MORE EFFICIENT LEARNING IN TRAFFIC GRIDS VIEWED AS COMPLEX ADAPTIVE SYSTEMS USING AGENT BASED MODELING

by

Marion Charles Lane

A dissertation submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in
Computing and Information Systems

Charlotte

2017

Approved by:

_____
Dr. Mirsad Hadzikadic

_____
Dr. Zbigniew Ras

_____
Dr. Weichao Wang

_____
Dr. Edd Hauser

ABSTRACT

MARION CHARLES LANE. More efficient learning in traffic grids viewed as complex adaptive systems using agent based modeling. (Under the direction of DR. MIRSAD HADZIKADIC)

Equation based modeling (EBM) is the most common form of scientific modeling. However, the creation of an appropriate EBM for a large system is likely to be complicated and computationally expensive. In contrast, consider the possibilities if even a large system is treated as a complex adaptive system (CAS), applying the basic tenets of a CAS to a model using the concepts of agent based modeling (ABM). ABM requires only the definition of the model environment, the identification of key agents, and a minimum number of key behaviors of those agents. It was a contention of this study that a CAS/ABM model would be easier to design, implement, execute, and extend than an EBM model. It was also a contention that the results would still be predictive and useful. The ABM herein was based on the Uptown Charlotte, North Carolina, traffic grid using traffic volumes based on actual Charlotte traffic counts. It operated with autonomous traffic signals as an adaptive CAS.

DEDICATION

To ELD, a daughter of Algiers, my best friend, and my constant memory.

ACKNOWLEDGEMENTS

I would like to acknowledge and give thanks to my advisor and dissertation committee chairman, Dr. Mirsad Hadzikadic, a truly patient man.

I would also like to acknowledge the remaining members of my committee, Dr. Zbigniew Ras, Dr. Weichao Wang, and Dr. Edd Hauser, for their insights and flexibility.

Finally, when I decided to model the environment for this project specifically to the Uptown Charlotte traffic grid, I realized that I needed to use actual Charlotte data for input. My initial assumption was that these data were available and easily accessible, but this was not the case. As often is the case, the key to success is simply finding the right people to ask and finding them took a circuitous route. Many thanks to Mr. Stephen P. Piotrowski, Ms. Jan Murphy, and Mr. Anthony Tagliaferri of the North Carolina Department of Transportation and Mr. Felix Obregon of the Charlotte Department of Transportation for their help.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

The purpose of this dissertation is to report the results of simulations run through a computer model based on the street grid of the Uptown area of Charlotte, North Carolina. This grid contained 29 roadways, one light rail line, and approximately 140 intersections controlled by a collection of traffic lights and stop signs. The traffic volumes which passed through the simulations were based on actual traffic measured on the streets of Uptown Charlotte.

At the outset the core contention was that this grid, implemented using the principles of Agent Based Modeling (ABM), would operate effectively and efficiently as a Complex Adaptive System (CAS) with autonomous agents (traffic lights), without central control. Furthermore, the lights would be able to learn from previous success, improving over time.

This contention was tested with a three-step process. Initially a performance baseline was established using a simple, fixed-phase algorithmic control system. Second, a basic, autonomous agent solution was implemented to demonstrate that the grid would in fact act as a CAS. Third, the basic autonomous solution was augmented adding functionality to allow the traffic light agents to learn from their past success and improve the operation of the grid. In general the basic autonomous solution did prove to be successful and superior to the fixed-phase system. Subsequently, learning improvements were small but consistent.

Previous research in the traffic control space had been done primarily with rule-based and equation based modeling (EBM) disciplines. There had been little research using the CAS/ABM approach. The initial expectation was to be able to show that a CAS/ABM would be easier to design, implement, execute, and extend and that the results would still be predictive and useful. As will be discussed later, the truth of this statement is dependent on project design boundaries and the availability of developmental design tools.

CHAPTER 2: BACKGROUND RESEARCH

There have been a number of research projects modeling the management of traffic in a grid. Rochner (Rochner, 2006) provided a general description of the approach that has normally been taken, proposing a three-tier structure. Layer 0 consisted of preprogrammed local traffic lights at individual intersections. Layer 1 consisted of controllers, still at the local level, but responsible for coordinating lights in the same neighborhood. Level 2 consisted of an off-line generator which could be invoked when current traffic presented a unique situation that had not already been preprogrammed into Level 0 and Level 1. The system learned from the new traffic condition, computed a solution for that condition, and loaded the new solution back into Level 1 to be downloaded to Level 0 when the new situation recurred. However, this report stopped at the proposal. It did not provide details of how such a structure would actually be implemented; nor, significantly, did it provide simulation results or any suggestions for scalability.

The majority the actual research that has been done was concentrated in the areas of cellular automation, self-organization, and machine learning, specifically reinforcement learning. The objectives of these studies were the same, more efficient traffic flow; but the methodologies differed significantly. Reinforcement learning was the most common approach. It was characterized by rule and/or equation systems that attempted to improve the efficiency of the next execution of the model by rewarding

system parameter settings that had shown improvement in the current execution. The normal sequence of operations (similar to Rochner) was initialization, execution of the simulator for an appropriate interval, recalibration of the model using the most successful current system values (learning), execution of the simulator for another interval, and repetition of the recalibration and execution steps, as needed.

Wiering, Steingrover, and Xu were worthy examples of reinforced learning. Wiering (Wiering, 2000) described an environment where both traffic lights and cars were agents. The grid was visualized as a network where intersections were nodes and streets were edges. There were six intersections. The lights were connected and could coordinate with each other. The cars were assigned destinations when they entered the grid and made routing decisions based on traffic conditions. The lights and cars were influenced by reinforcement learning algorithms that attempted to adjust light phases and car routes to allow each car to reach its destination as quickly as possible. The authors left it to future research to address "more realistic traffic simulators" which should be interpreted to include larger, more complex traffic grids and more extensive computational loads.

The Steingrover (Steingrover, 2005) environment was a sixteen node, right-angled grid implemented using the Green Light District traffic simulator. Cars were randomly assigned a destination node when they were "spawned" into the grid and then planned the most efficient path to their destinations. Reinforcement learning based on the traffic at neighboring lights and the degree of congestion at those lights was used to improve traffic flow. The authors stated concern for possible "computational problems" for networks larger than the tested sixteen-node grid.

Xu (Xu, 2014) also described a fixed, right-angled grid. It had twenty nodes, and was implemented with a multi-layer architecture on the RETSINA system. Each intersection maintained data on the state of the lights at the intersection and its neighbors, the state of individual lanes at the intersection, and a local coordinator. The local controllers managed local neighbors and coordinated to manage the grid as a whole using reinforcement learning. Once again, the authors expressed concern that unpredicted traffic situations for larger grids would be "computationally hard."

It was the assumption of this study that implementing rules and/or equation systems to adequately simulate any large target system using reinforced learning would be a significant task. Additionally, recalibration of the rule/equation systems after each execution interval would become increasingly more computationally expensive when required to scale up for larger systems. The parallel assumption of this study was that implementation and recalibration factors for a CAS/ABM implementations would be less costly than those for comparable reinforced learning systems. The anticipated reduction was one justification for this research.

Brockfeld (Brockfeld, 2001) was an example of research based on a cellular automaton model. It defined a simple square grid with intersections controlled by traffic lights overlaid onto a lattice of cells representing the incremental movement relationship boundaries among the agents. (This is the same type of layout lattice as the NetLogo environment on which the Uptown Charlotte grid was deployed.) The results were based on the cumulative progress of all cars. The length of the light phases was set, and the red and green phases for all north/south lights alternated with the phases of the east/west lights. Brockfeld found that the efficiency of traffic flow was dependent on the

relationship between the length of the go phase of a light and the length of the street segment being controlled by the light. These results were confirmed by my own research (Chapter 3: Previous Research) and were part of the basis for this project.

Gershenson presented a traffic control simulation model also developed on NetLogo that viewed traffic light control "not so much an optimization problem, but rather an adaptation problem, since traffic flows and densities change constantly." (Gershenson, 2005). It was based on the right-angled grid of the Gridlock model provided with NetLogo which includes single lanes of traffic in four directions. Gershenson's work included three traffic control methods that self-organized based on how individual agents measured waiting traffic. The first, "request," measured the number of cars waiting behind red lights. When that count exceeded a predetermined threshold, the light requested a go phase. The second, "phase," introduced a minimum go phase length to the request method. In order to get a new go phase the traffic waiting count must have exceeded the threshold and the current go phase must have exceeded the minimum go phase length. The third, "platoon," extended "phase" by adding additional checks to prevent traffic waves that were considered too long. Cools (Cools, et al., 2008) was an extension of Gershenson's work and aimed to further promote traffic waves. It was based on the "platoon" method of Gershenson, but removed the minimum go phase requirement and added additional conditions with the goal of keeping traffic waves together.

The works of Brockfeld, Gershenson, and Cools were key predecessors of this research. However, Gershenson and Cools focused specifically on producing traffic waves on relatively simple traffic grids using traffic data generated specifically for their

studies. The Charlotte grid was more complex, and traffic data were based on measurements of from the actual grid. Based on my previous research, the expectation was that waves (perhaps less structured) and self-organization would result organically when produced through the simple behaviors of autonomous complex system agents that are focused specifically on efficient learning. Specific details of the models used in this research are described in Chapter 6: The Simulation Environment.

CHAPTER 3: PREVIOUS RESEARCH

My history with CAS began as a class project for a Complex Systems class at the University of North Carolina Charlotte. My project choice was an ABM to manage a single right-angled intersection with two lanes of traffic heading in each of the north, south, east, and west directions plus a single east-to-north turn lane. The north and south lanes were always in the same stop-go phase. The east and west lanes were always in the opposite stop-go phase. The length of a go phase for each heading pair was fixed. The turn lane had a go phase only if there were cars in the lane. The traffic arrival rate was controlled with an external data file created specifically for this model. Unfortunately this initial effort was strictly algorithmic, controlled at the overall system level, not a CAS at all. However, it was an introduction to ABM, the NetLogo modeling tool, and a systematic way of thinking about traffic management.

I have since implemented four additional NetLogo ABM models with more ambitious goals for each successive effort. The second model was a modification of the class project, replacing the fixed-phase length control for the intersection with one that changed the phase only if there was traffic waiting in the opposite direction. Minimum phases were still fixed-length. Once a green phase began, it remained green for a minimum fixed time then changed only if there was traffic waiting in the opposite

directions. Although more responsive to traffic conditions, I would still not consider this a CAS. Control was still at the system level.

The third and fourth models expanded the environment. Instead of a single intersection, each dimension of the new grid could range independently from one to six streets; so the number of right angled intersections could vary from one to 36. Each street had two lanes of traffic, one in each direction. The traffic arrival rate could be determined by either of two modes: uniform, where a fixed number of cars was spread uniformly among all entry points, or random, where rates and times were controlled individually by heading by an external file. There were two operational modes (and thus, two new models): fixed and thru. With fixed, all the north/south lights were always in the same phase, and all the east/west lights were always in the opposite phase. Go phases were of fixed length. With thru, each individual light could request a go phase based on whether it had waiting traffic. Fixed mode showed some characteristics of autonomous CAS behavior but was still centrally controlled. Thru mode was my first true CAS model with each light acting simply but autonomously.

The fifth model was the Uptown Charlotte grid. It was an extension of the work of models three and four on a more complex, less symmetric grid.

