ARTIFICIAL IMMUNE SYSTEM BASED URBAN TRAFFIC CONTROL

A Thesis

by

PALLAV NEGI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2006

Major Subject: Mechanical Engineering

ARTIFICIAL IMMUNE SYSTEM BASED URBAN TRAFFIC CONTROL

A Thesis

by

PALLAV NEGI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Reza Langari
Committee Members,	Yoonsuck Choe
	Charles Culp
Head of Department,	Dennis L. O'Neal

May 2006

Major Subject: Mechanical Engineering

ABSTRACT

Artificial Immune System Based Urban Traffic Control. (May 2006)

Pallav Negi, B.E., University of Delhi

Chair of Advisory Committee: Dr. Reza Langari

Borrowing ideas from natural immunity, Artificial Immune Systems (AIS) offer a novel approach to solving many diagnosis, optimization and control problems. In the course of this research this paradigm was applied to the problem of optimizing urban traffic. The traffic was micro-simulated with each car on a two junction road system modeled individually. The cars themselves were programmed with 'personalities' to better simulate real traffic. A novel AIS was developed to detect, predict, and control anomalous traffic conditions. It was also used to optimize the flow of traffic through the road network. Benchmarking was performed against the well accepted TRANSYT traffic control system. Though the TRANSYT system performed better initially, the AIS control showed marked improvement over time as it adapted better to changing traffic conditions. This change was expected as TRANSYT is optimized for specific initial conditions unlike the AIS system which adapts to changes.

DEDICATION

To my parents and to my sister.

You're something else.

ACKNOWLEDGEMENTS

I would like to give special thanks to my advisor, Dr. Reza Langari, for giving me an opportunity to work on this thesis project. I could not have completed this work without the insight, knowledge and above all, the patience, with which he guided me. Dr. Langari has been an inspiring teacher and a great advisor.

I am very grateful to Dr. Kevin Balke who helped me get started in the field of Urban Traffic Control; his knowledge of the field was invaluable to this work.

NOMENCLATURE

AIS	Artificial Immune System
Ab	Antibody
Ag	Antigen
CFP	Cyclic Flow Profile
GA	Genetic Algorithm
HIL	Hardware in Loops
NN	Neural Network
PI	Performance Index
UTC	Urban Traffic Control
WBC	White Blood Cell

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

1. INTRODUCTION: THE NEED FOR TRAFFIC OPTIMIZATION

With the increasing size of cities, the people who live there are spending considerable time traveling on urban road networks. Smooth traffic flow has become crucial to the functioning of the modern world. The absence of an organized and timely transportation network lowers productivity and increases dissatisfaction amongst citizens. Incidents of 'road rage' are now very well recorded and are often the result of improper traffic flow.

Unintelligent traffic signals controlled simply by timers are no longer sufficient to handle the complex relationships arising out of the increased road and traffic systems. This has led to a spurt of research on a new breed of intelligent controllers. The first steps in this direction were taken when controllers were optimized at a system level and the timers were made flexible. The schedule changed based on the time of the day and on the location. This was followed by new research into traffic-responsive control which takes decisions based on the current traffic conditions at any given time.

An intelligent good traffic network should do more than just minimizing travel times. It should be able to handle random situations like car wrecks and flooded roads. It should also be able to handle semi-random situations like baseball games or parades. Lastly, it should be able to account for the flow of public transportation (buses, trams, trains, etc.) and emergency vehicles (ambulances, fire trucks etc.).

Artificial Immune Systems offer the twin advantages of parallel processing along with extreme adaptability. They supplement the pattern recognition ability of Neural Networks with the optimization and adaptability of Genetic Algorithms. Unlike Genetic Algorithms, there is no fixed model of population, evaluation and reproduction.

This thesis follows the style of *IEEE Transactions on Vehicular Technology*.

Likewise, AIS antibodies do not show the order and inter-dependence of Neural Network nodes.

The natural immune system provides a remarkable model for large scale distributed control and parallel processing. One of the biggest advantages offered by this paradigm is extreme scalability. There is only minimal communication required between individual members of the system. As such, the information exchange is independent of the size of the system. Hence, the same model can be used whether we have ten nodes, or a million, using only linear scaling. Each node takes decisions on its own, based on its genetic make up. It uses inputs, direct or indirect, from other parts of the immune system, but uses extremely localized decision making. In fact, the total destruction of one type of nodes will not impair the immune system as such.

In addition, the immune system also possesses excellent noise rejection properties. It is able to filter out and respond to specific threats from amongst the multitude of input it is receives. It can even recognize pathogens that have mutated beyond their original form, or new pathogens that are similar to previously encountered ones.

Lastly, the immune system is a very good classifier. It is able to recognize events or anomalies and classify them into pre-recognized groups. This adds an extra level of abstraction making human intervention easier. Also, this makes it easier for the system to predict traffic problems before they happen.

2. LITERATURE REVIEW

This section describes other approaches used in the control of traffic systems, in particular, of urban traffic systems. Urban traffic provides a unique set of circumstances because of the complexity of the road networks, the amount of traffic flowing, and the complex interactions that arise out of the varied driver personalities and driving conditions. The strategies can be broadly divided into fixed time control strategies and adaptive control strategies. Fixed time control strategies are optimized for certain traffic conditions and use the same timing sequence of traffic signal control for all conditions. In many cases, these may be semi-adaptive which is to say, they may have different phase and timing sequences for different times of the day. For example, there may be longer durations of the green signal along one road at night.

2.1 The TRANSYT Control Model

This fixed time method for traffic control was developed by Robertson in 1969. *[1]* It has since, been modified and enhanced by the work of several people. In its most basic form however, it is a system that starts off with a given phase sequence and phase timings. It then attempts to change them to optimize the traffic flow under given conditions. The changes are made using simple optimization algorithms like hill climbing which attempts to introduce small changes in the parameters and checks for a better performance index at each step. The important aspect is that TRANSYT must have a simulation model that can be run with different control variable values but the same traffic conditions multiple times. The model is first run with the initial conditions. The system then introduces small changes in the timing sequence and checks to see if the performance index improves. It then changes the parameters in another direction from the original and checks the performance index again. The set of changes that gave the best result are implemented and the search starts again from that point. The system repeats this search process until the performance index stops improving. This fixed time sequence is then implemented in the real world as shown in Figure 1.

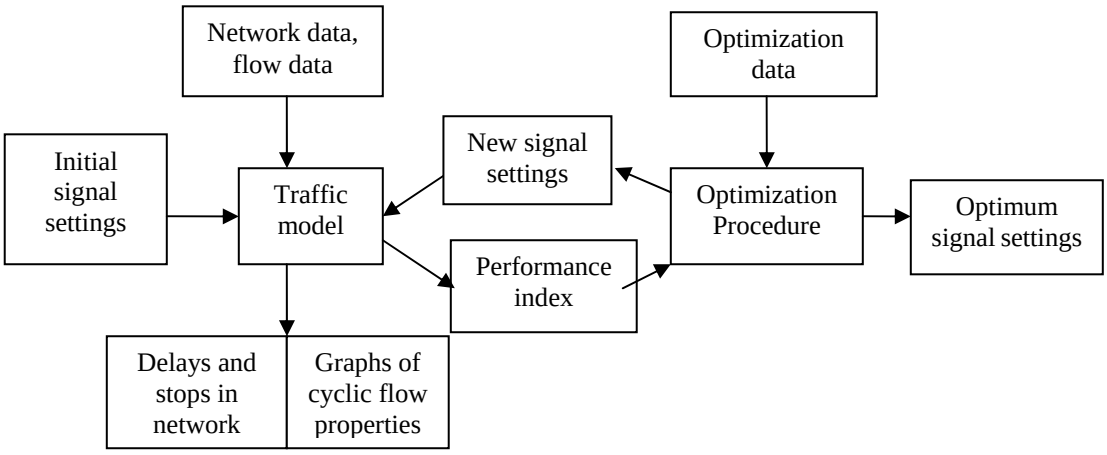

Figure 1: TRANSYT program logic flow. *[2]*

The performance criteria usually optimized for TRANSYT is the queue length or the waiting period for each vehicle. Waiting period or travel time is important in that smaller waiting times mean smaller driver dissatisfaction. One thing to be remembered about TRANSYT is that it has been optimized only for a given set of conditions. It may not perform as well during other conditions of traffic flow. However, real world experience has shown that even the first implementation of TRANSYT indicated savings of 16% in travel time as compared to the previous timing control in place. *[2]*

This research used a simple TRANSYT implementation as a benchmarking system against which to compare the performance of the AIS based traffic control. The AIS based control performed significantly better after the simulations had run for some time. This was as per expectation as TRANSYT is optimized for given conditions while AIS learns over time to improve its performance. The decision to use TRANSYT was based on two factors – the fact that TRANSYT is a widely used system (both for benchmarking, and for controlling traffic), and for ease of programming and standardization.

2.2 The MAXBAND Control Model

This is another fixed time control strategy. It derives its name from the basic optimization principle behind this approach – maximize bandwidth. In this case, bandwidth refers to the amount of traffic that can flow through a given series of green signals within a specified range of speeds, without having to stop at any red signal. *[3]*

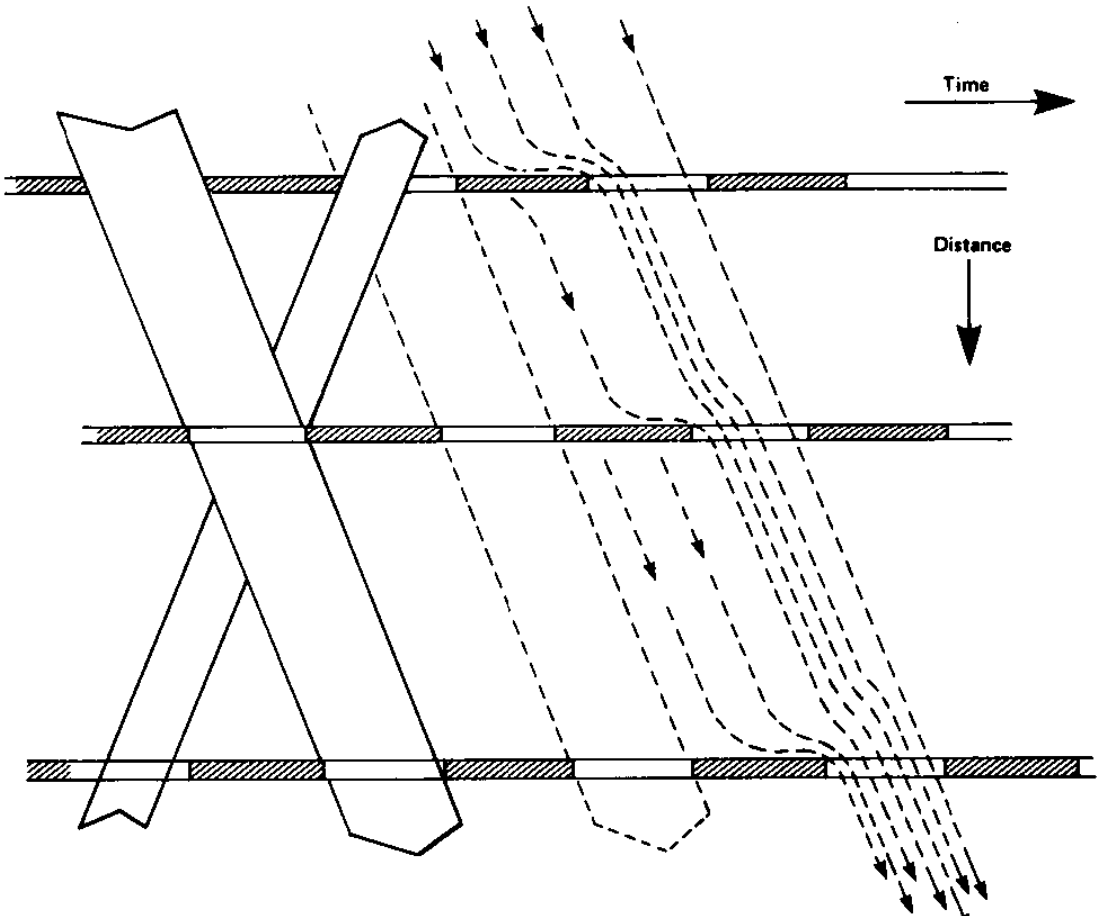

Figure 2: Time-distance diagram used in MAXBAND. The shaded areas represent red signals while the blank areas along the horizontal axis are green signals. The diagonal lines show the bandwidth of traffic that can pass through safely. *[4]*

The principle can be visualized using the time-distance diagrams which represent time along one axis and distance along the other. They can be used to see how many vehicles will be able to pass through subsequent traffic signals unhindered.

