Euclidean distance between lat-long coordinates.
33      *  @param p1  lat-long coordinates for first point
34      *  @param p2  lat-long coordinates for next point
35      */
36     def llDistance (p1: pt, p2: pt): Double =
37     {
38         val dx = p2._2 - p1._2
39         val dy = p2._1 - p1._1
40         Math.sqrt (dx * dx + dy * dy)
41     } // llDistance
42 } // LatLong
```

A.2.2 Arrival Model Code

ScalaTion handled simulation arrivals by using random variables to produce interarrival times. In principle, this is quite flexible, but there were no built-in random variable types that could produce realistic traffic arrivals based on real traffic data. The first step required to provide realistic arrivals was to find an appropriate mathematical model for system arrivals with multiple busy periods. The model that showed real promise for ease of use, flexibility, and accuracy was the Nonhomogeneous Poisson Process (NHPP), and specifically the procedure given by Leemis (L. M. Leemis, 1991), and in (L. M. Leemis, 2004). The original paper included pseudocode for generating event times based on the procedure, which was translated into scala code. This code is presented in Table A.4:

Table A.4: Source Code for Generating NHPP Event Times

```
1  //:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2  /** Compute arrival times of the NHPP.
3   */
4  def genTime: Double =
5  {
6       tlast = t
7       e += e_rv.gen
8       for (i <- 0 until lsum.dim if e <= lsum(i)) {
9            val lsum_i_1 = if (i == 0) 0.0 else lsum(i-1)
10           val d = e - lsum_i_1
11           t = dt * (i + d / (lsum(i) - lsum_i_1))
12           return t
13      } //
14      flaw ("gen", "cumulative e value larger than last lsum")
15      -1.0
16 } // gen
```

Note that the for loop on line 8 contains an if-statement, and actually only executes the body of the loop when the right interval in the cumulative rate function has been found. Once the NHPP was coded it could be used with the Source class provided in ScalaTion. This class produces new entities by using a given random variable to generate a new interarrival time and then putting that on the event scheduler.

A.2.3 Vehicle Code

When vehicles are created they are ultimately controlled by their act methods. The act method directs them by telling them to move down particular roads and sets up the vehicle's parameters such as velocity, acceleration, etc. The act method calls a method to choose a lane as well. The highway modeled in this project contains three lanes, and when vehicles are created in the system, they are placed in the lane with the fewest total number of other vehicles. If they all have the same number of vehicles then the algorithm defaults to a randomly chosen index. Internally, though the roads have a ScalaTion object that controls the motion of the vehicles on that segment, there is also a multidimensional ArrayBuffer that contains references to the vehicles. Each vehicle maintains a reference to the vehicle in front of it and the vehicle behind it. These references are called "pred" (for *predecessor*), and "succ" (for *successor*). This creates a basic linked list structure, and when vehicles are placed on the road the links have to be updated. This will eventually facilitate the inclusion of lane changing and merging.

Once the vehicles have been assigned a lane, their velocity is set. The simulation source is merely the simulation horizon on the freeway, but the vehicles are assumed to already be driving at some speed. All vehicles desire to drive *vmax* meters per second, which is a maximum velocity that can be set to a specific value, or randomly assigned, however, it is not always possible for the vehicle to be driving at this speed at the beginning. Once the vehicle was assigned a lane, and its predecessor determined, it is given the minimum of its own *vmax* value and the current velocity of its predecessor. However, it is possible that at times of sparse traffic, there is not a predecessor, and in this case, the vehicle is assumed to already be driving at its *vmax* value.

Now that the vehicle has been assigned a velocity, it is moved down the corridor, which is the ScalaTion object representing the traffic network from source to sink. The corridor handles the movement of the vehicle and updates its position using the car-following model. Once the corridor has moved the vehicle to the sink, the vehicle leaves the simulation at its destination. Vehicles have a destination variable to eventually facilitate multiple exits (multiple sinks). Once the vehicle exits the simulation the links to other vehicles must be reset.

Table A.5: Car case class