CHAPTER 4: COMPLEX SYSTEMS

Since the Scientific Revolution (mid-16th century – early 20th century) the predominant belief was that the study of science required the object of study to be subdivided into increasingly more elemental components. Scientific knowledge was defined as gaining greater understanding of the composition and interaction of those components, subdividing again, and repeating the process. This approach, Reductionism, has been successfully used to increase scientific knowledge in a wide range of physical and non-physical sciences. However it has been notably less successful with many others. How does weather work? Why can we not predict it reliably more than a few days in advance? Why does it seem that no two economists can seem to agree on how the economy will respond to new stimuli? Why does the stock market seem to respond irrationally to political and social changes? How do individuals in social insect colonies adapt their roles to fit the current needs of the colony without the presence of a manager to tell them what to do and when to do it? Why do traffic grids seem to work well some of the time but become completely jammed at others? (Mitchell, 2009)

Complex Systems is a fairly new discipline. In the mid-20th century, when it was becoming clear that new knowledge in physics and biology was requiring us to rethink our understanding of the world, many scientists began to doubt that we could reduce our way to scientific understanding of extremely complicated systems such as weather, economics, or insect colonies. This line of thought coalesced with the formation of the

Santa Fe Institute in 1984 to create what was called “the sciences of the twenty-first century” to study systems that were simply too complex to understand every detail. (Waldrop, 1992) (Santa Fe Institute, n.d.) Since its inception the Institute has been a center for the study of the theory of complex systems and to discover real-world applications.

Perhaps because of the newness of the discipline or perhaps because of the many dispirit areas to which scientists have sought to apply it, there does not even seem to be a concise, well-accepted definition of exactly what a complex system is. Melanie Mitchell stated her definition in terms of large networks, no central control, simple interactions among the components of the network, sophisticated information processing, and adaptation; and then wrote an entire book explaining what it means. (Mitchell, 2009) If we are not able to precisely define it, for this research at least we can enumerate the components of a complex system and use that collective description as a suitable definition.

A complex system has discrete agents. The agents interact autonomously with each other with simple behaviors resulting in very complex interactions that define the structure of the system. The operation of the system is non-linear, non-algorithmic. The system is self-organizing and has no overall manager. Over time the agents adapt their behavior and learn. The process to organization results from the autonomous interactions of the agents which over time produces order. (Hadzikadic, 2015) (Mitchell, 2009) This was the working definition of a complex system for this research.

The work of Mark Buchanan also addressed adaptive, self-organizing behavior. He described individual people as “social atoms” interacting with other “atoms” within

the human society. Ordinarily we think of these atoms as being compelled to act in their own best interest, and it seems collectively this selfish behavior generally works for the overall good. One could argue that in whatever ways these atoms are interacting this society has been fairly successful. However, there also seem to be some common behaviors such as charitable activities, wartime self-sacrifice, and risky rescue activities during natural disasters that are decidedly not in an individual's best interest. Buchanan's explanation for our tendency for these unselfish behaviors was that they have evolved from adaptations our ancestors made centuries ago because they proved to be good for the society. He also said that the best way to reveal these tendencies was not to observe individuals, but rather to experimentally observe overall societal patterns. (Buchanan, 2007) Extending Buchanan's analogy for the purpose of this study suggests that autonomous traffic lights may be considered social atoms as well, operating in the structured society of a traffic grid. Consider the lights to be initially operating selfishly but adapting and learning over time for the ultimate good of the entire traffic grid. The lights self-organize to a more efficient system without benefit of an overall manager, confirming the definition of a complex system.

Another approach to a complex system was the concept of a cellular automaton as first proposed by Stanislas Ulam in the 1940s and later extended by John Von Neumann, Arthur W. Burks, and others. Melanie Mitchell defined a cellular automaton as being "composed of large numbers of simple components (i.e., cells) with no central controller, each of which communicates with only a small fraction of the other components. Moreover, cellular automata can exhibit very complex behavior that is difficult or impossible to predict". (Mitchell, 2009, pp. 148-149) This seems to have much in

common with the definition of a complex system which we laid out earlier. In the early 1980's Stephen Wolfram proposed that all update rules for cellular automata (how the cells interact) fall into one of four classes. In class one all cells quickly settle into the same state after only a few cycles. In class two the cells, after briefly seeming to show signs of organization, eventually stagnate and simply oscillate within unconnected groups. The third class is characterized by continuing wild gyrations and failure to ever organize. For our purposes, class four resides in some place between classes two and three. It produces neither stagnation nor wild gyrations. Instead it does produce groups of cells that seem to be continually interacting nicely with each other. In 1986 Mark Langston, reviewing his own implementation of a cellular automaton, categorized classes one and two as being Order and class three as being Chaos. Class four he called Complexity. By fine tuning his update rules he was able to produce groups of cell structures that organized themselves and "grew and split and recombined with eternally surprising complexity". Langston defined this as operating at some point between Order and Chaos or at the "the edge of chaos". I would suggest using Langston's quote as an alternate definition of complex systems: a system capable of operating at the edge of chaos. (Waldrop, 1992)

Although not strictly a definition, there is also a graphic signature, based on the logistic map algorithm, found to be common to complex systems. Mitchell Feigenbaum demonstrated in the 1980's that the logistic map algorithm could be used to describe the behavior of dynamic systems in general (Mitchell, 2009). It was anticipated that if the Charlotte Uptown ABM did operate as a CAS, then it would produce that type of signature, too.

The algorithm is very simple:

$$x_{t+1} = Rx_t (1 - x_t)$$

x_t represents the items being measured over time with values in the range 0 - 1. t represents succeeding time intervals. R is a constant with different values appropriate for specific systems. As an example, iterating the equation for ten cycles with R = 2 and $x_0 = 0.2$ will produces the graph in Figure 1.

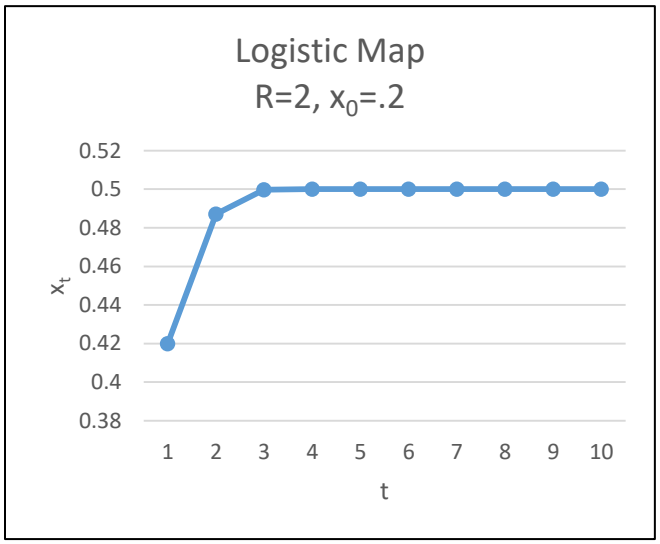

Figure 1: Logistic Map, R = 2, $x_0 = 0.2$

In fact, with a value of R = 2, the graph will always converge to the fixed point, 0.5, regardless of the initial value of x_0. As the value of R increases the curve becomes increasingly irregular. At values near 3.569 and higher, it becomes unstable, chaotic. Based on Feigenbaum's work, the logistic map can be used to describe the entire range of dynamic systems from order to chaos. If we posit that the curve of the logistic map with

R = 2 is the representation of the perfect complex system existing on that dynamic range and showing adaptive, emergent behavior, then I submit we should also be able to say that any system that operates with a similar signature has characteristics that are necessary but not necessarily sufficient for it to be considered a complex system.

Let x represent the number of cars (normalized into the range 0 – 1) on a traffic grid in serial time intervals. If the success of the grid control is defined as the ability of cars to pass efficiently through it and to avoid traffic jams, then the number of cars arriving into the grid must be offset by the number of cars leaving the grid. If we start as cars arrive and begin to fill the empty grid, the curve will have a steep positive slope. If more cars enter and remain in the grid than leave, the slope will remain positive. If traffic control is working well, eventually cars will begin to exit the grid at a rate no less than the arrival rate and the curve will flatten or turn downward. If the arrival rate and exit rates are constant and equal for a period of time, the curve should flatten at some steady state value, the fixed point. If the arrival rate rises or falls (normal ebbs and flows of traffic), the curve should fluctuate up or down temporarily but eventually flatten again. I consider this behavior to be a signature of a complex system. If the curve always has a positive slope or simply fluctuates without ever flattening, this would be an indication that the system is not acting as a complex system. In Chapter 8: Simulation Results as part of the verification of the model I will show that the performance curves of the Charlotte grid ABM did produce this signature.

(The terms complex system and complex adaptive system are often used interchangeably. The term complex adaptive system or CAS seems more appropriate for this study, so from this point forward I will use it when possible.)

CHAPTER 5: AGENT BASED MODELING

5.1 Equation Based Modeling vs. Agent Based Modeling

Uri Wilensky and William Rand declare that equation based modeling (EBM) is the most common form of scientific modeling. (Wilensky, 2015) EBM as described by H. Van Dyke Parunak is based on "individuals" and "observables" where individuals are system actors and observables are the measurable factors that describe system status. In an EBM model, the macro-level operation of the system is represented by equations which define the relationships among the observables. The individuals are recognized as part of the system but are not unique. All individuals of the same type are considered the same. Over time the repeated evaluation of these equations represents changes in system status through changes in the values of the observables. The validity of such models is directly related to how well the equations represent the overall system; and validation can be done by comparing model results to actual system results. (Parunak, 1998)

CAS's are by definition difficult if not impossible to understand on a macro level, so it would be correspondingly difficult to create an EBM equation system to represent them. However, it is possible to identify the key discrete individuals, agents, of complex systems and the key behaviors with which those individuals interact with each another. ABM requires the identification of the environment in which the agents exist, the agents, and behaviors as the basis of a model. Running multiple simulations allows the agents to

interact repeatedly with each other and for new values for the observables to emerge from those interactions rather than being input at the outset. By focusing on the results of the micro actions of individual agents rather the macro system as a whole, ABM is an excellent tool for revealing the properties of a CAS. ABM's can be validated at the system level the same as EBM models; but because they operate at the individual agent level individual agents can be observed, combined, and evaluated as well. (Parunak, 1998)

If we accept that generally CAS and ABM are a good match, should we assume that together they are also a good match the model of Uptown Charlotte? Charles Macal and Michael North suggest that the most import considerations for suitability are readily identifiable agents and behaviors and that any process structure changes be internal to the model. (Macal, 2006) Franziska Klugl and Ana Bazzan suggest that a complex system should have flexible and local interaction, heterogeneous agents and behaviors, learning at both the individual and overall levels, and allow for non-stationary equilibrium. (Klugl, 2012) The model for this project satisfied both Macal and North requirements. The agents were the autonomous traffic signals on the grid. The behaviors were how those traffic signals change based only on the grid environment with no external control. The behaviors may have changed as the lights learned, but the grid itself, once finalized, was constant. These agents, behaviors, and grid specifications also satisfied the first three Klugl and Bazzan requirements. The fourth, non-stationary equilibrium, was addressed by the fact that the model was able to respond to and recover from the ebb and flow of normal traffic arrival rates. In summary, based on these criteria the proposed model was a good fit for CAS and ABM.

5.2 Validation and Verification

Once an ABM is created and used for actual simulations, how can we be certain the results are relevant, accurate, and useful? Uri Wilensky and William Rand proposed three processes for evaluation: validation, verification, and replication. (Wilensky, 2015) Of the three, only validation and verification were germane to this project. Replication was left to further research.