In Figure 2, the solid diagonal bands represent ideal traffic flow and bandwidth. However, as with any real system, there is inertia in play here, and the actual traffic flow is shown by the dotted arrows. Some vehicles that should ideally have been able to pass the junction are unable to do so because of the lag in starting up for the cars that were ahead of them. Though this limits the accuracy of T-D diagrams, they are still very useful as visualization and give good approximations.

2.3 The SCOOT Control Model

The SCOOT traffic optimization system was developed sharing some of the basic principles from TRANSYT. In most implementations they share the performance index, as well as the traffic model used to simulate and optimize traffic flow. Queue estimation is also done using similar calculations in both models. Queue refers to the amount of standing traffic at a junction due to a stop signal.

Since the term bandwidth has little meaning in urban traffic situations due to the amount of traffic and complexity of the scenario, the performance index for SCOOTS is not based on bandwidth. The basic performance index is the queue length in each lane in addition to the number of vehicle stops that have been made. A general recommendation is that one vehicle stop is given the same weighting as twenty seconds of delay. *[4]*

SCOOT and TRANSYT also share the type of traffic simulation they perform. The simulation model in SCOOT is based on ‘Cyclic Flow Profiles’ as shown in Figure 3. The idea behind this is that traffic follows periodic increases and decreases in flow and this can be used to predict and model traffic flow. Unlike TRANSYT however, SCOOT offers online update in the signal timing. It maintains a model of the queue estimations and based on a principle of rolling horizons keeps predicting and optimizing the flow of traffic. A few seconds before every phase change, the SCOOT system attempts to check if extending the current phase by a fixed amount (traditionally 4 seconds) will improve the performance index or not. If it will, the system extends the current phase. This is

continued until a predetermined maximum phase time is reached, at which time the system changes to the next phase.

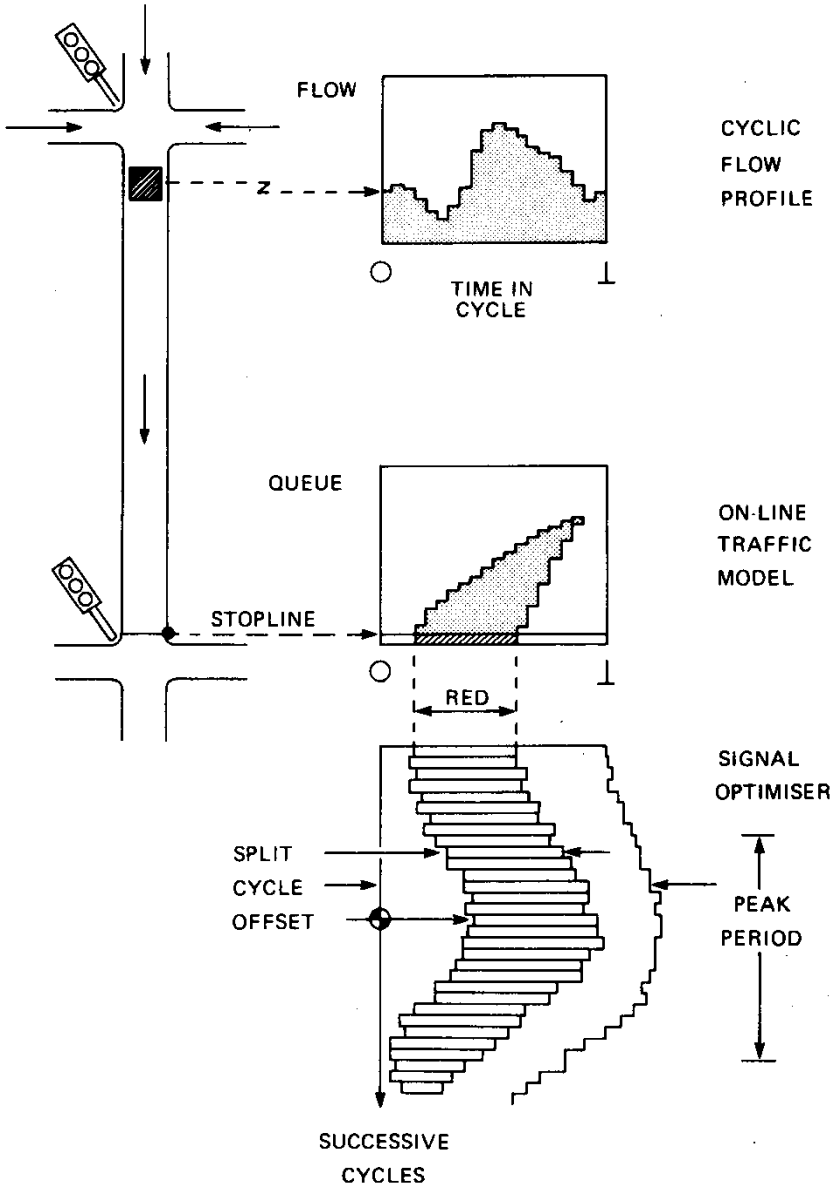

Figure 3: Key features to the SCOOT system. The Cyclic Flow Profile is used to predict traffic queues and the signal optimizer changes the split, cycle and offset to give the best performance index. *[4]*

The concept of split, cycle and offset as used in the simple implementation of SCOOT and TRANSYT are explained in Figure 4. Cycle time refers to the complete cycle of one signal phase. The offset refers to the duration of time after which the green signal is activated. The split is the duration of the green signal. This is a simplistic model that ignores the duration during which there must be a yellow or amber signal. However, for the purpose of this study, this representation was sufficient. For benchmarking, it must be kept in mind that the definition of the split, offset and cycle time was the same for both algorithms.

In a system like SCOOT, the placement of sensors is of specific interest. SCOOT requires sensors to be placed some way from the upstream end of the traffic link as this enables better queue estimation in this approach. In this case, upstream refers to the end of the link road where traffic enters, while downstream refers to the end where the traffic exits the link road. The properties measured are flow and occupancy, which are similar to density measurements. The system is run continuously from a central control computer and updates traffic signals within its domain. The update decisions are usually made at four second time steps. Similarly to TRANSYT, SCOOT's performance deteriorates under saturated traffic conditions.

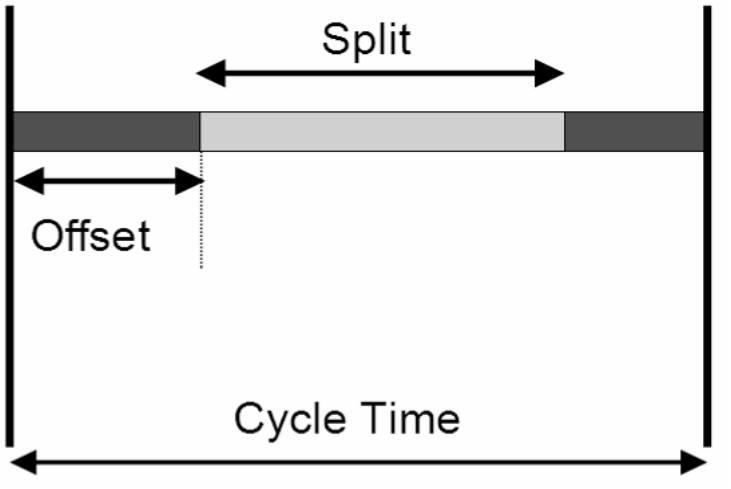

Figure 4: The concept of split, offset and cycle time.

2.4 The RHODES Control Model

RHODES is a real time adaptive strategy for managing urban traffic. It is a multi-tier system that utilizes several different algorithms to optimize traffic flow at several levels. Unlike the approaches discussed earlier, RHODES works on the principle of ‘phases’ and as such, this approach is called phase based as opposed to the previous parametric control approaches.

The idea behind phases is explained in Figure 5. One phase is said to be a unique set of directional traffic flows that occur concurrently.

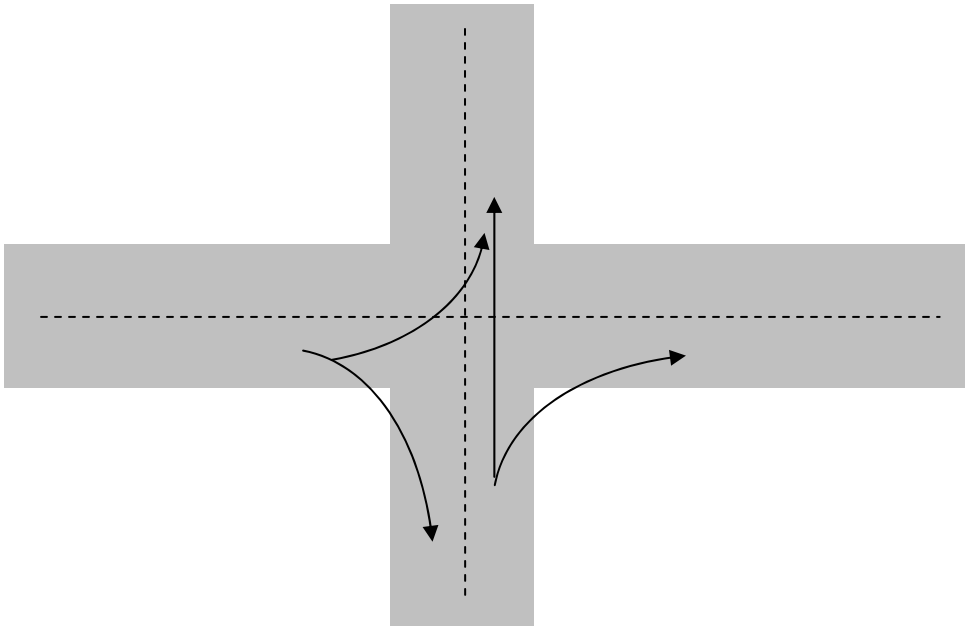

Figure 5: The concept of traffic signal phases.

As discussed earlier, RHODES is a multi-layered system. It attempts to optimize traffic in three tiers. At the lowest level is the Intersection Control which optimizes at the level of individual vehicles. The middle tier consists of Network Flow Control which optimizes on the basis of ‘green times’, or the concept of demand and phase. At the

highest level is the Network load estimator and predictor which works with the slow changing characteristics in a traffic control environment.

The Network Flow Control uses REALBAND *[5]* to optimize traffic using the concept of 'platoons' as shown in Figure 6. A platoon is a large number of vehicles moving together which for the purpose of optimization are treated as one body. It is attempted to keep the platoon moving together unbroken in order to improve driver experience. Conflict resolution is done using the T-D diagram. This level also assigns the maximum 'green time' to a phase and specifies the constraints for the intersection level controls. REALBAND uses a T-D diagram much like MAXBAND does. The optimization algorithm searches through the options tree to find the optimal decision at each junction.