```
1  //::::::::::::::::::::::::::::::::::::::::::::::::::
2  /** Car case class that represents vehicles in the
3   *  network. The arguments for the Vehicle constructor
4   *  come from above.
5   */
6  case class Car() extends Vehicle("c_" + count, this, VectorD(τ, amax,
7          bmax, 0.0, vmax, T, s, len, δ))
8  {
9      //::::::::::::::::::::::::::::::::::::::::::::::::::
10     /** act method for cars generated at the first
11      *  source (these cars are already on the highway)
12      */
13     def act(): Unit =
14     {
15         val j = chooseLane(laneChoice.gen.toInt) //lane number
16         lane = j
17         this +=: cars(j)
18         if (cars(j).length > 1) {
19             pred = cars(j)(1)
20             cars(j)(1).succ = this
21             velocity = min(vmax, 0.9 * pred.velocity)
22         } else {
23             velocity = vmax
24         } // if
25         cor.move(j)
26         cars(j) -= this
27         dest.leave()
28         if (succ != null && pred != null) {
29             succ.pred = pred
30             pred.succ = succ
31         } else if (succ != null) {
32             succ.pred = null
33         } else {
34             pred.succ = null
35         } // if
36     } // act
37
38     //::::::::::::::::::::::::::::::::::::::::::::::::::
39     /** Method to choose which lane the car should be in
40      *  @param ii the lane chosen randommly
41      */
42     def chooseLane(ii: Int): Int =
43
44         var j = 0
45         var n = cor.cars(ii).length
46         for (i <- cor.cars.indices) {
47             val nn = cor.cars(i).length
48             if (nn < n) {
49                 n = nn
50                 j = i
51             } // if
52         } // for
53         j
54     } // chooseLane
55
56 } // Car
```

A.2.4 Car-Following Code

Originally SCALATION did not include car-following models in any way. Movement within a simulation was controlled by setting the total time it should take an object to move down a pathway and evenly dividing this along the pathway itself. For vehicles on a highway, this type of movement would not work because vehicles must be able to speed up and slow down depending on what is happening around them. In order to achieve more realistic movement of the vehicles, the implementation of car-following models was required. The Motion class serves as a general-purpose car-following model class, where in fact many car-following models could be implemented.

Table A.6 shows two functions implementing the car-following mode of the IDM, and the free-driving mode, respectively.

Table A.6: Source Code for IDM Implementation

```
1  //::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2  /** Return the acceleration of the vehicle based on the Intelligent
3   *  Driver model.
4   *  @param an  max acceleration
5   *  @param bn  max deceleration
6   *  @param sp  length of previous car
7   *  @param Vo  desired velocity
8   *  @param xn  position of current car
9   *  @param vn  velocity of current car
10  *  @param xp  position of previous car
11  *  @param vp  velocity of previous car
12  *  @param T   time headway
13  *  @param so  distance headway
14  *  @param δ IDM exponent
15  */
16 def IDM (an: Double, bn: Double, sp: Double, Vo: Double, xn: Double,
17         vn: Double, xp: Double, vp: Double, T: Double, so: Double,
18         δ: Double): Double =
19 {
20     val Δx = xp - xn - sp
21     val Δv = vn - vp
22     val ss = so + vn * T + (vn * Δv) / (2.0 * Math.sqrt (an * bn))
23     val a  = an * (1.0 - (vn / Vo) ~^ δ - (ss / Δx) ~^ 2.0)
24     a
25 } // IDM
26
27 //::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
28 /** Return the acceleration of the vehicle based on the IDM
29  *  when there is no previous car.
30  *  @param an  max acceleration
31  *  @param Vo  desired velocity
32  *  @param vn  velocity of current car
33  *  @param δ IDM exponent
34  */
35 def IDMFree (an: Double, vn: Double, Von: Double,
36              δ: Double = 4.0): Double =
37 {
38     an * (1.0 - (vn / Vo) ~^ δ)
39 } // IDMFree
```

The IDM was only one of the car-following models used in the work. The other was Gipps' Model and the code for implementing it is presented in Table A.7.

Table A.7: Source Code for Gipps' Model Implementation

```
1   //-------------------------------------------------------------
2   /** Return the velocity of the vehicle based on Gipps' model.
3    *  @param an  the max acceleration of driver n
4    *  @param bn  the max deceleration of driver n (negative #)
5    *  @param sp  the size of the predecessor's vehicle
6    *  @param Vn  the desired velocity of driver n
7    *  @param xn  the current position of driver n
8    *  @param vn  the current velocity of driver n
9    *  @param xp  the current position of the predecessor
10   *  @param vp  the current velocity of the predecessor
11   *  @param τ  the reaction time of driver n
12   */
13  def gipps (an: Double, bn: Double, sp: Double, Vn: Double,
14             xn: Double, vn: Double, xp: Double, vp: Double,
15             τ: Double): Double =
16  {
17      val free = vn + 2.5 * an * τ * (1.0 - vn / Vn) *
18      Math.sqrt (0.025 + vn / Vn)
19      val cong = bn * τ + Math.sqrt (bn * bn * τ * τ -
20      bn * (2 * (xp - sp - xn) - vn * τ - vp * vp / bn))
21      Math.min (free, cong)
22  } // gipps
```

The car-following model itself is only a part of the system that controls vehicle movement within the simulation. Also required are details about joining roads and exiting roads. The code needed for constructing the traffic network is located in the following classes:

Within the Lane code, the move method is where the action takes place, and that code can be viewed in Table A.8

The if-statement on line 18 handles the situation when a vehicle has reached its destination exit while still in the current lane. In this case, its predecessor and successor vehicles must be linked up to keep the order of vehicles intact and the car-following model running correctly. There are actually several cases that must be processed, depending on whether the vehicle exiting the simulation actually has a predecessor or successor at all. The vehicle will remain in the while loop inside the move method until it has either exited the system, in which case its "done" parameter is now true, or it has reached the end of this lane (reached the next Sensor) and will be moving on to the next Section of the Highway.

Table A.8: Move Method in Lane.scala