Validation is "the process of determining whether the implemented model corresponds to, and explains, some phenomenon in real world." (Wilensky, 2015) For this study the real-world pattern for the model was the traffic grid of Uptown Charlotte. The question was whether the model would be able to handle simulation traffic similarly to the way the Charlotte grid handles actual traffic. The performance of any traffic grid is dependent on the traffic arrival rate and how efficiently traffic is able to exit the grid. The raw traffic data used for this project was obtained from the Charlotte Department of Transportation (CDOT) (Obregon, 2016), so successfully using it was an excellent validation test.

All of my previous research (Chapter 3. Previous Research) was concerned with the efficiency of a symmetric, right-angled grid where cars traversed the entire grid, entering on one side and exiting on the opposite side. The path all cars took was essentially equivalent. The average time on the grid for all cars was used to measure the efficiency of the grid. However, for the Uptown Charlotte grid, this was not a good metric. For the morning and evening rush hours many cars should not follow that traversal model. For instance, for the morning rush hour many of the cars should enter

the grid at the periphery and disappear without traversing it to represent the cars that stay within the boundaries of the grid for extended periods in parking lots or garages. Conversely, for the evening rush hour many cars should not enter the grid at the boundaries but rather from internal parking lots and garages where they were "stored" throughout the day.

We can assume that data derived from CDOT data represented arrival rates similar enough to actual external arrival rates and that traffic could be stored and released from parking facilities to accurately represent internal entrances and exits. The total number of arriving cars was those that entered from the edges plus those departing from parking lots and garages. The total number of exiting cars was those that exit at the edges plus those that entered parking lots and garages. The number of exiting cars was still a reasonable validation metric. It is my position that favorable results using this metric validated the model. Further discussion of steps that had to be taken to fine-tune and further validate the model is detailed in Chapter 6.8: Model Validation.

Verification is "the process of determining whether an implemented model corresponds to the targeted conceptual model." (Wilensky, 2015) The core program implementation that was the basis of the Charlotte grid model was my fifth version of a NetLogo (Wilensky, 1999) program to control traffic lights. Each succeeding version was a refinement of the previous. Each version produced either expected or explainable results. I believe the Charlotte version performed similarly, therefore verifying the model.

CHAPTER 6: THE SIMULATION ENVIRONMENT

This chapter discusses factors that must be reflected in the model and then proceeds to the discussion of the model itself.

6.1 Autonomy

What does it mean for a CAS agent to be autonomous? Specifically, what does it mean for a traffic light to be autonomous? In a laboratory environment it is easy to assume that every light has access to and can make decisions from traffic data of the entire grid. It is doubtful this would even be practical in real life without some sort of central controller, and it could be argued that it is neither necessary nor desirable. A light has direct control only over the cars behind it in its own block. Even if a light has knowledge of traffic in blocks behind it or in front of it, many things can alter when and if an individual car ever arrives at a light's block. A car may turn. It may be slowed by intervening lights. Cars ahead of it my turn, and new cars may turn into its path altering the timing of when it will arrive at the light. There are related concerns for predicting the arrival of traffic of opposing directions at the light's intersection. For these reasons for this study a light was considered autonomous when it made decisions based on traffic behind it in its own block.

6.2 Time Slices and Simulation Modes

Of a twenty four hour day three time slices were considered to be of interest for simulations runs. The ability to accommodate and recover from peak arrival rates is a good soft metric to measure the efficiency of a traffic grid. Therefore two time slices were chosen around the morning and evening rush hours. The middle of the day is also interesting because it starts with a morning peak and then could be expected to maintain equilibrium at a lower level until reaching the evening rush. The hours between evening rush and morning rush have relatively light traffic and are of little interest. The three time slices that reflected that traffic flow for this study were EarlyPeak from 5:00 AM through 9:00 AM, MidDay from 9:00 AM through 2:00 PM, and LatePeak from 3:00 PM through 6:00 PM. Note that EarlyPeak and MidDay both included the 9:00 AM hour in order to provide the peak and recovery situation mentioned earlier. All hours are inclusive. Also note that Early Peak's duration was five hours, MidDay's duration was six hours, and LatePeak's duration was four hours.

Table 1: Traffic Arrival Rates by Time Slice

Arrival Rates				
Time Slice	Hours (Inclusive)	Total Hours	Overall Average Car Arrivals	Hourly Average Car Arrivals
EarlyPeak	5:00 AM - 9:00 AM	5	52,386	9,960
MidDay	9:00 AM - 2:00 PM	6	46,234	7,311
LatePeak	3:00 PM - 6:00 PM	4	48,741	11,789

The heaviest hourly rate traffic occurred in LatePeak. EarlyPeak was the second heaviest even though it was one hour longer. The heaviest single hour was 5:00 PM in LatePeak.

There were four simulation modes. Basic mode (BM) was non-autonomous and algorithmic, simply a set duration go cycle. Near mode (NM) was based on a simple autonomous decision by individual lights based on cars waiting. Variable minimum go length modes (fixed, VMGF, and random, VMGR) tracked more environmental data points and allow the lights to learn.

6.3 Parameters for Intersection Geometry

Before addressing the details of the ABM for this project we must first understand in general how an intersection operates. The assumption was a standard four-way intersection, north-south and east-west. All lights operated in a green-yellow-red-green cycle with the north-south lights operating in tandem and always in the same phase. The east-west lights operated similarly. There were three parameters which controlled when the lights changed phase for both algorithmic and autonomous control.

The first was caution phase length. In actual intersections the caution phase is present strictly as a safety measure to allow traffic traveling in one direction to clear the intersection before traffic in the opposing direction is allowed in. The assumption for this study was that all cars stop for caution lights and there were no red light runners. The caution phase was essentially idle time for the intersection; so the performance of the grid would logically improve by making it as brief as possible, the minimum amount of time for the last car to cross when a light turns to caution. That would be the time required to cross the width in patches of the intersection. Eighty four per cent of the intersections on

the Uptown Charlotte grid were four by four. Adding one tick for a car that was only partially past a light which had just changed to caution produced an ideal caution phase length of five ticks. Earlier research also confirmed five as the optimum value, so all simulations of this study were run with a caution phase length of five.

The second parameter was designated the near factor, and it was defined as a number of patches behind a light. If there was at least one car in the near zone defined by the near factor, then the light would consider itself a candidate for a go cycle. If the near zone was not occupied, the light would not consider itself for a go. Intersection efficiency was very dependent on the tight relationship between the light's near factor and the caution phase length. If the near factor was greater than the caution phase length, the idle time of the intersection would be unnecessarily lengthened. If the near factor was less than caution phase length, the light might rotate to green before traffic had cleared the intersection. If the near factor and caution phase length were equal, opposing red-to-green and yellow-to-red light changes occurred simultaneously, and the intersection worked smoothly.

The third parameter defined the minimum length of a go phase, the min go length. If there was traffic attempting to pass through an intersection in all directions, once a light enters a go phase how long does it stay there? The objective was to avoid having short, choppy go phases with accompanying delays for each intervening caution or excessively long go phases which would unduly delay traffic in the opposite direction. In practice the min go length became a factor only when traffic was heavy in both directions, and we will discuss fine-tuning the min go length with caution phase length and near factor when heavy traffic conditions did occur.

VMGF and VMGR simulations also required the three basic parameters, but to facilitate learning there were four additional as well. Learning was based on variations of the light's min go length due to traffic accumulating or dissipating behind the light. A parameter, min go select factor, was used to determine whether the light's block was "full." For each tick, if the block was full, the min go length parameter was incremented by the value of the min go increment parameter. If the block was not full, the min go length parameter was decremented by the value of the min go decrement parameter. Each time the light went into a caution phase, the current value of the light's min go length was used in conjunction with the current through count of the intersection to establish the optimum value for the best min go length parameter for the light. One additional parameter was added for variable min go length simulations, min go max factor, which was used to set the maximum value of min go length. When the min go length parameter was being incremented, its upper limit was controlled by min go max factor. When it was being decremented, its lower limit was not allowed below its starting value.

6.4 Arrival Rates

A key component in determining the efficiency of any traffic grid is the traffic arrival rate, which may vary widely depending on a number of factors. The rush hours at the beginning and end of a business day are familiar. Weekday traffic might be heavier than weekend traffic. Traffic might increase for special events. It might be seasonal. It might also be affected by random events such as traffic accidents, detours, or road closures.

If a model uses a generic grid, it would be sufficient to simply create arrival rates designed to illustrate the desired operation of the grid. For instance to establish a performance baseline, a set of uniform arrival rates would be appropriate. To demonstrate a morning rush hour it would be logical to identify the primary incoming arteries and have the arrival rates increase on those arteries slowly from 4:00 AM until 6:30 AM. The rate of increase should then increase more rapidly until peaking around 8:30 AM before receding to a lighter, uniform rate between 9:00 AM and noon. The same type of peak should occur in reverse for the evening rush hour.

For a more specific environment such as Uptown Charlotte as used for this study, it was highly desirable to use actual traffic data from that grid; and the City of Charlotte Department of Transportation (CDOT) made its traffic volume data available. CDOT has used the Miovision system to gather traffic data for all intersections of Uptown Charlotte. Miovision allowed CDOT to report intersection activity every fifteen minutes recording right turns, left turns, and through traffic. For the simulations of this study these readings were distilled into hourly traffic arrival rates for each of the grid's entry points for each of the model's time slices. (Obregon, 2016)

6.5 Traffic Queues

At each entry point of the grid there was a designated entry patch. A car entered the grid and was counted when it was created on an entry patch. The timing for creating new cars was based on arrival rate of an entry point, and it was unlikely that the arrival rate would require the creation of a car at every tick. During peak arrival times it was possible that a newly created car would not be able to move off the entry patch because

of heavy traffic ahead of it. NetLogo does allow multiple cars to occupy the same patch, but obviously this does not occur on actual streets. Allowing it in the model would have distorted the statistics of the grid. In real life the traffic would simply back up on the streets leading into the grid, and that overflow would dissipate naturally as traffic was eventually absorbed into the grid.

The model managed this situation by maintaining traffic queues at each entry point. When the entry patch was not empty, instead of actually creating a new car it was symbolically added to the patch's queue. For subsequent ticks, if no new car was due and there was a car in the queue, the queue count was decremented and the queued car was created.

In practice the queues did not come into play unless traffic was heavy, and then they prevented the grid itself from having to contain unrealistic traffic volumes. However excessively large queue counts did indicate that the grid extended was being stressed and might be an indication that light management was not performing efficiently.

6.6 Uptown Charlotte Model

In Chapter 5: Agent Based Modeling an ABM model was defined as having three basic components: environment, agents, and behaviors. This simulation model met these requirements. The environment was implemented with NetLogo and was based on the traffic grid of Uptown area of Charlotte, North Carolina. The grid contained 29 streets, one light rail line, and approximately 149 intersections. There were two-, three-, and four-lane streets. There were one-way and two-way streets of each type. There were 54 entrances to the grid and 65 exits. (Entrances and exits were defined in terms of lanes.

For instance, a street with two lanes leaving the grid counted as two exits.) There were 50 parking entrances and 132 parking exits. There were six streets that had been converted over time to include internal dead ends. In some cases there were multiple dead ends on what was once a through street. These dead ends could have in theory also be considered entrances and exits although all their entrance counts for this study were zero.