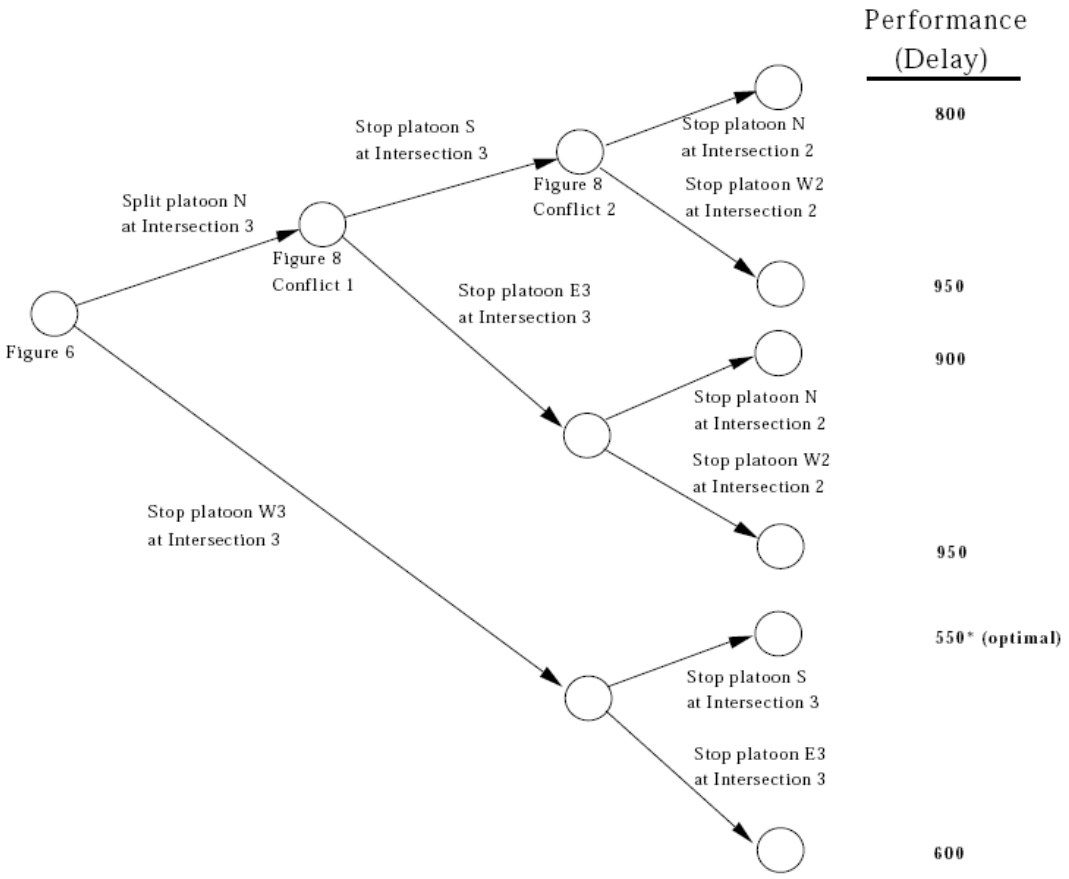

Figure 6: REALBAND decision tree. *[6]*

The Intersection Control layer uses the PREDICT *[7]* algorithm to simulate traffic flow and is assisted by traffic sensors placed in upstream locations much as in the SCOOT method for traffic control. An optimizer (over local traffic conditions) chooses the best amount of green time to assign to each phase. The phase timing is constrained by the limits imposed by the higher levels. This ensures that local optimization does not result in problems when looking at the system globally.

2.5 Other Recent Research in the Field of UTC

This section describes some of the recent research specifically in the field of UTC.

In *[8]* the authors present a blend of a back-propagation neural network with a fuzzy expert system to optimize flow through urban roads. They compare the system developed against a simple ANN and a fuzzy expert system. The system developed by them shows better performance at lower implementation costs per their measurement criteria. However they fail to compare its performance to a standardized system.

In *[9]* the authors develop a UTC system based on Box's algorithm. They develop the method based on the assumption that they are able to achieve real time input from a camera and able to parse the visual information to extract the number of queued, incoming, outgoing, and left turning vehicles in real time. No explanation is given on how the real time processing is assumed to be completed.

In *[10]* a hierarchical system for road traffic management is explained. The system attempts to improve traffic flow by utilizing green waves which are already implemented in many traffic systems. Green waves refer to the principle that cars traveling within the green wave meet only green signals all along their path due to optimal timing of the signals.

3. THE NATURAL IMMUNE SYSTEM

The natural immune system fights protects the organism at two levels often called *primary* and *secondary* immunity. The first line of defense is general purpose and fast acting, while the second is adaptive and takes action specific to the invading pathogens.

The first line of defense consists of *general purpose defense mechanisms* which act as deterrents to all pathogens. This includes the skin and mucous membranes which act as a physical barrier along with secretions like saliva and tears which contain immunoglobulins. These are also the link between the first and the second levels of the immune system. The *secondary response is adaptable* and geared towards specific pathogens. Most work in replication of immune systems deals with the secondary response, mainly because the primary response systems are more or less akin to the systems already in place. For example, a voltage regulator that cuts off input above a certain voltage is essentially a primary response immune system. It acts like a physical, general purpose barrier to the antigens, which in this case would be a voltage spike. Figure 7 shows a block diagram representation of this scheme.

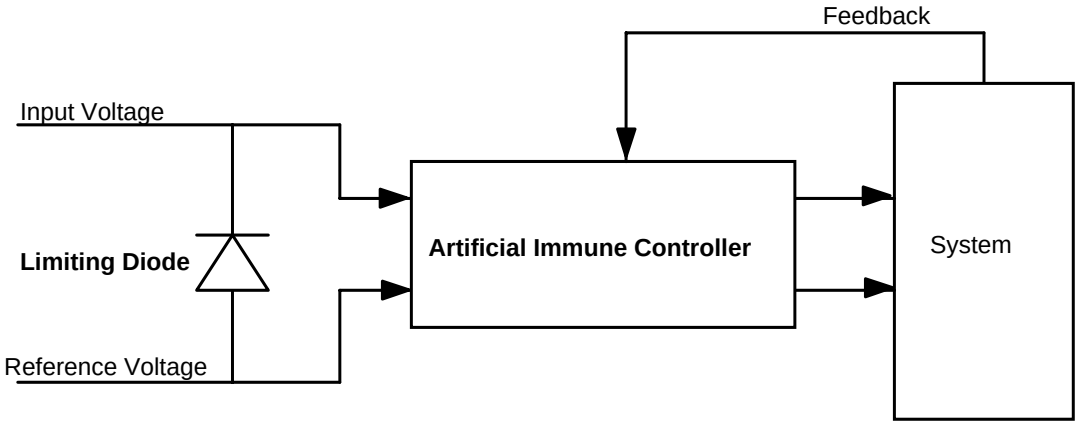

Figure 7: Example of primary and secondary immune protection as applied to a physical system. The limiting diode acts as the general purpose primary response while the AI controller is the second layer.

The secondary line of defense in the natural immune system is more specific in its response and is mediated by white blood cells (WBC's). Of these WBC's, the *lymphocytes* are responsible for both cell mediated immunity and humoral immunity. Lymphocytes are further sub classified as B and T type cells. B-cells are responsible for humoral immunity which leads to the production of antibodies (Ab) in response to an antigen (which can be anything from a microbe to somebody else's blood cell). T-cells, on the other hand, are responsible for cell mediated immunity, and also for inducing the B-cells to produce antibodies.

Cell mediated immunity by T-cells uses a special type of immune cells called 'killer cells'. These cells are similar to T-cells in origin and attack the invading pathogen without the help of antibodies. This type of immunity is known to be most effective against viral attacks, as opposed to the B-cell mediated immunity. This type of immune response also causes cells to release special chemicals like cytokines that activate other layers of immune defense like the B-cells which are activated by the T-cells when they detect an infection or antigen.

B-cells have distinctive molecular structures which form the basis of the pathogen recognition process. Y-shaped antibodies are formed on the surface of these cells. Antibodies recognize molecular patterns in the pathogen cells and are hence able to alert the immune system when a pathogen is encountered. Additionally, antibodies once produced stay in the system. This enables a faster response when identical or similar pathogens are encountered again. On identification of an antigen the reproduction of its corresponding antibodies is sped up in order to fight the infection. Furthermore, to recognize antigens better, the antibodies undergo continuous mutation. This is explained in detail later.

T-cells are responsible for making the B-cells react to the problem. Activated B-cells start reproducing rapidly in order to fight the infection. In addition, they undergo a

process called affinity maturation. This process involves mutation to make the B-cells recognize the Ag faster and better. B-cells that have been simulated mutate in order to increase their affinity towards the invading Ag cells. The rate of mutation is decided by the affinity of the existing cells. Cells with high affinity mutate at a much lower rate than cells that have lower affinity. This ensures that B-cells are able to find and fight the infection much faster next time. *[11]*

Some cells become memory cells. These types of cells have a very long life span and are able to recognize the same infection should it recur in the future. This avoids the time and resources that would otherwise have been required.

The Ab *recognizes* a portion of the Ag called its *epitope*. An *idiotype* is a part on the variable regions of a set of Ab. Each Ab type has a distinct set of idiotypes which can be used to uniquely identify the Ab type. An Ag typically has several different types of *epitopes*, and can be recognized by several different antibodies (Figure 8). The *paratope*, also known as V-region, for *variable region,* is an antibody portion responsible for matching (recognizing) an antigen. It is variable because it can alter its shape to achieve a better match (complementarily) with a given antigen. The strength and specificity of the Ag-Ab interaction is measured by the *affinity* of their match. *[12]*

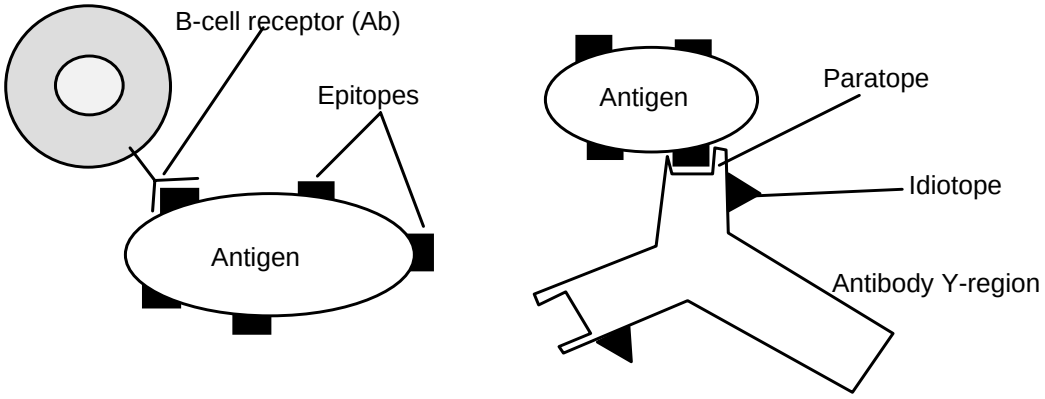

Figure 8: B-cells, Ag, Ab, epitopes, paratopes, and idiotypes. *[12]*

The natural immune system has been studied in far greater detail that is possible to cover in the scope of this thesis. However the brief overview here was aimed at familiarizing the reader with the basic concepts of immunity that were used in the course of this research. When applied to artificial immune systems, the natural immune system provides great opportunity in view of its adaptability and scalability. It combines philosophies from several different artificial intelligence approaches like genetic algorithms, distributed computation and pattern matching, and combines them into one system.

Although the immune system is not perfect in its recognition or control of antigens, it has evolved over millennia to serve as a very good model for distributed intelligence. The different cells in the body have little or no contact with each other, and as such are insignificantly small compared to the entire system. However they manage to detect and control almost all the infections faced by the human body over a lifetime. Moreover, the immune system shows adaptability and is able to fight new infections in addition to remembering infections it fought earlier.

4. ARTIFICIAL IMMUNE SYSTEM CONCEPTS

Artificial Immune Systems (AIS's) borrow metaphors from natural immune systems to solve real world problems. Most AIS models have a few common features. One is the definition of 'self' and 'non-self'. In general, 'non-self' refers to unwanted or abnormal conditions while 'self' refers to normal operating conditions. For example, in the case of traffic systems, smooth flowing traffic would constitute self. A car wreck on the other hand, would be an example of non-self. In some cases, the distinction between the two is not clear. An important part of the traffic control AIS function would be to distinguish between seemingly non-self conditions and true non-self conditions. An example of this is slow moving traffic due to the 'wave' effect in which cars slow down and speed up in waves due to the inertia of vehicles closest to the traffic signals at the time the signal changes from red to green. The system should not mistake this for a potential traffic jam condition where it thinks all vehicles are progressively slowing down to a stop.

The report now describes four models that are currently popular in the design of AIS algorithms. Out of these, Negative Selection is probably the most important and most widely used.

4.1 Negative Selection

In this model, the system generates antibodies that recognize non-self intruders. But creating antibodies exhaustively specific to all such antigens can result in an undesirably high number of antibodies. The number of antibodies can be reduced by eliminating antibodies that recognize *harmless non-self characteristics*, and/or those antigens which would not be encountered in the real world. For example, in a steam plant, certain combinations of pressure and temperature in a system of pipes might be physically impossible to produce. Antibodies which recognize such combinations can safely be discarded without reducing the system's efficacy in the real world.