```
1  //:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2  /** Move the vehicle down the lane using the vehicle's own
3   *  update method, which utilizes a car-following model
4   *  to determine the vehicle's next coordinates.
5   *  @param x  The distance along the lane at which the
6   *            the vehicle begins if coming from a ramp.
7   */
8  def move (x: Double = 0.0)
9  {
10     val actor = director.theActor.asInstanceOf [Vehicle]
11     actor.disp = x
12     tally (actor.τ)
13     while (actor.disp < ratio * length && !actor.done) {
14         trace (this, "moves for " + actor.τ, actor, director.clock)
15         actor.update()
16         director.animate (actor, MoveToken, null, null,
17         calcPoint (actor.disp / ratio))
18         if (actor.dest.x <= actor.t_disp) {
19             actor.done = true
20             var p: Vehicle = null
21             var s: Vehicle = null
22             if (actor.pred != null) p = actor.pred
23             if (actor.succ != null) s = actor.succ
24             if (p != null && s != null) {
25                 p.succ = s
26                 s.pred = p
27             } else if (p != null && s == null) {
28                 p.succ = null
29             } else if (p == null && s != null) {
30                 s.pred = null
31             }
32             actor.pred = null
33             actor.succ = null
34         }
35         actor.schedule (actor.τ)
36         actor.yieldToDirector ()
37     } // while
38 } // move
```

A.3 Genetic Algorithm Code

To facilitate the calibration of the model using a GA, the decision was made to write custom code for this purpose. The code was still written to be fairly generic, but knowledge of this simulation system helped tailor the process to work well with these models. The GA is a standard approach to the idea where a pool of candidate solutions is created at the start of the procedure, here using random variables as described earlier in the paper, and then proceeding using the evolutionary concepts of crossover and mutation to generate new candidates as the algorithm progresses.

Table A.9: Solve Method for Genetic Algorithm

```
1  //::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2  /** Solve the optimization problem.
3   * @param seeds  a (possibly null) array of initial
4   *               candidates provided by the user.
5   */
6  def solve (seeds: Array[VectorD] = null): Tuple2 [Double, VectorD] =
7  {
8      initPool (seeds)
9      sortPool ()
10     println ("----------------------------------------")
11     println ("Generation 0:")
12     printPool ()
13     breakable { for (i <- 0 until max_iter) {
14         println ("----------------------------------------")
15         println ("Generation " + (i + 1) + ":")
16         nextGen ()
17         sortPool ()
18         printPool ()
19         epochs += pool(0)._1
20     }}
21     pool(0)
22 } // solve
```

The solve method is the director of the optimization but crossover and mutation happen in their own methods, which are actually called by the nextGen method.

The crossover method takes the two vectors being crossed and randomly generates an index (using value k at which to divide the parameter space. The random integer j determines which of the two vectors is used for the first part of the child and which is used for the second part of the child. The mutate method applies a multiplicative change to the values in the vector being mutated. Currently, these mutations are kept relatively small, with the random variable randMut being a uniform random variable on the interval $[-0.2, 0.2]$. This means that any mutation is at most a 20% change from the previous value, ensuring that mutations are not too large.

Table A.10: Crossover and Mutation Methods for GA

```
1  //::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
2  /** Calculate the crossover of two solutions.
3   *  @param x1  the first solution for the crossover
4   *  @param x2  the second solution for the crossover
5   */
6  def cross (x1: VectorD, x2: VectorD): VectorD =
7  {
8      val k = randInd.igen            // generate a random index
9      val j = randInd.igen
10     var x = x1
11     if (j % 2 == 0) x = x1(0 until k) ++ x2(k until x2.dim)
12     else            x = x2(0 until k) ++ x1(k until x1.dim)
13     x
14 } // cross
15
16 //::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
17 /** Perform a mutation on a solution.
18  *  @param x  the solution on which to perform the mutation
19  */
20 def mutate (x: VectorD)
21 {
22     for (i <- 0 until x.dim) {
23         x(i) = x(i) * (1.0 + randMut.gen)  // apply a multiplicative
24                                            // factor to the current
25                                            // index-value of the
26                                            // solution.
27     }
28 } // mutate
```