The grid was drawn to scale based on the dimensions of the 2015 Charlotte Uptown map by Jeff Simpson. (Simpson, 2015) It was bounded on the north by 11th Street, on the south by Stonewall Street, on the east by McDowell Street, and on the west by Graham Street. An image of the model interface is included as Appendix A.

To meet the ABM agent requirement there were two types of agents in the model: traffic signals and cars. There were three types of traffic signals: traffic lights, stop signs, and train signals. In total, the number of signals was approximately 700. (Similarly to the way entrances and exits were counted, each signal controlled a single lane. If there was a two-lane, one-way street being controlled by a traffic signal, it was be counted as two lights.) All signals were represented by colors on appropriate environment patches.

Stop signs and train signals operated the same in all modes. Cars approached a stop sign, stopped, and then merged into traffic when there was an opening. Train signals were triggered to turn red when a train approached a crossing and turn green when the train cleared the crossing.

Cars were agents of secondary interest. Each car was represented by a car-like icon that was scaled to match the Simpson map. Car movement was simulated by

relocating the icon from patch to patch across the grid. Cars stopped, sped up, slowed down, or moved as allowed by the signals and other traffic. When a car at rest began to move, it gradually accelerated from zero miles per hour to a scaled maximum speed of 20 miles per hour.

All vehicles were seen as cars or trains. Trains operated only on train tracks. Only cars operated on streets. All cars were considered the same. There were no high-priority emergency cars. There were no bicycles, motorcycles, large trucks, large vans, buses, or pedestrians. All agents operating on the streets were cars scaled to a length of fifteen feet.

Cars did not make autonomous decisions and as such were not considered autonomous agents. Car behaviors were not defined or considered in this model. In normal traffic, cars might turn depending on decisions of their drivers. It was not the intent of this project to investigate the motives of drivers or to optimize the routes to destinations; but in order to represent normal traffic patterns, some cars should turn. This was accomplished by generating random turn decisions for each car, based on random turn tendencies assigned when a car entered the grid. Cars did not learn.

To complete the ABM requirements, behaviors were identified for the light agents. The behaviors of primary interest were the basis for the lights to decide when to demand a go cycle. In BM mode the go decision was made by an overall system controller not by the lights. In NM mode that decision was made by individual lights based on the number of cars waiting in its near zone. Learning in VMGF and VMFR modes required that the light's decision factors to expand to include the variation of the

count of cars waiting in its block. The behaviors of the lights are discussed in detail in the next section

6.7 Autonomous Light Agent Behavior Rules

Referring to the working definition of a CAS from Chapter 4: Complex Systems, a key component is a preferably small, preferably simple set of behavior rules of the agents. This is the cornerstone of the goal stated in Chapter 5: Introduction that an ABM will be "easier to design, implement, execute, and extend" than a comparable EBM.

No rules were required for BM simulations. There were only three rules for NM simulations. For VMGF and VMGR simulations there were four rules, the original three plus one additional. NM simulations were controlled by the lights of an intersection by the relationship between the caution phase length and the near factor. There was a single min go length for all lights which came into play only during time of very heavy traffic around the intersection. There were three original rules:

1. If there is near traffic only in the current go direction, change nothing.
2. If there is near traffic only in the current stop direction, begin the caution/stop phase regardless of the current go phase duration.
3. If there is near traffic in both directions and the current go phase duration is less than or equal to min go length, change nothing. If there is near traffic in both directions and the current go phase duration is greater than min go length, begin the caution/stop phase.

VMGF and VMGR simulations had the same relationship between caution phase length and near factor, but they also reacted to the accumulation or reduction of cars

behind the light. Each light had its own min go length, and under certain conditions the value of min go length was incremented or decremented. The additional rule for VMGF and VMGR simulations was as follows:

4. Count all the cars in the block behind the light. If the count is greater than the count from the previous tick, increment min go length. If the count is less than the count from the previous tick, decrement min go length. At the beginning of each caution phase for the light, recover the count of cars through the light during the previous go phase. If the count is greater than current best through count, update the best through count and the best through count min go length. For both VMBF and VMGR the up-side value of min go length is limited by the min go max factor parameter.

6.8 Model Validation

Does the model "correspond and explain some phenomenon in real world" is the question posed earlier when we introduced the concept of validation. At that point the only factors we had to consider were a grid arranged and scaled to match Uptown Charlotte and traffic data derived from official CDOT readings. That seemed like an excellent start, but the results of initial BM and NM simulation runs were abysmal. Study of the bottlenecks then present on the grid revealed obvious design and implementation errors. The review of those errors and their resolutions were instructive toward model validation, and correcting those problems produced a better model.

The original version of the grid was drawn in such a way that when matched with the NetLogo environment, the top scaled speed for a car was ten miles per hour. In

retrospect, this was not realistic and simply too slow to allow cars to clear the grid. (This was definitely not a good limitation if the primary validation metric was to be exit count.). The model simply bogged down. Rescaling and redrawing the grid to provide for a 20 miles per hour top speed was a significant improvement, but it was not a final fix. It solved the speed problem, but then revealed other flaws.

Even when using official arrival rates, some of the model's intersections were simply overwhelmed. Typically each situation was different, so there were several solutions. For instance one intersection considered through traffic to be just entering the grid when actually it had entered at an upstream intersection. Entry traffic was essentially being doubled at the problem intersection. At another intersection the traffic counts were for four lanes, but it was being forced to enter via only two. Still another instance was a single lane that should have been one straight lane plus a dedicated turn lane. Correcting these errors improved but still did not completely solve the overall problem. There were still too many cars trying to occupy too small a space.

Refining the random selection process for turning cars allowed more cars opportunities to turn off the main entrance arteries. Allowing cars to change lanes avoided the occurrence of full lanes alongside empty lanes behind lights and produced a natural clumping of waiting cars at red lights. Adding parking lot entrances allowed cars to exit as they would normally do in large numbers during EarlyPeak. The converse was true for parking exits which allowed parkers to quickly enter and exit the grid during LatePeak.

The result was a model which handled Charlotte traffic well. It would be expensive to design a traffic system to handle the short peaks with absolutely no

congestion. It would be more reasonable to design a system that might struggle a bit at short peak demand, but recover when the peak subsided. That is what the actual Uptown Charlotte grid does. That is what this model did, further evidence it was validated.

CHAPTER 7: SIMULATIONS – PLAN AND RESULTS

Simulations were run in four sequential groups corresponding to the four simulation modes. Each succeeding group built on the results of the previous. The simulations were run on a variety of Windows 10 PC computers using the model built on the NetLogo Version 5.1.0.

7.1 Baseline Simulations

The baseline group was designed to contribute to model validation and to compare elemental BM and NM performance. It was composed of individual simulations for each hour for each time slice in both BM and NM modes. The objective was to exploit all appropriate combinations of the basic intersection geometry parameters described in Chapter 6, caution phase length, near factor, and min go length. The value of caution phase length for all simulations was set at five.

There were 75 BM hourly simulations (15 hours x 5). The near factor parameter was not used for BM. There was a simulation run with min go length parameter values of 20, 35, 50, 65 and 90 for each hour.

There were 300 NM hourly simulations (15 hours x 4 x 5). The near factor parameter values were 5, 10, 15, and 20. The min go length parameter values were 20, 35, 50, 65 and 90.

7.2 Near Factor Simulations

The near factor group was designed to evaluate performance over an entire time slice varying near factor, and min go length. Again, the value of caution phase length for all simulations was set at five.

There were 84 NM slice simulations (3 slices x 4 x 7). The near factor parameter values were 5, 10, 15, and 20. The min go length parameter values were 20, 35, 50, 65, 90, 105, and 120.

7.3 Variable Min Go Simulations

The variable min go simulation group was designed to allow a light to improve its performance by learning the optimum value of its min go length parameter. Simulations were run over an entire time slice for the most promising parameter combinations from the Near Factor group. Caution phase length and near factor were both set to five for all simulations. The min go length parameter varied based on learning.

Meaningful variations of the min go length parameter occur only during times of high volume traffic. This threshold was seldom reached during MidDay. To determine if a full set of variable min go simulations for MidDay was justified, a small subset of simulations across a representative set of parameters values was run (six selected three-run sets). These runs showed that while the exit percentage was very high (99+ percent) Rule 4 was invoked less than 1 percent of the time. The conclusion drawn was that the MidDay traffic volumes were not high enough to benefit from learning. Therefore no further

MidDay simulations were run. VMGF and VMGR simulations were run only for EarlyPeak and LatePeak.

There was a total of 162 three-run sets of VMGF simulations, 81 each for EarlyPeak and LatePeak (2 slices x 3 x 3 x 3 x 3). For each of three starting values for the min go length parameter, 50, 65 and 90, values for the min go select factor parameter were set in turn to 0.65, 0.75, and 0.85. The values for the min go increment parameter were three, five and eight. For a min go increment of three, the min go decrement parameter had values of one, two and three. For a min go increment of five the min go decrement parameter had values of one, three, and five. For a min go increment of eight, the min go decrement parameter had values of one, four, and eight.

Using the best three VMGF results for each of the starting min go lengths for both time slices, three VMGR three-run simulation sets were run with the same parameter combinations except for a random starting min go length. The best result for each time slice was rerun for an extended six-run simulation set.

7.4 Simulation Results

At the end of the Baseline simulations corresponding BM and NM exit percentages were compared for each individual hour of the three time slices. The NM runs were superior in all cases. In general, the queue counts for the BM runs were high, and the NM queue counts were very low. The exit percentage for NM simulations was in the 90 – 94 percent range.

The Near Factor simulations were run with the same parameter combinations over all three time slices. The objective was to find the combinations that produced the best

exit percentages for an entire time slice. The exit percentages for all time slices were in the 95 – 99 percent range. Generally, all exit percentages within a time slice were similar. The queue counts for MidDay and LatePeak were zero. The queue counts for EarlyPeak were not zero, but neither were they alarming. The best NM parameter combinations, designated the Variable Min Go Test Set, were chosen from those with min go lengths of 50, 65, and 90 based on a combination of factors: exit percentage, a contiguous range of parameter combinations, and queue count.

The Variable Min Go simulations were designed to demonstrate that autonomous lights could learn from past success and adapt their behavior. There were nine three-set VGMF simulation runs for each of the Variable Min Go Test Set parameters from the Near Factor simulations. All of the VMGF runs for EarlyPeak with min go length of 50 and 65 improved the exit percentage by small but consistent amounts. For min go length of 90, only one of the nine results failed to exceed the best NM result. The remainder improved by small but consistent amounts. Results were similar for LatePeak. Only one LatePeak VMGF simulation (from the min go length 65 group) failed to improve corresponding NM results. As with EarlyPeak, the improvements were small but consistent. Generally the overall exit percentage range was 98 – 99 percent. The queue counts for LatePeak were all zero. There were some occasional high queue counts in EarlyPeak, but most were zero to low.

The six best VMGF simulations were rerun as three-segment VMGR simulations with randomly generated initial min go lengths. The best three-segment for each time slice was re-run for an extended six-segment simulation. The VMGR exit percentages were essentially the same as the standard set by the VMGF runs, but none exceeded the VMGF

runs. The extended VGMR runs generally matched their shorter mates. The exit percentages were in the same range as the VMGF results, but there were a number of queue counts that indicated that traffic was not moving smoothly outside the grid.