This model shown in Figure 9 is the most popular in current AIS applications. The general steps to implementing it are as follows. *[13]*

1. Create a set of self-cells based on the system's ideal operating conditions and values.
2. Randomly generate antibodies and try to match them to the self-cells.
3. If the antibody matches the self cell, discard the antibody and generate another. Otherwise, store the antibody in the immune set.
4. Repeat steps 2 and 3 till the desired number of antibodies are obtained, or the preprocessing time exceeds the desired limit.

The steps 1 through 4 described above form the preprocessing stage of the general purpose Negative Selection algorithm and need be performed only once. After this, the system can be put on-line to monitor incoming data in real time by matching the antibody set against it. In case of a match, there is a high probability of an undesirable input. The greater the number of antibodies matching the input, the greater the probability of a non-self condition having been encountered.

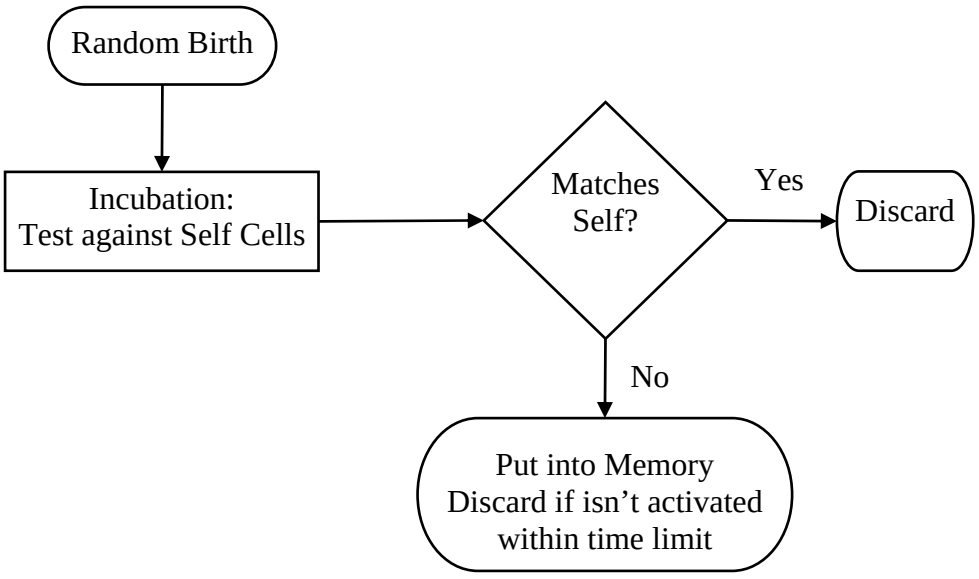

Figure 9: Typical life cycle of an antibody in Negative Selection.

Such an implementation works best if the initial training set of self cells covers all possible ‘good’ values. Its accuracy also increases with the number of antibodies that are generated in the preprocessing stage. Assuming that the initial data set used for training was complete, and a sufficiently large number of antibodies were generated, the Negative Selection algorithm gives the advantage of never reporting a false negative. Figure 10 gives a graphical representation of how the Ab are spread over the self space and recognize non-self entities.

The post processing stage can also include mutation to continuously improve the system’s accuracy and reliability. If one type of non-self data is set is encountered very often, an antibody specific to this antigen could be introduced. Or, the antibodies that match it could be mutated to give more accurate recognition of other similar problem areas. In other words, it errs on the side of caution.

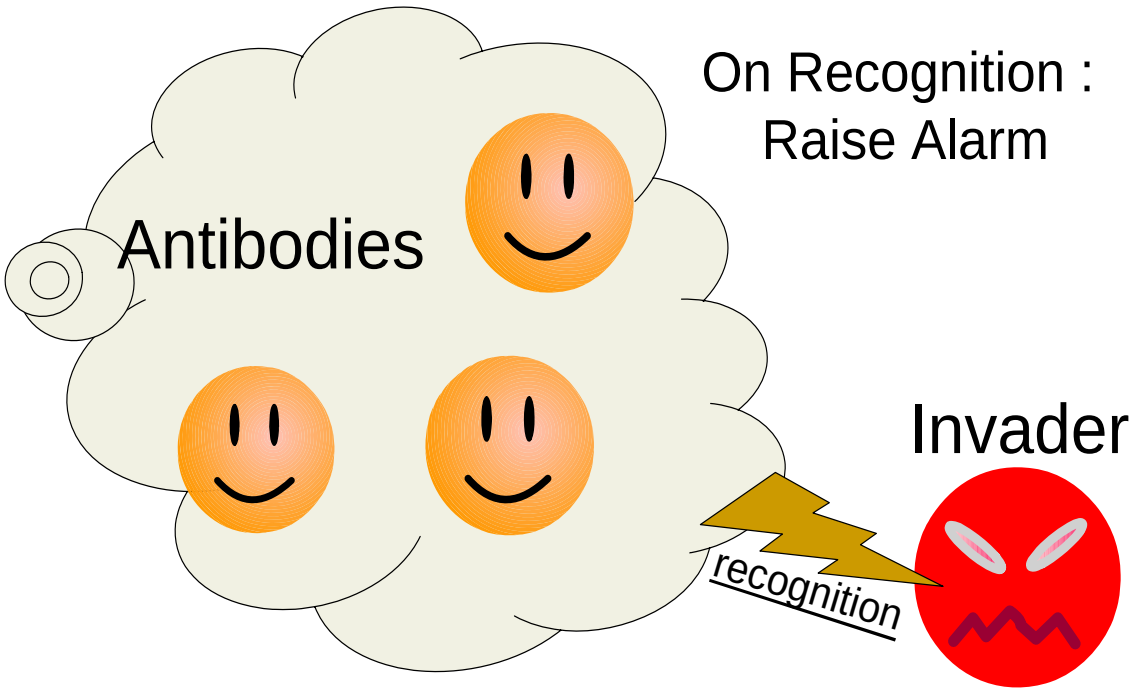

Figure 10: Graphical representation of Ab and Ag in Negative Selection

Negative Selection offers some inherent advantages. For example, it makes it difficult to determine the ‘self’ space by analyzing the antibodies. This might be important in security critical situations. Also, since Negative Selection never reports false positives, it can be used in multi-layer environments. False negatives would likely be removed in subsequent trials in the different layers.

4.2 Positive Selection

This model works complementary to the Negative Selection algorithm in that, the antibodies are trained to recognize *self* instead of *non-self*. The antibodies in this case are generated randomly and selected if they *match* the self-training set. The advantage of this approach is that it will never result in a false positive. The disadvantage is that it will sometimes give false alarms when there is no actual fault in the system. This may in general, be more damaging to the system than the Negative Selection model described earlier.

4.3 Immune Network Model

The immune network theory was first proposed by Jerne *[14]* and proposes a complex system of interaction between Ab-Ab pairs along with Ab-Ag pairs. According to this model, the paratopes are used to recognize idiotopes on both antigens and antibodies. The recognition can result in a positive or a negative response. A positive response would mean that the production rate is increased, while suppression means that the production rate is decreased. Production of one type of antibody can therefore stimulate the production of more such antibodies along with other types of antibodies. In addition, after the infection has been successfully cleaned, the immune network can prevent the uncontrolled production of any one type of antibody. This is useful in preventing an uncontrolled immune response which would be detrimental.

The population variation in such a model depends not only on the birth of new (and sometimes novel) cells, and death of old (or unused) cells, but also on their

concentrations and interactions between them resulting in a complex and interdependent system that is capable of self-learning and recognition.

4.4 Clonal Selection

This theory has come to the forefront in recent years as the more popular explanation of how the immune system in the body actually works. It supposes that cells, with receptors to match antigens, already exist in the body. These cells, when they come into contact with the antigens reproduce according to the following mechanism.

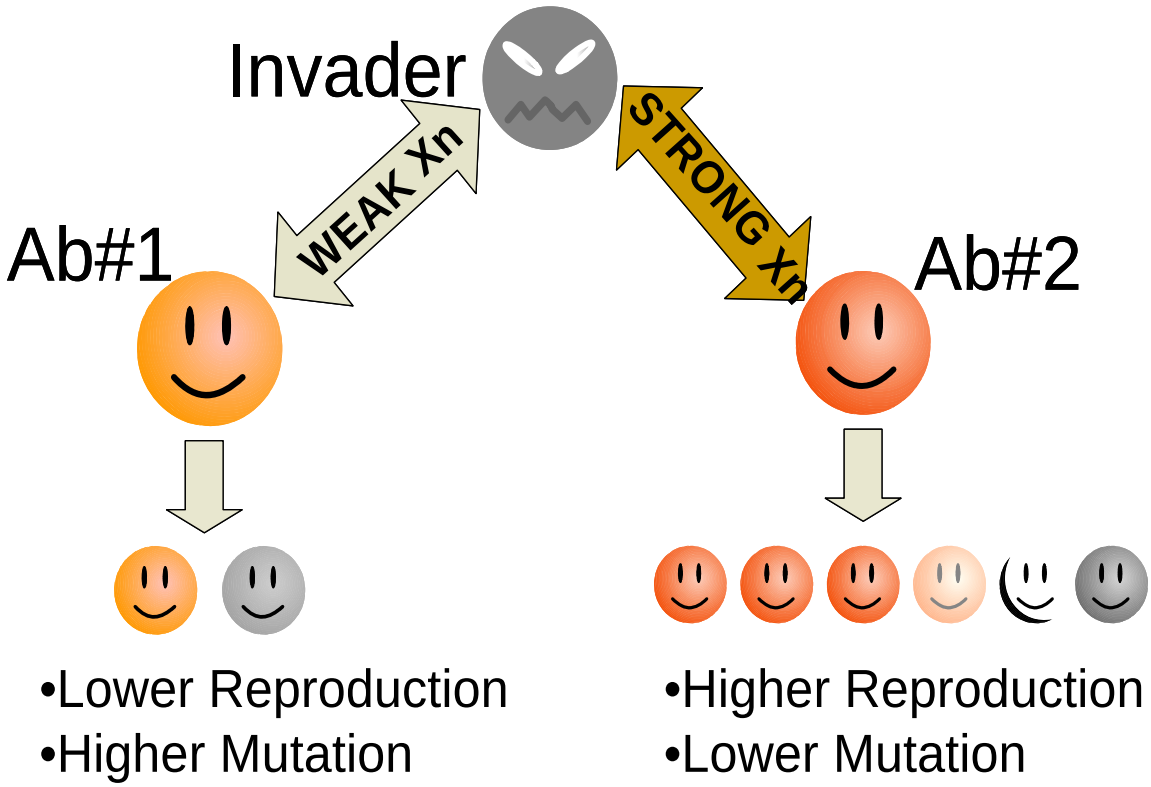

Figure 11: Reproduction via the method of Clonal Selection.

When the Y-cells on the surface of a B-cell recognize an epitope, it is said to be activated. Activated B-cells undergo a process of reproduction (cloning) and differentiate into two different types of immune cells called plasma and memory cells. Plasma cells are responsible for secreting Ab while memory cells are long lived cells with high affinity. Due to the accelerated growth and reproduction, cells that have the greatest affinity for the invading pathogen end up with the greatest concentration. These grow in concentration and affinity (via mutation). The memory cells are responsible for recognizing the antigen in case of a future infection by the same, or similar pathogens. This process of pattern recognition and selection is called Clonal Selection and is similar to natural selection except that it occurs on a much faster time scale. *[12]*

It is believed that cells that show higher affinity show lower rates of mutation whereas cells that have lower affinities show greater rates of mutation (Figure 11). This ensures that the Ab show continuously increasing affinity towards the invading antigen. In addition, it increases the possibility of the Ab being able to recognize new antigens similar, but not the same, as the ones currently encountered.

The approach used in the current research for traffic control used the method of Negative Selection and Clonal Selection. Negative Selection was used for matching the Ab to the antigens, while Clonal Selection was used as the basis for Ab reproduction and propagation.

4.5 Previous Applications of AIS

One of the first applications of Artificial Immune Systems was in the field of computer network security *[15]*,*[13]*. They were able to apply the principles of Negative Selection to detect harmful network traffic using linkwise pairing of computers and the data transaction between them as antigens and antibodies. They also introduce concepts of Negative Selection as a growth mechanism for the Ab along with allowing Ab to grow into memory cells which stay in the system much longer than normal cells would.

In *[16]* Singh et al. simulate a large (numbering in thousands) group of robots and organize them to achieve given targets using Artificial Immune Systems. The robots were shown to collaborate using immune principles. Random Brownian motion was used until an antigen was found. Robots then triggered responses in each other and the response spread to the team much as in cells reacting in an immune environment.