There are two appendices at the end of this document containing data tables from these simulations. Appendix B contains two tables, Table 2 and Table 3. Table 2 summarizes the progression of results from Near Factor, Variable Min Go Fixed, and Variable Min Go Random simulations for the EarlyPeak slice. Table 3 contains a similar progression from LatePeak. Appendix C contains samples of the data collected from individual from Near Factor, Variable Min Go Fixed, and Variable Min Go Random simulations.

CHAPTER 8: EVALUATION

During the course of this research we have defined a complex adaptive system and identified a signature performance characteristics of such a system. We have defined agent based modeling and related it to complex adaptive systems. We have discussed what is necessary to validate and verify such a model. We have described the design and implementation of a specific ABM of the Uptown Charlotte traffic grid and laid out justifications for its validation and verification. We have described a series of simulations run through that model, and I contend those simulations confirm that the model did operate as a CAS.

8.1 CAS and Adaptation

The concept of adaptation is key for a CAS, and two types should be considered. In our definition of a CAS we required that the operation be non-linear, non-algorithmic. The agent behavior rules should allow the agents to react but should not absolutely dictate their behavior. (An example of algorithmic behavior would be a traffic light with fixed phase lengths which after a specific time interval always changes from green to red regardless of the environment.) In general, if the agent begins in state zero (s_0), and could possibly move to any of n other states ($s_1 \ldots s_n$), the rules should not require moves to other states in a specific time or sequence. The classic sheep-wolf-grass model is an example of this type of simple adaptive but non-learning behavior.

However, traffic lights are a special case. The sequence of the states is always the same (go, caution, stop, go), but the duration of the go/stop phases might vary. The rules should allow the light to gather environmental data and change state based on its interpretation those data. The agents adapt, but those adaptations are always based on the same rules and parameter values. The length of a state may vary, but the rules do not.

The second type of adaptation is related to learning. Instead of having a set of rules and values that are static, the light keeps track on an ongoing basis of the set of light parameters that have produced the best throughput results and when necessary dynamically changes the rules or parameter values themselves based on those results.

As the simulations of this study progressed from group to group, the level of adaptation progressed. The BM simulations were algorithmic. There was no adaptation at all. The original three rules of the NM simulations allowed a light to progress from green to yellow to red and back to green state in an unalterable sequence, but the times spent in the green and red phases were unpredictable. The lights adapted the length of the green and red phases based on a static rules set and changing environmental conditions. They were autonomous and adaptive but did not learn.

The VMGF and VMGR simulations extended the behavior rule set. Where the NM simulations had a system-wide, fixed min go length parameter, the VMGF and VMGR simulations allowed each individual light's min go length parameter to increase and decrease as the traffic behind the light rose and fell. Theoretically, the minimum time for a go phase would increase to allow more of that waiting traffic to pass through the light. The light would store the through count/best min go length combination that produced that highest through count. That most productive best min go length value

would be used for subsequent phases until a new high through count was recorded. Rule 4 allowed individual lights to dynamically improve their performance by learning which parameter combinations were most efficient. In a different sense, they were also autonomous and adaptive. They also learned. I consider both the adaptive and adaptive/learning types to be CAS.

8.2 Is This Model A CAS?

Does the Uptown Charlotte model "correspond to the targeted conceptual model", the real Charlotte grid. I believe it does on the basis of the same criterion I used previously with earlier models: Generally it performs as expected; but when there are unexpected results, they logically lead to corrected assumptions.

A model grid scaled to match the actual grid responding to traffic loads derived from actual data failed to perform adequately when there were implementation errors. However, when the errors were corrected it operated easily when traffic was moderate to light, strained when anticipated peaks arrived, and then returned to normal quickly when the peaks passed.

Is it a CAS? We have asked this question at various stages in the discussion of this project, and each time my answer has been yes. At this time I would like to refer back to Chapter 4 and present more CAS-supporting evidence.

In Chapter 4 we introduced a pair of analysis tools that could be used to assess the degree to which a given system has characteristics of a CAS, the cellular automata of Wolfram and Langston and Feigenbaum's logistic map. NetLogo and therefore this model are examples of a cellular automata. Wolfram observed there are four classes that

can be used to describe the behavior rules of any cellular automaton. Classes one and two might initially show a tendency toward a dynamic self-organization, but eventually stagnate. Class three exhibits wild gyrations and lack of organization. Class four produces cells that self-organize and continually interact with each other. Langston named class four Complexity and described it as acting on the edge of chaos. (Waldrop, 1992)

During the early development of the model there were, of course, logic bugs in the code for the agent behavior rules. Some versions of the behavior code produced simulation runs where certain lights changed phase perhaps one time or two times and then seemed to lock and never change again. There were other versions where lights seemingly changed phase every tick, and the changes were not necessarily in the proper sequence. It produced turmoil on the grid. Certainly the lights did not "know" their behavior did not match the behavior of an efficient traffic grid. The lights did not "know" there were errors in the definition of the rules, nor did they "know" when they were operating properly after the errors had been corrected. They were merely operating with the rules as defined. I suggest that the miscoded rules that quickly locked up the grid were simply an indication that at that time the grid was operating as a class one or class two cellular automaton, Langston's Order. Erroneous rules that produced a grid with seemingly too many phase changes with no apparent purpose was actually as a class three cellular automaton, Langston's Chaos. When the errors were corrected, the lights began interacting smoothly and dynamically managed the traffic. I believe at that point the grid began operating as a class four cellular automaton, Langston's Complexity, and has been operating as a CAS at the edge of chaos since.

Further evidence of CAS behavior can be found in the analysis of the logistic map function. Figures 2 and 3 show plots of the number of cars on the grid over time during characteristic EarlyPeak Near and LatePeak Near simulations. If the model operated as a CAS and if the logistic map is a valid indicator of a CAS, then obviously the plots of Figures 2 and 3 should resemble the earlier plot of the logistic map (Chapter 4: Complex Systems, Figure 1). Casual comparisons might indicate this is not the case, but a closer inspection is warranted.

The Table 1 logistic map chart depicts one continuous time interval with constant input. Both the EarlyPeak and LatePeak charts depict multi-hour simulations (five-hour for EarlyPeak and four-hour for LatePeak) with different traffic arrival rates for each hour. Looking specifically at the EarlyPeak chart, notice that it initially arises from zero and quickly reaches a plateau where it stays until it begins to rise again near tick 3,600. (Tick 3,600 is the end of the first hour and the beginning of the second hour.) The second hour ends at tick 7,200 when the rise-plateau sequence repeats again two more times for hours three and four, respectively. At the end of hour four the curve falls and plateaus again at a lower level. This behavior matches the typical arrival rate for EarlyPeak. The EarlyPeak curve is actually a combination of five hourly curves; and if the hourly curves are compared individually to the logistic map curve, they adequately match. Each hour of EarlyPeak individually has the CAS signature. Analyzing the LatePeak curve reveals a similar pattern. The ragged nature of the simulation curves when compared to the smooth logistic map curve is due to the fact that traffic arrivals are not continuous.

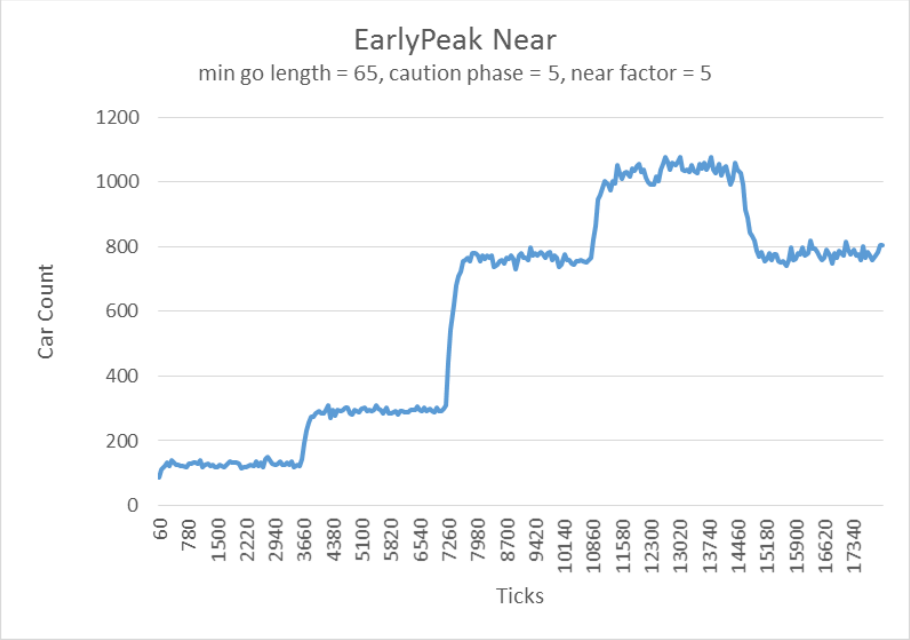

Figure 2: EarlyPeak Logistic Map Curve

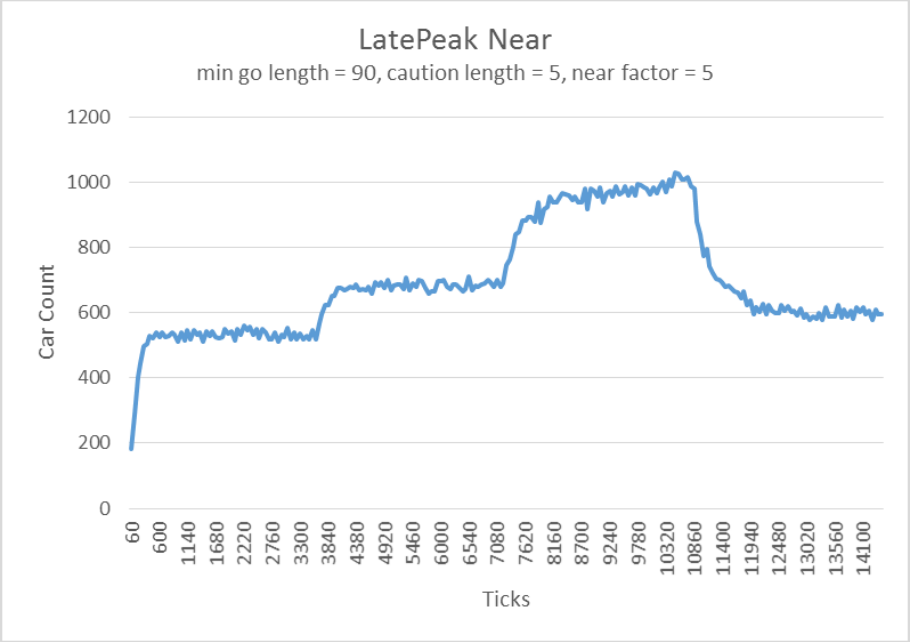

Figure 3: LatePeak Logistic Map Curve

In NetLogo terms, at an individual entry point cars are not necessarily created at every tick.

8.3 EBM vs. ABM

If the structure of an EBM is visualized as a graph, then the basis of the model is the relationship among the vertices of the graph, edges expressed as equations. If there are edges connecting each pair of vertices, then the graph is complete. If the number of vertices is n, then the number of equations required to build the EBM is $\frac{n(n-1)}{2}$. If the number of vertices is increased, the number of equations grows quickly. A graph of five vertices would require ten relationship equations. If five additional vertices are added to that model, the number of equations jumps to 45. If the Uptown Charlotte grid with approximately 700 traffic signals and signs as agents were to be implemented as an EBM, more than 200,000 equations would possibly be required. Obviously maintaining such a model would be cumbersome and executing it would be computationally expensive.