In *[12]*, an artificial immune network model is formed where the concepts of immunity are applied to form a network that classifies and filters large amounts of data. As such, AIS is known to be a very good classifier system and are often used in pattern recognition. Its application in this research will utilize this property of AIS to identify abnormal travel conditions on urban road networks.

[17],*[18]* describe the use of Artificial Immune Systems to the field of sensor diagnostics. In *[17]* AIS were used to diagnose and identify faults in the sensor system for a gas lift well. The sensor readings were fed through an AIS network which identified faults based on a set of empirical rules. Intelligent polling was then used to verify the occurrence of faults in the sensors. In *[18]* an AIS was trained using part of a large amount of sensor data. The remaining sensor data was then used to test the performance of the AIS to verify it being able to identify sensor faults. The AIS outperformed an enhanced artificial neural network in terms of time and gave comparable results in terms of sensitivity.

5. OBJECTIVES

The primary objective of this research was to study the feasibility of using Artificial Immune Systems in the field of Urban Traffic Control. UTC offers a very complex system, with semi-periodic behavior. AIS will be used to predict any abnormalities that are about to arise in the UTC situation and to efficiently control it in a timely manner. The work can be divided into four stages.

The first stage will be the development of a simulation environment that allows the proposed system to be tested. The simulation environment must be suitably accurate in its representation of a real world urban traffic system. To this end, the simulation will be based on micro-simulation where the motion of each vehicle is planned out in real time.

The second stage will be the development of an AIS that is able to detect anomalous situations that occur in the urban traffic simulation. In addition, this system must provide a method to control the environment in normal conditions as well as being adaptive enough to handle abnormal conditions or problems that may arise.

The third stage of the research will be the benchmarking of the AIS control against a known standard UTC algorithm. The benchmarking algorithm used will be TRANSYT. This is a well accepted benchmarking solution. The TRANSYT methodology used in the benchmarking shall be a simple one. It shall not account for the enhancements that were made to the original TRANSYT algorithm over the course of time. Also, the simulation model used for TRANSYT shall be the same as that used for the AIS based UTC in order to maintain the sanctity of the results. This is not expected to unfairly bias the report in favor of one system or the other because everything except for the control algorithms will be maintained same for both tests.

It should be kept in mind that the system is being analyzed for feasibility and the simulation software developed will not be of sufficient accuracy to accurately test the system before deployment in the field. However, the system will in the future be shifted over to hardware in the loop testing with well accepted standard simulation software CORSIM available at the Texas Transportation Institute (TTI) at Texas A&M University.

To this end, the fourth stage of this research will be to begin the process of shifting over the entire AIS control algorithm to a HIL testing facility at TTI. An overview of the available HIL facilities will be presented and preliminary research will be conducted into making the necessary transition. It must be noted that the fourth stage is part of future research and will not be completed for the purpose of this thesis.

6. STAGE I: TRAFFIC SIMULATION SOFTWARE

6.1 Types of Traffic Simulation Models

In general there can be two types of traffic simulation models, namely, macro-simulation and micro-simulation. Macro-simulation uses bulk properties (like density) to model the flow of traffic through a road network. Macro-simulation has been adopted in several earlier approaches because of the low computation requirements and simplicity in programming. Moreover, traffic flow on a system of roads can be approximated as a fluid flowing through a series of pipes, with valves representing the stop and start at traffic junctions. For this reason, the macro-simulation approach is able to give very good results in modeling traffic flow.

Micro-simulation on the other hand, models the motion of each vehicle individually. For a long time this approach was not popular due to the increased computational requirements, especially over larger network simulations. However, with the increase in computation power in today's computers, micro-simulation is gaining ground again. It offers several advantages. For example, individual driver behavior can be modeled in micro-simulation but not in macro-simulation. Also, we can monitor the motion of single vehicles and it is able to represent under-saturated conditions better than macro-simulation is.

6.2 Description of Simulation Used

For the purpose of this study, a micro-simulated model was developed. There were two main factors affecting this decision, namely:

- The system represented was a two junction system and traffic flow would not be too computationally expensive.
- The micro-simulation model allowed for programming of behaviors (aggression levels) into individual drivers in order to better simulate the erratic behavior or real traffic.

The development of the simulation model can be divided into the following stages in chronological order:

1. System Model
2. Behavior Model
3. Graphical Representation

6.2.1 System Model

The basic building blocks for the traffic are the vehicle and driver properties. These are grouped together into one class (agent) called the vehicle. This class describes:

1. Vehicle dimensions: For the sake of simplicity, this was kept constant for all vehicles. A more accurate simulation model would have introduced changes in vehicle dimensions in order to account for real life conditions. However, since the simulation was just to be used in a preliminary analysis, this added complication was avoided. Also, since both the AIS and the benchmarking TRANSYT software would run on the same simulation, this would seemingly not bias the results in favor of one of the other.

2. Destination: A unique number identifying the vehicle/driver's destination. In this case, the number refers simply to the intersection where the vehicle goes off the map, for example the points on figure 12 marked 21 or 24. The destination is important in the decisions made by the driver to change the lane they are traveling in and also in order to decide where to make the turn.

3. Current location: A unique set of numbers identifying the vehicle's location between two intersections, this consists of the beginning intersection and ending intersection. (for example, the vehicle displayed in blue will have a set of numbers like 11-12-80-E representing the starting intersection, ending intersection and 80 units of the path complete in the East direction). The current

location of all vehicles is also stored in a globally accessible array. There are separate arrays that store the location of the vehicles, as well as the speed of the vehicles. This allows for the sensors to be simulated. In order to maintain realism, the sensors were only allowed to measure the traffic flow (average flow speed) and road occupancy which are standard measurement criteria on UTC systems.

4. Speed: Each vehicle class object stores the speed at which it is traveling. It also keeps track of the acceleration. These two values are used to calculate the motion of the vehicle at each time step. The speed and acceleration values change depending on the surroundings of the vehicle.

5. Driver skill/risk behavior: This is be a number representing the amount of allowance a driver will look for before safely being able to make a turn, change lanes etc. This is also called the driver personality or aggressiveness. In most cases, this number is set to zero implying that the driver moves like a standard driver. In a small number of cases (5%) the driver aggressiveness might be increased or decreased by a fixed number (0-3). This represents a percentage aggressiveness in how early or how late the driver changes lanes. It also decides the amount the driver accelerates by.

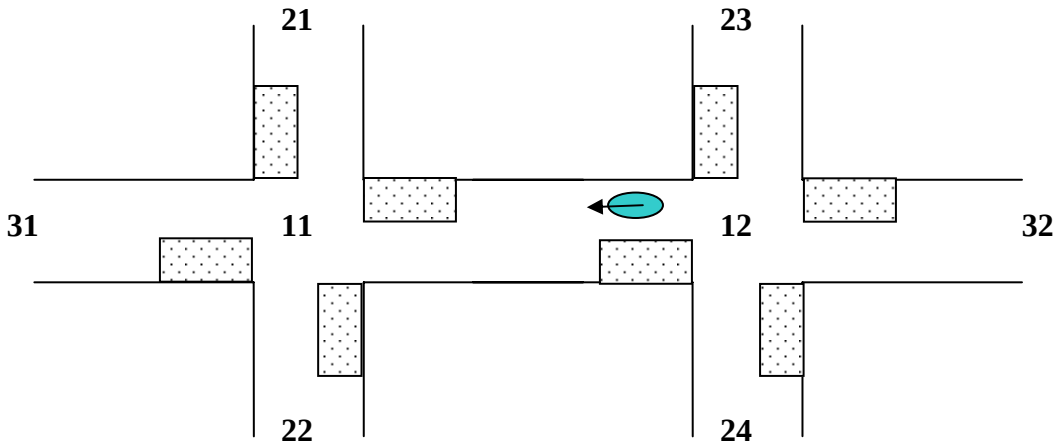

Fig 12: Representation of the two junction road network tested during research. The shaded regions represent the areas covered by sensors and constitute 25% of the longest link length.

Apart from modeling traffic, a map was created to represent the road network. Each node was uniquely indexed with numbers to represent it. The two different types of nodes were:

- Intersections: These are points where the roads meet, and contain traffic signals (for example, numbers 11 and 12 in Figure 12). The traffic signals are used in several ways to decide the simulation behavior. Drivers change their lanes if they are in the long lane to turn based on their distance from the traffic signal. Most drivers start attempting to change their lanes at 50 car lengths from the traffic signal. Some do not, depending on their aggressiveness level. These are also used to decide the placement of the 'sensors' that detect traffic as shown in figure 12.
- Sources/Sinks: These represent places where traffic is introduced into the map, and where it exits (example, nodes 32 and 23). These are also populated with entry/exit 'gravity'. Basically this means how many cars wish to enter/exit at these nodes. By changing these values with time, simulation of office hours, weekdays etc can be achieved.

The last part of the physical environment representation consists of the traffic signals. These are placed at every intersection facing the four cardinal directions. Logic to control the signal is programmed into the controllers. The logic units are programmed as separate agents, the decision of when to turn the signal is going to be part of the AIS algorithm.

Traffic signals are however ‘hard’ coded such that they cannot be made to give conflicting passages. (i.e. potentially dangerous lane combinations being allowed to move at the same time). They are also hard coded with the maximum time that they can allow one phase (or one green signal) to continue. This might seem counter-intuitive in an adaptive system since the system might want to let the signal remain green if there is no cross-traffic, however this is a limitation most real world signals implement. Future research could possibly analyze the effect of removing this limitation.

6.2.2 Behavior Model

The behavior model is used to decide how each vehicle behaves. Each vehicle agent’s movement is based upon the current location, destination, surrounding traffic and traffic signals. It also depends on the driver ‘personality’ as encoded above.

The basic rules to be followed while modeling the vehicle movement are described as follows:

1. The vehicle is introduced at the map at one of the entry points based on the ‘exit/entry densities’.
2. The vehicle decides its behavior at the upcoming intersection based on what its destination is. For example, if the vehicle shown in figure 1 has to exit at node 23, it will have to turn left at intersection 12.
3. The vehicle attempts to chooses the best lane to be in, based on its destination. If it is within 50 car lengths of the given turning it will attempt to change its lane.

4. If it is possible to enter the desired lane, the vehicle enters that lane. Otherwise it continues on its current path. A vehicle is said to be able to change lanes if it can shift lanes without the other vehicles in the new lane colliding with it. Collision, or the possibility of one, is calculated based on the speeds, acceleration and current position of each vehicle.
5. If the vehicle is not in the correct lane, it will continue on the same path and attempt to change lanes at the next junction. The destination is updated to reflect this change. If it is still unable to change lanes, the vehicle will continue on its current course and exit the system.
6. There are several more complicated lane changing algorithms available that represent real world traffic better. These were not implemented for the sake of simplicity. Some other approaches that could have been added were:
    a. The vehicle slows down as it approaches the desired turning junction in its attempt to find a better opportunity to change lanes. The deceleration would increase with the closeness of the turning junction.
    b. In case a vehicle is forced to stop because it is unable to enter the correct lane, one of the following will occur the vehicle will wait a predetermined time interval before continuing its path and attempting to turn at the next intersection. Vehicles on the destination lane at random (based on the behavior modifiers for the driver) will slow down to let the ‘trapped’ vehicle enter the correct lane.
7. The vehicle scans the road ahead of it. In case there is no obstruction (vehicular or stop signal), it accelerates at a predetermined rate until it reaches the maximum allowable speed. In case it scans an obstruction ahead it takes action based on the following rules.
8. In case there is another vehicle up ahead, the vehicle slows down such that it is able to come to complete stop in ten time units. (Similar to the rule of thumb of staying 2 seconds behind the car in front). In case there is a stop sign up ahead, the vehicle starts slowing down when it is ten time units away from the stop

signal and decides its deceleration based on its current speed. The other factor affecting vehicle behavior at the traffic junction will be the behavior modifiers of the driver as described.

And important aspect of the movement modeling is the behavior modifiers attached to the driver or each vehicle. These are random factors, which will generally have similar values but in cases will be given deviant values. This is to ensure randomness and emergent behavior that models the real world.

For part ‘4’ of the above algorithm, whether or not lane changing is safe should ideally be decided depending on the size of the vehicle and the distance between the last two vehicles. Also, the vehicle attempting to change lanes should ‘signal’ its intention to do so like in the real world. Vehicles within visual range of this signal attempt to adjust so as to allow it to shift lanes if they are further behind the vehicle in front than a specific threshold. This threshold should be decided by the driver behavior modifier but will in general have one specific value.