At the outset, creating the Uptown Charlotte model as an ABM was much more straightforward. Identify the agents, the 700 lights and stop signs. Create their behavior rules, the original three were subsequently increased to four. The code for all four rules was implemented with fewer than 40 lines of well-spaced, well-documented NetLogo code. Expectations stated earlier in Chapter 1 Introduction was that an ABM would be "easier to design, implement, execute, and extend", and I initially expected it would be an easy call. That may still be true, but there are some qualifications.

NetLogo is a cellular automaton. Its design and implementation are very good for models with turtles on a patch interacting equally with the eight surrounding patches.

NetLogo seems featured and tuned to facilitate that type of relationship. However, traffic grids require the turtles (cars) to proceed in straight lines, turn at right angles, and stop and start in response to traffic signals and other turtles on specific patches. It is not reasonable for multiple turtles to occupy the same patch. A turtle directed in error is not necessarily constrained by roadways and lights indicated only by color.

I came to NetLogo as an experienced designer, programmer, and implementer. I would describe NetLogo and its syntax as feature-rich but idiosyncratic and difficult to use. It has hints of object oriented principles, but it is not object oriented. The documentation is limited: a dictionary and a brief hit-and-miss programming guide. It works well for refreshing one's knowledge on a specific feature but is of limited use as a general learning tool or an overall architecture or design guide. It is a rich but ultimately frustrating platform. It is, however, free.

The conceptualization of the Uptown Charlotte model was straight forward, but the implementation was a frustrating struggle. Building the model as a one-off was more difficult than it needed to be. Much of that frustration was having to build the tools to build the model within NetLogo, and at this point I do not believe building an ABM vs. an EBM is a simple decision. If there were easily accessible front-end tools to build the environment interface, I would likely believe differently. Building the ABM model is easier…building a one-off graphic environment is not.

8.4 Platoons, Convoys, and Waves

Earlier discussion (Chapter 2: Background Research) mentioned self-organizing groups of cars variously called platoons, convoys, or waves. Some of the cited research

projects had tooled their models to promote these groups with the assumption that they increased the efficiency of the grid. I pointed out that my experience was that these groups tended to self-organize organically, spontaneously when the focus was only on the efficiency of the autonomous lights. This was true with the various simulation groups with the Uptown Charlotte model as well when four key conditions were met.

First, obviously there must be sufficient traffic volume. It is just not a platoon with only three of four cars. Second, the stop phases of the light must be long enough for the cars to congregate behind a light. Third, the go phase must be long enough to allow the platoon to stay together moving through the light. Finally, subsequent lights must repeat conditions two and three. My observation is that the autonomous lights operating as described earlier effectively did just that without communicating with each other. Platoons spontaneously self-organized without special prompting.

CHAPTER 9: CONCLUSIONS

9.1 Contribution and Implications

The stated goal at the beginning of this research was to build an ABM model of the Uptown Charlotte grid and test whether it could operate as a basic CAS with the traffic lights operating as autonomous agents. The second goal was to test whether the agents could learn from past success and improve their individual performance and thus collectively the performance of the entire grid. The third goal was that this learning could be real-time, without a recalculation-reload step associated with EBM models and that the ABM model would be easier to design, implement, and execute than a comparable EBM model.

The grid, scaled to represent Uptown Charlotte and tested with actual Charlotte data, did perform as the actual grid performs. For each time slice it was able to absorb and recover promptly from peak loads; and it did so operating as a CAS.

As designed, during the NM simulations the agents adapted but did not learn. During the VMGF simulations the traffic light agents adapted, learned, and improved their performance by monitoring the traffic accumulating in the block behind the light and modifying the value of a key parameter to take advantage of productive parameter combinations. The learning improvements were not large, but they were consistently

better than the results of the non-learning rule set. There was no significant difference between VMGF and VMGR results.

I believe there were three factors that influenced the size of the improvements. The first was that the actual Charlotte grid (and thus the model grid) was relatively efficient even without learning. There was not much room for improvement. For EarlyPeak and LatePeak both the pre-learning and post-learning exit percentages were in the 98 percent range, but the post-learning percentages were typically 0.01 – 0.03 percent higher.

The hours within each time slice influenced the near 100 percent exit count as well. Recall that each slice was designed to show the response to and recovery from a traffic peak. There were no double peaks. The peak within a slice was always followed by at least one hour with less traffic allowing the peak level to dissipate. If the grid were not inherently efficient, it would not have recovered post-peak; and the exit percentage would not have approached 100.

The third factor was that the learning rule, Rule 4, was invoked at an intersection only when traffic began to increase in opposing directions. In practice even for the EarlyPeak and LatePeak slices this happened at fewer than ten intersections.
Most of the intersections never met the traffic volumes required to learn, and Rule 4 was never invoked.

9.2 Future Work

The model works for Charlotte, but it is still a general model. The grid itself is designed to match Charlotte, but the agent behavior rules are not specific to any

particular grid. In theory, applying the model to a grid designed to match another city using actual data from that city would produce results specific for that city. In fact if that new grid were less efficient than the Charlotte grid, it would be an interesting question to pose whether the learning delta would be larger for the new grid due to the greater opportunity for improvement. For this project there was no effort put to creating another grid. That will be left to future work. If such a project were undertaken my suggestion would be to resist the impulse to begin creating the grid itself at the outset. The first tasks should be to develop the front-end tools to automate the definition of the model environment anticipating the creation and future maintenance of the grid itself.

REFERENCES

Barrero, F. et al., 2010. Internet in the development of future road-traffic control systems. *Internet Research,* 20(2), pp. 154-168.

Bazzan, A., 2009. Opportunities for multiagent systems and multiagent reinforcement learning in traffic control. *Autonomous Agents and Multiagent Systems,* Volume 18, pp. 342 - 375.

Bazzan, A. L., de Oliveira, D. & daSilva, B. C., 2010. Learning in groups of traffic signals. *Engineering Applications of Artificial Intelligence,* Volume 23, pp. 560 - 568.

Bazzan, A. L. & Klugl, F., 2013. A review on agent-based technology for traffic and transportation. *The Knowledge Engineering Review,* 29(3), pp. 375-403.

Brockfeld, E. B. R. S. A. S. M., 2001. Optimizing traffic lights in a cellular automaton model for city traffic. *Physical Review E,* 64(5), p. 56132.

Buchanan, M., 2007. *The Social Atom Why the Rich Get Richer, Cheaters Get Caught, and Your Neighbor Ususlly Looks Like You.* 1 ed. New York: Bloomsbury.

Cai, C., Wong, C. K. & Heydecker, B. G., 2009. Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C,* Volume 17, pp. 456-474.

Cools, S.-B., Gershenson, C. & D'Hooghe, B., 2008. Self-organizing traffic lights: a realistic simulation. *Advances in Applied Self-organizing Systems,* pp. 41 - 50.

Cui, C., Takamura, M. & Lee, H., 2011. Traffic signal control for a multi-forked road. *Artificial Life and Robotics,* Volume 16, pp. 253-257.

De Martino, A., Marsili, M. & Mulet, R., 2004. Adaptive drivers in a model of urban traffic. *Europhysics Letters,* 65(2), pp. 283-289.

de Oliveira, D. & Bazzan, A. L., 2006. Traffic Lights Control with Adaptive Group Formation Based on Swarm Intelligence. *ANTS 2006, LNCS 4150,* pp. 520-521.

de Oliveira, D., Bazzan, A. L. & Lesser, V., 2005. Using Cooperative Mediation to Coordinate Traffic Lights: A Case Study. *AAMAS '05.*

de Oliveira, D., Ferreira, P. R. J., Bazzan, A. L. & Klugl, F., 2004. A Swarm-Based Approach for Selection of Signal Plans in Urban Scenarios. *ANTS 2004, LNCS 3172,* pp. 416-417.

Dong, C., Ma, X. & Wang, B., 2010. Advanced information feedback strategy in intelligent two-route traffic flow systems. *Science China,* 53(11), pp. 2265-2271.

enotes, n.d. *Approximately How Many Cars Were Registered.* [Online]
Available at: http://www.enotes.com/homework-help/approximately-how-many-cars-were-registered
[Accessed 7 September 2016].

Gershenson, C., 2005. Self-organizing traffic lights. *Complex Systems,* Volume 16, pp. 29 - 53.

Hadzikadic, M., 2015. *ITIS8500-091 Class Notes.* s.l.:UNCC.

Hare, M. D. P., 2003. Further towards a taxonomy of agetn-based simulation models in environmental management. *IMACS,* Volume 64, pp. 25-40.

Jacob, C. & Abdulhai, B., 2010. Machine learning for multi-jurisdictional optimal traffic corridor control. *Transportation Research Part A,* 44(2), pp. 53-64.

James, R., 2009. America: Still Stuck in Traffic. *Time*, 09 07.

Kammoun, H. M. et al., 2014. Adapt-Traf: An adaptive multiagent road traffic management system based on hybrid ant-hierarchical fuzzy model. *Transportation Research Part C,* Volume 42, pp. 147-167.

Klugl, F. B. A. L. C., 2012. Agent-Based Modeling and Simulation. *Association for the Advancement of Artificial Intelligence,* Volume Fall 2012, pp. 29-40.

Macal, C. M. N. M. J., 2006. Introduction to Agent-Based Modeling and Simulation. *MCS LANS Informal Seminar*, 29 11.

Mitchell, M., 2009. *Complexity A Guided Tour.* 1 ed. New York: Oxford University Press.

North Carolina Department of Transportation, n.d. *Traffic Surveys.* [Online]
Available at: https://www.ncdot.gov/projects/trafficsurvey/
[Accessed May 2016].

North Carolina Department of Transportation, n.d. *Traffic Volime Maps.* [Online]
Available at: https://www.ncdot.gov/travel/statemapping/trafficvolumemaps/
[Accessed May 2016].

Obregon, F. A., 2016. *Personal email.* s.l.:Charlotte Department of Transportation (CDOT).

Parunak, H. V. D. S. R. R. R. L., 1998. Agent-Based Modeling vs. Equation Based Modeling: A Case Study and Users' Guide. *Proceedings of Multi-agent systems and Agent-based Simulation,* pp. 10-25.

Piotrowski, S. P., 2016. *Personal email.* s.l.:North Carolina Department of Transportation (NCDOT).

Rochner, F. B. J. M.-S. C. S. H., 2006. An Organic Architecture for Traffic Light Controllers. *Proceedings of the Informatik 2006 - Informatik fur Menschen*, P-93(Lecture notes in informatics), pp. 120-127.

Santa Fe Institute, n.d. *Santa Fe Institute.* [Online]
Available at: http://www.santafe.edu
[Accessed May 2016].

Simpson, J., 2015. *Charlotte Uptown.* Charlotte: SimpsonMaps.com.

Steingrover, M. S. R. P. S. N. E. B. B., 2005. Reinforcement Learning of Traffic Light Controllers Adapting to Traffic Congestion. *Proceedings of the 17th Belgium-Netherlands Conference on Artificial Intelligence*, Volume October 2005, pp. 216-223.

Stoller, G., 2012. New Report: Road congestion wastes 1.9 billion gallons of gas. *USA Today*, 25 03.

U.S Department of Transportation - Federal Highway Administration, n.d. *Traffic Signal Timing Manual Chapter 6 - Office of Operations.* [Online]
Available at: http://www.ops.fhwa.dot.gov/publications/fhwahop08024/chapter6.htm#6.3
[Accessed 22 November 2015].