6.2.3 Graphical Simulation

Though graphical simulation was the first part of the system to be implemented, it was not used consistently through the development of the algorithm. In fact, after the initial development, it was set aside and the final graphical simulation was implemented last. This was because initially graphical simulation was required to confirm that the system was working as desired, that the vehicular readings obtained were real readings and not just random numbers that seemed to make sense (Figure 13). However, once this was confirmed, the utility lost its value for simple testing purposes. In fact, it slowed down simulation due to the amount of processing power required.

Later on in the development of the algorithm, the original single threaded graphical representation was replaced by a multithreaded version of the program. This allowed the

system to run without using up all of the system's resources and graphical representation once again became easy enough to test and view.

At every time interval, the vehicle locations is be written to a memory map. This is pasted directly into video memory in order to speed up the performance of the simulation. On even a normal modern day PC this can simulate a few thousand vehicles per second safely. However the true performance is achieved when this system is used without graphical simulation. Hence, the initial graphical simulation is used just to demonstrate that the simulations model real world traffic correctly. Henceforth, the graphics will be displayed to allow the actual AIS system to run on all the traffic sensors modeled. In the case of a small map, like in figure 1 however, the system can be able to run with graphics on.

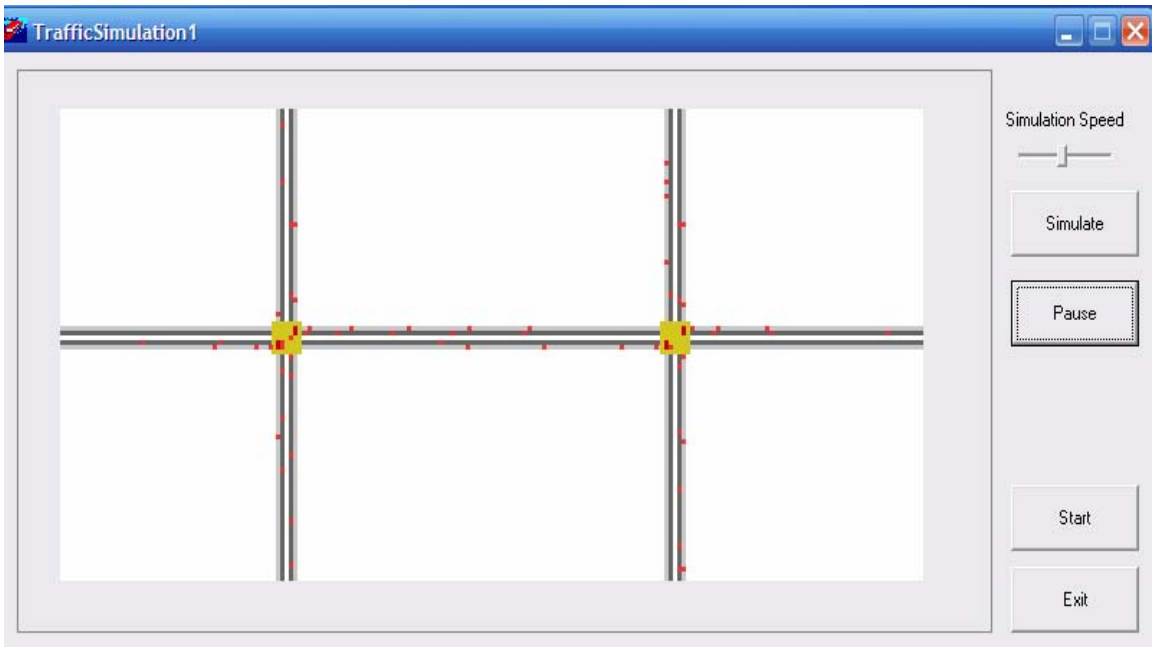

Figure 13: Graphical simulation of the two junction road network modeled for the purpose of current research.

7. STAGE II: IMPLEMENTING THE AIS TRAFFIC CONTROL

7.1 Sensor Assumptions

Initially, the sensors were assumed to be mounted at both the upstream and the downstream locations. These were also assumed to provide information about the velocity and the position of each of the vehicle within the sensing zone. The sensing zone extended from the edge of the intersection to 25% of the length of the link road. However, later on into the research, this was changed to more accurately reflect real life scenarios.

The sensors are now located only at downstream locations as shown in figure 12. They still cover 25% of the longest link road however. Another change made was in the values measured by the sensors. It was more reasonable to make the sensors read only average velocity of the vehicles passing (flow) and occupancy data. As mentioned before, these are valid assumptions and several systems tested on the field use the same measurement values.

7.2 Cell Architecture

Self cells, in AIS, refer to ‘normal’ traffic conditions. These contain the following information:

- Pair-wise nodes (between which the link road runs)
- Traffic information (identifying the flow value of traffic)
- Temporal information (hash number identifying the time and day)

In this case, the ‘day’ information stored referred only to whether the day was a weekday or a weekend. The reasoning behind this is that traffic flow over the weekdays is pretty constant in its characteristics. On weekdays, traffic would flow towards offices and schools in the morning. During evening there would be an influx of traffic from the

office side of the city to the residential side. On a two junction node, this will be a semi-cyclic process.

On weekends however, traffic takes on different characteristics. There are more vehicles out on the road at night and office going traffic during the mornings is infrequent. Vehicles can be expected to go towards malls and shopping complexes, or entertainment centers like movie halls. These rules were also used to model the flow of traffic with more vehicles having one type of destination during specific times of the day.

Based on the above, Self Cell architecture can be described as a class that contains the following information:

- Link information: This is stored as a set of two numbers, for example, the N-S link in the top right of figure 12 contains link information stored as the numbers ‘23’, ‘12’ representing the links at the two ends. The number ‘23’ comes before the number ‘12’ signifying the direction of motion is from ‘23’ towards ‘12’.
- Traffic information: This information may contain whatever number the traffic sensors are able to detect or identify. In our case, these were the flow and occupancy data values. Once again, this information was stored as two integers. The first integer has a three digit value representing the percentage occupancy multiplied by 10. For example, 340 represented 34.0% occupancy. The second integer represented the average speed in miles per hour multiplied by 10, for example, a value of 456 represented 45.6 mph.
- Temporal information: This information was also used simply for matching Ab to Ag just like item (2) above. This contained to numbers, one a Boolean value indicating Weekday or Weekend (0 or 1), and the other an integer value from 1 to 4 representing morning, noon, evening, and night respectively.

Antigens store the same information as self cells. In fact, they have identical structure to that of self cells as described above. What differentiates them from regular self cells is the performance index. If we are given a given specific value of the performance index, any cells that occur when the performance index is above the threshold value fall into the self cell category. All other cells fall into the non-self or Antigen category. As described earlier in the report, the performance index used in this case was based on average queue lengths in the area. This was used both for the AIS and the TRANSYT based control strategy.

A worse Performance Index (PI) should give rise to more Ag in a given area than in other areas. This is what increases the concentration of Ag in areas where PI is worse. But for this reason, it stands to logic, that PI should be measured only over local areas. In the case of our simulation, PI was measured localized because only a two junction system was modeled. If we were to model a larger junction system, it would stand to sense to use similar PI but the PI for one junction should take into consideration only the queue lengths from the current junction, and its immediate neighbors.

Antibodies store the ‘recognition’ code. Each Ab is matched against Self cells via XOR comparison (i.e. checked whether the bits match or don’t) through a randomly generated mask. This mask is important because it ensures the system does not allow problems to slip through unrecognized. This approach was identified in papers on network security *[13]*.

However, for the case of our type of self-cell architecture, a more robust approach was to compare the ‘bytes’ in the system instead of the bits. The secondary recognition methodology used was to match the ‘bytes’ corresponding to similar data sets in the system. This ensured temporal information was compared only against temporal information; traffic information was compared only against traffic information, etc.

When the system was initialized, Ab were generated randomly to match against Self cells (good cells). If the Ab matched the cell, it was discarded. The aim was that the Ab must match only the bad information. Similarly, if a known bad situation occurred, that was placed in the Ab pool as a known bad situation. Future similar situations would be recognized by the AIS and appropriate control cells generated. Matching was done based on Euclidean distance between the two cells.

7.3 Control Cell Generation

Once an Ab had been activated by an abnormal event, it would produce control cells. Control cells encode the type of action to take. (For example, they decide how long the green signal on a particular lane must last). Although the AIS is adaptable to using phase based optimization instead of parametric optimization, in order to compare with TRANSYT, they both used parametric optimization.

When the system was first initialized, control cells were generated based on simple rules. These rules were based on common sense, for example, if a lane is blocked, attempt to unblock it by changing the signal to green. Once this was achieved, the control cells were modified (within the simulation itself) to attempt to find better reactions to the given situation. The modification was done based on simple hill climbing optimizations and random mutations (without the crossover seen in genetic algorithms).

Control cells are explained in further detail in section 7.6.2.

7.4 Secondary Ab and Control Cells

AIS control junctions offer maximum efficiency if they are used in a distributed manner but are still able to communicate with neighboring junctions to a limited degree. This was useful in the following manner.

Every time one junction identified an incident, it sent information about it to the neighboring junctions (Figure 14). Since the junctions are relatively close by, the cost of communication is not very high. Also, since there are only a limited number of neighbors that one junction can have, the complexity of information transfer is not high. This information was used by the second junction to form secondary Ab. These Ab, when activated, were used to predict the arrival of heavy traffic and similar situations. This meant that the junctions were able to prepare for situations ahead of time before they actually happened.

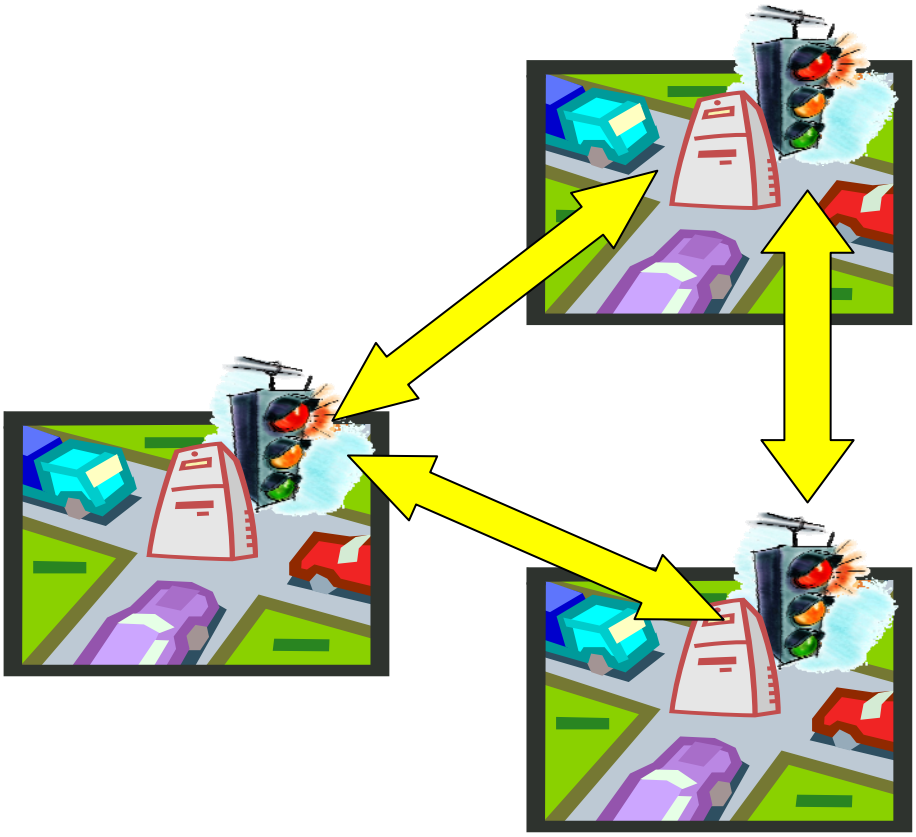

Figure 14: Neighboring junctions communicate to generate secondary antibodies.

Secondary Ab are different from primary Ab in that neither of the junctions mentioned in their pair-wise connection are the same as the junction which stores them. For example, junction 1 would have primary antibodies with a pair-wise junction connection of 1-2, 1-3 etc. However, it would have secondary Ab of type 2-3, 3-4 etc. These are very useful in *predicting* the increase in traffic flow. Once a junction knows that secondary Ab from another junction precede a heavy influx of traffic, it can clear out its cross-traffic faster before the heavy traffic arrives.

Secondary Ab also offer a great deal of help in classifying problems. It is difficult to classify situations based on information from just once link road. However if the junction knows the situation over several links it can predict whether there is just a minor car wreck, or a big traffic jam, or special events like a football game letting out.

7.5 Training Phase

The system starts with normal traffic and keeps a lookout for abnormal conditions. Ab are generated to match abnormal conditions. Control cells that work best for certain situations are generated as described in the rules earlier. Also, Ab are propagated to neighboring junctions to ensure faster recognition using secondary Ab.

During the training phase, the Negative Selection principle of growth is followed. If an Ab matches a self cell, it is discarded, however if it does not, it is allowed to continue growth and eventually matures into a full grown Ab. This method ensures that Ab that come out of the training phase do not match self conditions by accident and sound false positives.

7.6 Algorithm Details

This sections deals with describing in detail how the various parts of the algorithm worked. It starts off first by describing various subsections of the algorithm.

7.6.1 Ab-Ag Affinity Calculation

AIS based algorithms use several different methods for calculating the affinities between Ag and Ab pairs. Some of the most common ones are Euclidean distance, Hamming distance and R-consecutive bit matching. In our case, we started off first by using Euclidean distance to determine Ab-Ag affinity. However this was later discarded because the aim of the system is to be implemented on simple controllers at traffic junctions. Hence, every attempt to simplify the calculations had to be made. Euclidean distance involves the calculation of squares and square roots; hence the affinity matching procedure was changed. Since Euclidean distance did give good results initially, the new algorithm tried to maintain a similar matching protocol and hence we used the sum of absolute differences between the compared number sets.

The algorithm for calculating affinity is described below.

1. Assume the system is trying to examine a potential Ag with the encoded information {23, 12, 340, 456, 0, 2}. The encoded information describes a set of conditions for traffic flowing from junction 23 towards junction 12. The occupancy is 34.0% and the average speed is 45.6 mph. This condition is on a weekday (descriptor number ‘0’) and during the evening (descriptor number ‘2’) as described in section 7.2.
2. The cell under examination, say current-cell, is matched against all known Ab in order to determine its affinity to them.
3. Consider an Ab with the structure {*start-jn, end-jn, occ, flow, day-type, time-of-day*}. Only Ab with starting junction (*start-jn*), ending junction (*end-jn*), day type (weekend or weekday) and time of day matching with the cell under review are examined. This is decided using fast memory lookup which saves time and computation.
4. The affinity is calculated as the sum of the absolute difference between the relevant characteristics. In this case, the relevant characteristics are just the occupancy and flow data. The AIS performs better if there are more relevant

characteristics available but that was not implemented here due to the limitations of most commonly implemented sensors on road networks.

5. In this example, if the value of *occ* and *flow* in the Ab were 550 and 320 respectively, the corresponding value of 'distance from Ab' is going to be calculated as described in the following formula |550-340| + |320-456| which gives the absolute sum as 346. It is important to note that *the lower this number, the higher the affinity*. No real inverse is calculated since that adds another step in the computation; instead the threshold is used as a 'minimum' instead of the 'maximum'. However it must be kept in mind that conceptually, the lower the calculated number, the higher the affinity.
6. The Ab that have distance from Ag below a given threshold, are said to have sufficient affinity to be activated. These Ab are each assigned a weight based on the value of distance from Ag. The weight is calculated as the difference between the maximum distance out of all the distance values and the distance for the current Ab. For example, if there are 3 Ab, with distances 10, 8, and 6; their corresponding weights would be 0, 2 and 4. This in effect, causes the Ab with the lowest distance to have the highest weight. This rank is equal to the weight assigned to each Ab and is used to determine how much priority we wish to give to the control cells associated with each Ab.
7. The reasons for selecting this particular method for calculating affinity are explained above. They can be summarized as:
    a. Maintain similarity to Euclidean distance which was seen to give the best performance out of standard affinity calculation algorithms like Hamming distance, Euclidean distance and r-consecutive bit matching.
    b. Maintain simplicity of calculations because the algorithm is to be implemented on simple and cheap processors that can be marketed as a complete package in traffic controllers.
    c. The algorithm is uses less computational power as opposed to memory requirements. This is because of reason (b) where we try to minimize

computations. In addition, direct memory lookups are very fast and give much better performance for systems which need to work in real time.

d. Lastly, in this case, each Ab is limited to 8 bytes of memory (1 byte each for the two junctions, and temporal information. 2 bytes each for the traffic flow characteristics). Storing even a million Ab uses just 8 MB of memory approximately which is really cheap to implement in a real system.

7.6.2 Selecting and Applying Control Cells

The Ab-control cell system works very much like the B-cell and T-cell system in real immune networks. The Ab are useful in recognizing unwanted conditions (Ag) that occur. Once the Ab are alerted when they have a close enough affinity to a cell they are matched against, the Ab initiate defense mechanisms. The defense mechanism in this case is called a ‘control cell’ and in essence consists of a description of how to divide the split and offset timing signals between various phases in the traffic control.

The concept of phases is explained in Figure 15. One phase refers to a specific combination of traffic flow in various lanes that can occur concurrently without any lane’s traffic interfering with the flow of traffic in other lanes.

Though real traffic conditions can have many more phase sequences depending on the type of traffic flow, and the junction, our system was assumed to have twelve different phases. These can be divided into four subgroups, each of which is similar to the others except rotates by a multiple of 90 degrees due to symmetry of the roads.

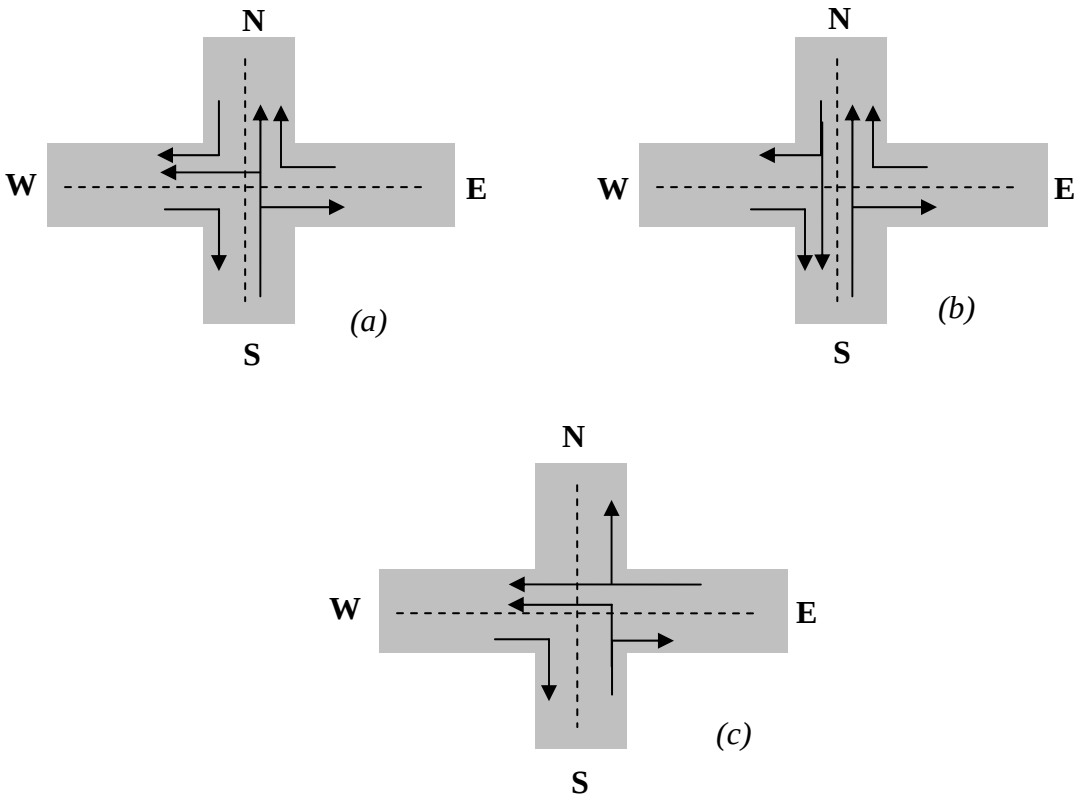

Figure 15: One subgroup of ‘phases’ used in the traffic simulation. These four phases can be used to generate the remaining nine phases by rotating them about the center where the dotted lines meet at 90 degree angles repeatedly.

Now, as has been mentioned in brief earlier, the same Ab may activate more than one control cell. Similarly, the same control cell, may be activated by more than one Ab. Mapping however works only one-way, i.e. from Ab to control cells and never the other way round.

The decision about which phase is to be implemented was decided based on the requirements for each lane. The phase that satisfies the most urgent requirements for lane movement was given preference. The decision on which lanes require free movement most recently was based on the control cells that had been activated.

Consider the set of phases described in figure 15. Let the center junction be denoted by the letter ‘J’. In case the maximum queues were along lanes N-J, S-J, W-J, and E-J in that order. Then the system would use phase (b) as shown in figure 15 in order to maximize traffic flow along the N-S corridor. This would be done by seeing which control cells were most active i.e. which corresponding Ab had the greatest affinity to the current-condition cell. However, at this point in the decision making process, the system would also have to decide based on other factors. Whether the queue build-up on the S-J link was so high that phase (a) might be most profitable since it allows maximum clearance of the S-J traffic. Or in a situation where the junction has a past history that shows that phase (c) shows best results because most of the traffic on the S-J link tends to turn onto the J-W link. These are secondary conditions that the system learns about from experience and adaptation. This in essence explains the basics of how the control cells are generated and various phases decided (Figure 16).

Keeping the above discussion in mind, the algorithm for decisions about how, and how long to sequence the traffic lights can be explained by the following algorithm. The algorithm for mutating the control cells will be explained after this. The reason these two are different is because selecting the phase and the control cells implemented is then used to mutate them and come up with better matches. The selection algorithm:

1. Define ‘control cells’ which contain the cycle time, split, and offset information for each lane. These cells have the form {*start-jn, end-jn, cycle-time, split, offset, phase-ID*}. Phase-ID refers to a unique number used to identify the twelve phases as already described in the previous paragraphs. These control cells are the final cells that act to make a traffic signal work. They define the time between the start of one red signal and the start of the next one. They also define the amount of time into one red signal that the green signal comes on. Lastly, these cells define the sequence in which the lights have to be turned on or off. As such, this is not the only possible implementation of control cells for an AIS

system but they made it simpler to implement the AIS as a comparison to the TRANSYT system because TRANSYT uses controls signals using this exact methodology. They act as a way to fill in the grey area between phases as defined above and the parameter dependant TRANSYT type implementation which most controllers understand. Additionally, they did not result in performance deterioration and were hence kept as such.

2. The control cells are stored in a lookup table that matches each Ab to one or more control cells. Once an Ab is activated, the corresponding control cells are stored in a decision table.
3. If an instance of the activated control cell does not already exist in the decision table, then an instance of the current control cell is added and is assigned a weight equal to the ranking of the Ab which corresponded to this control cell. For a description of Ab ranking, refer to step 6 in the algorithm presented in section 7.6.1.
4. In case an instance of the control cell already exists in the decision table, then the current priority/weight assigned to it is incremented by the weight that would otherwise have been assigned to it. Steps 3 and 4 can be explained as follows. Assume the antibody at rank 54 (Ab-54) activated control cell 1 (CC-1). In this case, CC-1 will be added to the decision table with a priority value of 54. Now assume, another Ab at rank 32 activates the same control cell. Since the control cell is already in the decision table, its existing priority of 54 is incremented by 32 to give a final priority of 86.
5. The above process works on the principle that when something goes wrong with the system, not just once, but several types of Ab are activated by the problem. By combining information from all the activated Ab one can arrive at a more accurate resolution of the exact problem than one would if we just looked at specific Ab.