U.S. Department of Transportation - Federal Highway Administration, 2009. *Manual on Uniform Control Devices (MUTCD).* [Online]
Available at: http://mutcd.fhwa.dot.gov/pdfs/2009r1r2/pdf_index.htm
[Accessed 22 November 2015].

Waldrop, M. M., 1992. *Complexity The Emerging Science at the Edge of Order and Chaos.* 1 ed. New York: Simon & Schuster.

Werbach, A., 2013. The American Commuter Spends 38 Hours a Year Stuck in Traffic. *The Atlantic*, 06 02.

Wiering, M., 2000. Multi-Agent Reinforcement Learning for Traffic Light Control. *Proceedings of the 17th Conference on Machine Learning*, pp. 1151-1158.

Wilensky, U., 1999. *NetLogo.* s.l.: Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL..

Wilensky, U. W., 2015. *An Introduction to Agent-Based Modeling: Modeling Natural, Social, and Engineered Complex Systems with NetLogo.* 1 ed. Cambridge, MA: The MIT Press.

Xu, Y. Z. Y. L. M., 2014. Multiagent Based Decentralized Traffic Light Control for Large Urban Transportation System. *Mathematical Problems in Engineering*, Volume 2014, p. ID 104349.

Zheng, X., Recker, W. & Chu, L., 2010. Optimization of Control Parameters for Adaptive Traffic-Actuated Signal Control. *Journal of Intelligent Transportation Systems,* 14(2), pp. 95-108.

APPENDIX A: MODEL INTERFACE

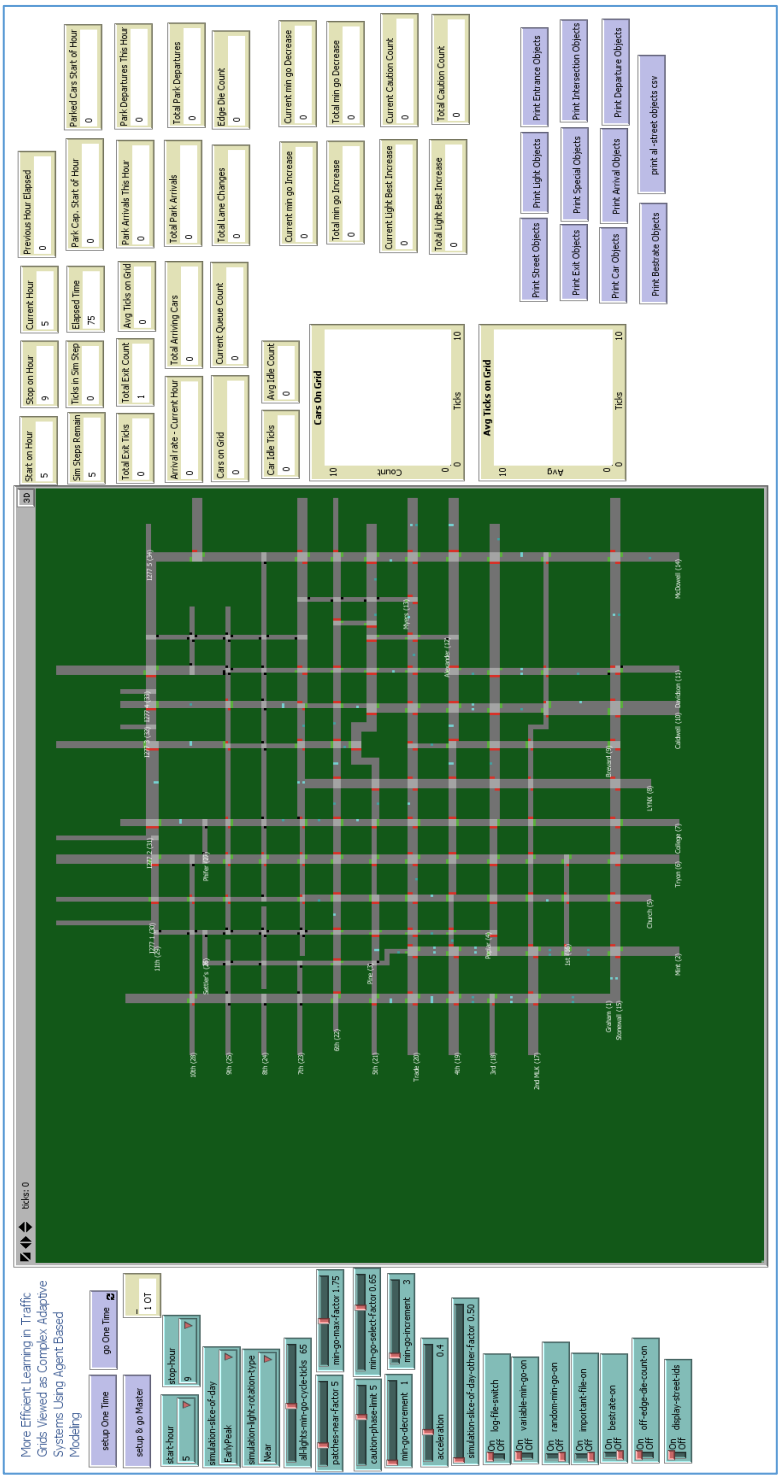

Figure 4: NetLogo model interface

APPENDIX B: FIXED VS. VARIABLE MIN GO SUMMARY TABLES

Table 2: EarlyPeak Near – Summary Data from Near vs. Variable Min Go simulations

EarlyPeak Near								
Fixed vs. Variable Min Go								
Fixed			Min Go Parms			Fixed Initial	Random	
Initial Min Go	Near	High Exit Per Cent	Min Go Incre.	Min Go Decre.	Min Go Select Factor	High Exit Per Cent	High Exit Per Cent x3	High Exit Per Cent x6
50	5	98.51	3	1	0.85	98.53		
50	10	98.44	3	2	0.65	98.55		
50	15	98.19	3	3	0.65	98.53		
50	20	97.98	5	1	0.65	98.55	98.49	
			5	3	0.65	98.54		
			5	5	0.65	98.57	98.49	
			8	1	0.85	98.54		
			8	4	0.65, 0.85	98.55	98.52	98.05
			8	8	0.85	98.53		
65	5	98.47	3	1	0.65, 0.75, 0.85	98.52		
65	10	98.39	3	2	0.85	98.53		
65	15	98.19	3	3	0.85	98.53		
65	20	97.64	5	1	0.75	98.52		
			5	3	0.75	98.52		
			5	5	0.75	98.55	98.52	
			8	1	0.65, 0.75	98.55	98.53	98.54
			8	4	0.65	98.56	98.51	
			8	8	0.75	98.53		
90	5	98.55	3	1	0.85	98.54		
90	10	98.29	3	2	0.75, 0.85	98.52		
90	15	98.07	3	3	0.65	98.55	98.50	
90	20	97.24	5	1	0.75	98.53		
			5	3	0.75, 0.85	98.53		
			5	5	0.85	98.54		
			8	1	0.75	98.55		
			8	4	0.65	98.56	98.55	98.42
			8	8	0.65	98.55	98.51	

Table 3: LatePeak Near – Summary Data from Near vs. Variable Min Go simulations

LatePeak Near								
Fixed vs. Variable Min Go								
Fixed			Min Go Parms			Fixed Initial	Random	
Initial Min Go	Near	High Exit Per Cent	Min Go Incre.	Ming Go Decre.	Min Go Select Factor	High Exit Per Cent	High Exit Per Cent x3	High Exit Per Cent x6
50	5	98.76	3	1	0.65	98.76		
50	10	98.65	3	2	0.85	98.77		
50	15	98.53	3	3	0.85	98.78	98.74	
50	20	98.07	5	1	0.65, 0.85	98.77		
			5	3	0.85	98.76		
			5	5	0.65	98.78	98.76	
			8	1	0.85	98.78	98.76	98.76
			8	4	0.85	98.77		
			8	8	0.85	98.77		
65	5	98.77	3	1	0.75	98.78	98.73	
65	10	98.72	3	2	0.65	98.77		
65	15	98.41	3	3	0.65	98.78	98.73	
65	20	97.64	5	1	0.75	98.76		
			5	3	0.75, 0.85	98.74		
			5	5	0.75	98.77		
			8	1	0.75	98.76		
			8	4	0.65	98.77		
			8	8	0.65	98.80	98.77	98.78
90	5	98.72	3	1	0.85	98.76		
90	10	98.66	3	2	0.65	98.76		
90	15	98.54	3	3	0.85	98.77		
90	20	97.54	5	1	0.75	98.77		
			5	3	0.75	98.76		
			5	5	0.85	98.77	98.75	
			8	1	0.75	98.76		
			8	4	0.75	98.79	98.60	
			8	8	0.85	98.80	98.77	98.77

APPENDIX C: SELECTED SAMPLES OF SIMULATION DATA TABLES

Table 4: LatePeak Near – Data from Near factor group simulations

LatePeak Near										
Start Hour	Stop Hour	Min Go Length	Near	Arrival Count	Exit Count	Exit Per Cent	Avg Ticks on Grid	Avg Idle	Queue Count	Exit Per Cent Rank
15	18	20	5	46,514	45,910	98.70	225	794	0	8
15	18	20	10	46,292	45,655	98.62	242	1,183	0	13
15	18	20	15	46,091	45,419	98.54	267	1,616	0	16
15	18	20	20	46,006	45,201	98.25	300	2,298	0	22
15	18	35	5	46,620	46,041	98.76	222	715	0	3
15	18	35	10	46,371	45,761	98.68	237	1,046	0	9
15	18	35	15	46,252	45,594	98.58	265	1,642	0	15
15	18	35	20	46,022	45,147	98.10	317	2,755	0	23
15	18	50	5	46,632	46,055	98.76	221	718	0	2
15	18	50	10	46,358	45,734	98.65	239	1,123	0	11
15	18	50	15	46,121	45,443	98.53	276	1,907	0	18
15	18	50	20	45,869	44,983	98.07	314	2,679	0	24
15	18	65	5	46,614	46,041	98.77	222	725	0	1
15	18	65	10	46,393	45,798	98.72	239	1,114	0	6
15	18	65	15	46,045	45,314	98.41	289	2,238	0	19
15	18	65	20	45,829	44,746	97.64	345	3,424	0	25
15	18	90	5	46,624	46,029	98.72	221	721	0	5
15	18	90	10	46,371	45,750	98.66	243	1,232	0	10
15	18	90	15	45,983	45,310	98.54	302	2,535	0	17
15	18	90	20	45,676	44,552	97.54	366	4,014	0	26
15	18	105	5	46,610	46,026	98.75	219	676	0	4
15	18	105	10	46,254	45,620	98.63	246	1,299	0	12
15	18	105	15	45,890	45,103	98.29	318	2,861	0	21
15	18	105	20	45,588	44,433	97.47	383	4,269	0	27
15	18	120	5	46,609	46,006	98.71	221	712	0	7
15	18	120	10	46,238	45,589	98.60	254	1,509	0	14
15	18	120	15	45,898	45,118	98.30	306	2,600	0	20
15	18	120	20	45,328	43,991	97.05	417	5,110	0	28

Table 5: EarlyPeak, Variable Min Go, Initial 50 – Data from VMGF group simulations, initial min go of 50

EarlyPeak, Variable Min Go												
Initial 50												
Start Hour	Stop Hour	Min Go Length	Near	Min Go Incre.	Min Go Decre.	Min Go select factor	Arrival Count	Exit Count	Exit Per Cent	Queue Count	Total Caution Cycle Count	High Exit Per Cent
5	9	50	5	3	1	0.65	52,523	51,696	98.43	0	176,559	98.53
5	9	50	5	3	1	0.65	52,528	51,750	98.52	0	174,258	
5	9	50	5	3	1	0.65	52,225	51,425	98.47	302	174,662	
5	9	50	5	3	1	0.75	52,430	51,654	98.52	88	175,370	
5	9	50	5	3	1	0.75	52,284	51,484	98.47	239	174,933	
5	9	50	5	3	1	0.75	52,329	51,522	98.46	199	174,186	
5	9	50	5	3	1	0.85	52,427	51,654	98.53	108	177,355	
5	9	50	5	3	1	0.85	52,528	51,746	98.51	0	177,951	
5	9	50	5	3	1	0.85	52,527	51,747	98.52	0	177,594	
5	9	50	5	3	2	0.65	52,526	51,746	98.52	0	177,046	98.55
5	9	50	5	3	2	0.65	52,437	51,645	98.49	88	174,144	
5	9	50	5	3	2	0.65	52,405	51,646	98.55	120	173,529	
5	9	50	5	3	2	0.75	52,521	51,719	98.47	0	178,940	
5	9	50	5	3	2	0.75	52,523	51,737	98.50	0	175,596	
5	9	50	5	3	2	0.75	52,523	51,721	98.47	0	173,544	
5	9	50	5	3	2	0.85	52,527	51,752	98.52	0	180,517	
5	9	50	5	3	2	0.85	52,526	51,754	98.53	0	177,247	
5	9	50	5	3	2	0.85	52,532	51,738	98.49	0	177,884	
5	9	50	5	3	3	0.65	52,524	51,751	98.53	0	176,683	98.53
5	9	50	5	3	3	0.65	52,520	51,746	98.53	0	177,439	
5	9	50	5	3	3	0.65	52,524	51,743	98.51	0	178,347	
5	9	50	5	3	3	0.75	52,526	51,753	98.53	0	177,400	
5	9	50	5	3	3	0.75	52,523	51,726	98.48	0	179,056	
5	9	50	5	3	3	0.75	52,520	51,732	98.50	0	178,617	
5	9	50	5	3	3	0.85	52,523	51,733	98.50	0	179,409	
5	9	50	5	3	3	0.85	52,521	51,744	98.52	0	179,831	
5	9	50	5	3	3	0.85	52,519	51,735	98.51	0	178,417	

Table 6: LatePeak, Variable Min Go, Random Initial – Data from VMGR group simulations, random starting min go

LatePeak, Variable Min Go													
Random Starting													
Start Hour	Stop Hour	Min Go Length	Near	Min Go Incre.	Min Go Decre.	Min Go select factor	Min Go max factor	Arrival Count	Exit Count	Exit Per Cent	Queue Count	Total Caution Cycle Count	High Exit Per Cent
15	18	50	5	3	3	0.85	1.25	46,062	44,675	96.99	0	203,318	98.74
15	18	50	5	3	3	0.85	1.25	46,006	44,532	96.80	0	201,745	
15	18	50	5	3	3	0.85	1.25	46,000	44,577	96.91	0	202,995	
15	18	50	5	3	3	0.85	1.50	46,382	45,794	98.73	0	195,774	
15	18	50	5	3	3	0.85	1.50	46,428	45,845	98.74	0	195,220	
15	18	50	5	3	3	0.85	1.50	46,382	45,794	98.73	0	195,186	
15	18	50	5	3	3	0.85	1.75	46,617	46,029	98.74	0	199,922	
15	18	50	5	3	3	0.85	1.75	46,628	46,024	98.70	0	199,551	
15	18	50	5	3	3	0.85	1.75	46,619	46,027	98.73	0	198,654	
15	18	50	5	5	5	0.65	1.25	46,532	45,957	98.76	0	200,968	98.76
15	18	50	5	5	5	0.65	1.25	46,558	45,981	98.76	0	196,784	
15	18	50	5	5	5	0.65	1.25	46,570	45,975	98.72	0	196,771	
15	18	50	5	5	5	0.65	1.50	46,465	45,866	98.71	0	196,879	
15	18	50	5	5	5	0.65	1.50	46,417	45,831	98.74	0	196,340	
15	18	50	5	5	5	0.65	1.50	46,463	45,858	98.70	0	194,645	
15	18	50	5	5	5	0.65	1.75	46,632	46,025	98.70	0	196,328	
15	18	50	5	5	5	0.65	1.75	46,687	46,096	98.73	0	195,002	
15	18	50	5	5	5	0.65	1.75	46,626	46,015	98.69	0	194,984	
15	18	50	5	8	1	0.85	1.25	46,345	45,742	98.70	0	204,629	98.76
15	18	50	5	8	1	0.85	1.25	46,367	45,770	98.71	0	204,989	
15	18	50	5	8	1	0.85	1.25	46,303	45,705	98.71	0	204,694	
15	18	50	5	8	1	0.85	1.50	44,235	43,079	97.39	1198	174,066	
15	18	50	5	8	1	0.85	1.50	42,844	40,214	93.86	2432	166,916	
15	18	50	5	8	1	0.85	1.50	42,777	39,537	92.43	2609	164,236	
15	18	50	5	8	1	0.85	1.75	46,420	45,843	98.76	0	193,748	
15	18	50	5	8	1	0.85	1.75	46,415	45,807	98.69	0	193,346	
15	18	50	5	8	1	0.85	1.75	46,382	45,798	98.74	0	192,441	

Table 6 (Continued)

LatePeak, Variable Min Go													
Random Starting													
Start Hour	Stop Hour	Min Go Length	Near	Min Go Incre.	Min Go Decre.	Min Go select factor	Min Go max factor	Arrival Count	Exit Count	Exit Per Cent	Queue Count	Total Caution Cycle Count	High Exit Per Cent
15	18	65	5	3	1	0.75	1.25	44,562	43,013	96.52	816	170,862	98.73
15	18	65	5	3	1	0.75	1.25	44,884	44,264	98.62	610	175,760	
15	18	65	5	3	1	0.75	1.25	45,018	44,324	98.46	464	174,842	
15	18	65	5	3	1	0.75	1.50	44,741	43,236	96.64	846	177,361	
15	18	65	5	3	1	0.75	1.50	44,097	42,227	95.76	1217	167,123	
15	18	65	5	3	1	0.75	1.50	43,848	42,423	96.75	1493	172,726	
15	18	65	5	3	1	0.75	1.75	46,546	45,952	98.72	0	188,128	
15	18	65	5	3	1	0.75	1.75	46,519	45,930	98.73	0	187,552	
15	18	65	5	3	1	0.75	1.75	46,539	45,936	98.70	0	187,517	
15	18	65	5	3	3	0.65	1.25	46,439	45,847	98.73	0	195,688	98.73
15	18	65	5	3	3	0.65	1.25	46,413	45,801	98.68	0	195,054	
15	18	65	5	3	3	0.65	1.25	46,399	45,792	98.69	0	195,729	
15	18	65	5	3	3	0.65	1.50	46,409	45,807	98.70	0	186,935	
15	18	65	5	3	3	0.65	1.50	46,360	45,750	98.68	0	186,832	
15	18	65	5	3	3	0.65	1.50	46,412	45,804	98.69	0	185,577	
15	18	65	5	3	3	0.65	1.75	46,182	45,595	98.73	0	191,853	
15	18	65	5	3	3	0.65	1.75	46,182	45,576	98.69	0	191,645	
15	18	65	5	3	3	0.65	1.75	46,254	45,666	98.73	0	189,760	
15	18	65	5	8	8	0.65	1.25	46,685	45,923	98.37	0	211,494	98.77
15	18	65	5	8	8	0.65	1.25	46,644	45,884	98.37	0	193,946	
15	18	65	5	8	8	0.65	1.25	46,631	45,872	98.37	0	191,718	
15	18	65	5	8	8	0.65	1.50	46,660	46,059	98.71	0	193,986	
15	18	65	5	8	8	0.65	1.50	46,648	46,076	98.77	0	193,706	
15	18	65	5	8	8	0.65	1.50	46,705	46,114	98.73	0	192,160	
15	18	65	5	8	8	0.65	1.75	46,444	45,867	98.76	0	195,335	
15	18	65	5	8	8	0.65	1.75	46,474	45,890	98.74	0	195,066	
15	18	65	5	8	8	0.65	1.75	46,492	45,906	98.74	0	194,863	

Table 6 (Continued)

LatePeak, Variable Min Go													
Random Starting													
Start Hour	Stop Hour	Min Go Length	Near	Min Go Incre.	Min Go Decre.	Min Go select factor	Min Go max factor	Arrival Count	Exit Count	Exit Per Cent	Queue Count	Total Caution Cycle Count	High Exit Per Cent
15	18	90	5	5	5	0.85	1.25	46,688	46,096	98.73	0	191,898	98.75
15	18	90	5	5	5	0.85	1.25	46,668	46,071	98.72	0	191,113	
15	18	90	5	5	5	0.85	1.25	46,670	46,071	98.72	0	192,342	
15	18	90	5	5	5	0.85	1.50	46,623	46,041	98.75	0	190,239	
15	18	90	5	5	5	0.85	1.50	46,626	46,038	98.74	0	189,990	
15	18	90	5	5	5	0.85	1.50	46,632	46,029	98.71	0	191,099	
15	18	90	5	5	5	0.85	1.75	46,574	45,974	98.71	0	186,923	
15	18	90	5	5	5	0.85	1.75	46,581	45,996	98.74	0	188,659	
15	18	90	5	5	5	0.85	1.75	46,584	45,985	98.71	0	188,187	
15	18	90	5	8	4	0.75	1.25	46,361	45,459	98.05	0	190,113	98.60
15	18	90	5	8	4	0.75	1.25	46,422	45,551	98.12	0	189,207	
15	18	90	5	8	4	0.75	1.25	46,366	45,484	98.10	0	190,003	
15	18	90	5	8	4	0.75	1.50	46,097	45,105	97.85	375	195,788	
15	18	90	5	8	4	0.75	1.50	45,583	44,753	98.18	794	184,920	
15	18	90	5	8	4	0.75	1.50	45,574	44,738	98.17	804	183,598	
15	18	90	5	8	4	0.75	1.75	46,684	46,032	98.60	0	185,981	
15	18	90	5	8	4	0.75	1.75	46,654	46,000	98.60	0	186,023	
15	18	90	5	8	4	0.75	1.75	46,641	45,978	98.58	0	187,867	
15	18	90	5	8	8	0.85	1.25	46,643	46,060	98.75	0	186,495	98.77
15	18	90	5	8	8	0.85	1.25	46,601	46,026	98.77	0	189,181	
15	18	90	5	8	8	0.85	1.25	46,647	46,053	98.73	0	188,919	
15	18	90	5	8	8	0.85	1.50	46,574	45,971	98.71	0	197,071	
15	18	90	5	8	8	0.85	1.50	46,547	45,969	98.76	0	196,914	
15	18	90	5	8	8	0.85	1.50	46,566	45,982	98.75	0	197,196	
15	18	90	5	8	8	0.85	1.75	46,337	45,733	98.70	0	189,513	
15	18	90	5	8	8	0.85	1.75	46,331	45,734	98.71	0	188,832	
15	18	90	5	8	8	0.85	1.75	46,316	45,680	98.63	0	188,893	