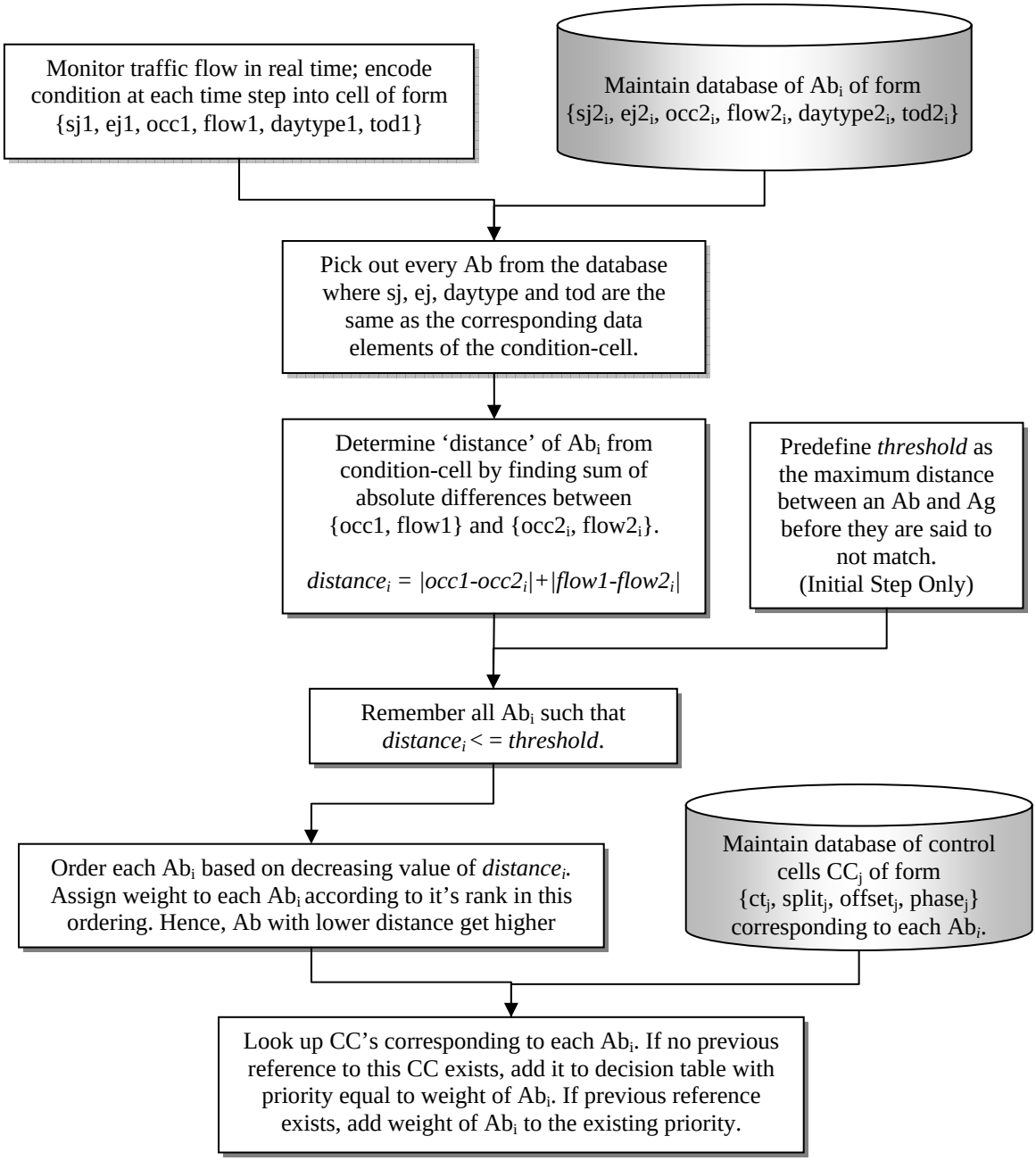

Figure 16: Logical flow of the algorithm to calculate and use affinity values for Ab. The figure also explains how control cells are used.

7.6.3 Antibody Generation

Ab generation depends on the constant calculation of a performance index (PI). In order to maintain the same index as is used in the implemented version of TRANSYT, the PI was chosen as the average sum of queues over the local network. It has been pointed out that in a larger scale simulation, the concept of a local PI would have to be considered carefully. In the heuristic system discussed here, the antibodies and the decision making are highly local. In fact, that is one of the most attractive features about this implementation, that it is distributed and scaleable. Hence, in a larger simulation, it must be kept in mind that the PI should be calculated only for local neighborhoods. This is to say, the PI at one junction depends only on the queues at that junction, and the junctions immediately neighboring it.

Now Ab generation occurs in two stages. The first stage is during training when most of the Ab are generated and stored into the database. The second stage occurs during actual running of the system when the Ab are generated in real time to account for problems that had not already been encountered.

During both training and deployment, Ab generation is partly dependent on the performance index to decide whether a given situation is a problem (Ag) or not. However, there is a minor difference in how problems are identified during training and during deployment. During training, possibly erroneous situations are introduced into the simulation on purpose. In such a case, the Ab do not check PI but are generated regardless of the PI value to match the problem condition. For example, problems introduced may be exceptionally heavy traffic flow introduced into one lane, or one car purposely ‘stalled’ on the road. During deployment however, we do not have this luxury and must create Ab based simply on PI values within the neighborhood.

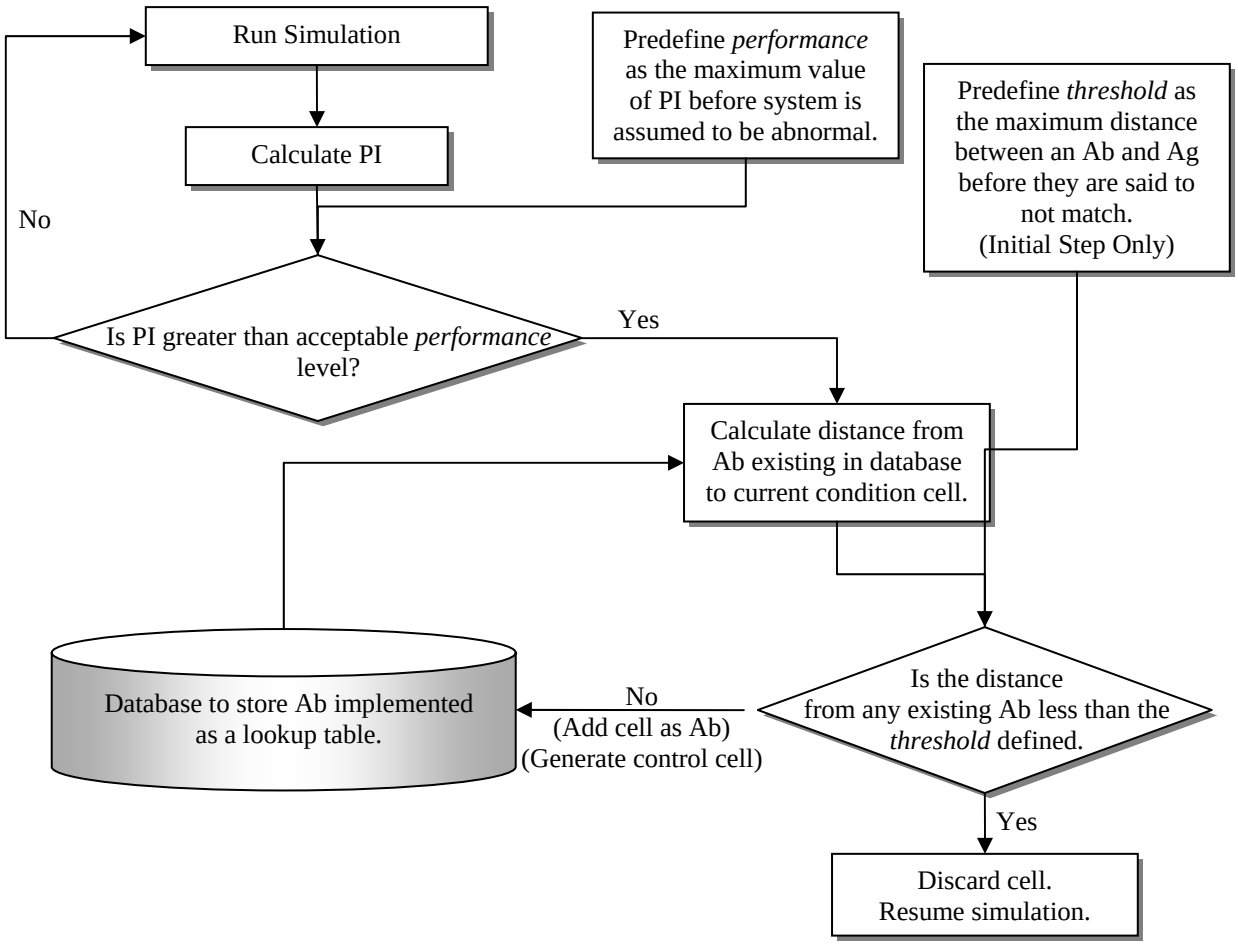

Figure 17: Logical flow of the algorithm to generate Ab while simulation is running based on PI. Generation of Ab also occurs from other mechanisms which do not depend on PI as discussed in section 7.6.3.

During the training phase, in addition to the above mechanism, an additional human controlled mechanism is added to simulate conditions that might arise in the real world. These situations as already discussed above generate their own Ab and are used to make the system work better.

The step in the flow diagram (Figure 17), where the Ab are added to the database also involves the generation of a control cell that matches the given Ab. The initial control

cell is generated based on ‘common sense’ rules. The entire process of control cell generation is explained in the following section.

7.6.4 Control Cell Generation and Evolution

As explained in section 7.6.2, control cells store information about the timing sequence and about the phase to be used. This representation enables made it easy to implement the TRANSYT benchmarking system on the current simulation itself.

As shown in figure 17, if a new type of Ab is discovered, it is added to the database storing all other Ab. At the same time, this Ab is associated with a specific control cell that is the first control cell used when the Ab is activated. The generation of the control cell depends on a somewhat complicated series of rules even though each of these rules in itself is simple. The rules are implemented as a series of if-else conditions and hence are computationally not very expensive. These can be summarized as follows:

1. Attempt to clear traffic on the lane with the longest queue. In order to do this, each lane is assigned a weight based on the size of standing queue of traffic on that lane. The phase that allows the maximum number of high priority lanes to move is selected.
2. This selection sometimes allows more than one type of phase to be selected. In case one or more phases allow the same type of lane clearance; phases are assigned based on the last phase that occurred right before this incident was detected. In many traffic systems, phases follow a kind of ‘sense’; that is to say, one type of phase usually follows another. For our example, phase ‘a’ is allowed to follow phase ‘c’ as shown in figure 15.
3. The decision on which phase to select is also based on the past history of the lane. In many cases, past experience will show that one type of phase is better suited to clear traffic of one kind even though it may seem counterintuitive. For example, if most of the traffic in figure 15 moving along the S-J lane turns left, then phase ‘c’ might end up making more sense than phase ‘a’ to clear traffic

faster. Such decisions about past experience have been observed to appear as part of the emergent behavior of the AIS control implementation.

In order to allow an evolutionary emergent behavior to develop in addition to the rule-based generation of control cells initially, certain steps were taken.

1. Newly generated antibodies were communicated between neighboring nodes. Also, each node communicated to the neighboring nodes which Ab had been activated in the last decision step. These steps acted to serve two purposes:
    a. The receiving node kept one original copy of the Ab intact. In case incidents of activation of this Ab in the first node always preceded the activation of one set of Ab in the receiving nodes (with an 80% similarity rate in the activated Ab each time), the receiving node used this Ab to form its own control cell where such a control cell made sense. This control cell in essence acted to preemptively guess at the occurrence of an incident.
    b. Each node translated the Ab to an Ab in it's own local coordinate system. For example, from figure 12, an Ab with starting and ending junctions 32 and 12 was converted to an Ab with starting and ending junctions 31 and 11 for node 11 using symmetry. This allowed for the nodes to learn faster.
2. In cases with more than one option in the control cell to select for maximum lane movement, the entire set of possible control cells was tested over a period of time when they were activated. Also, random mutations with a 5% probability allowed for one phase to be changed into another phase even if it did not fulfill the rule based criteria. In such cases however, a record was kept to check whether it resulted in a faster resolution of the problem. The speed at which the problem was resolved was determined based on how fast the system was able to change to another phase. The system giving the fastest performance for a given set of Ab was given preference evolutionarily allowing for the best system to emerge using survival of the fittest. Historical

3. The emergent behavior for the distributed system was also seen when randomly changing the phases between the two lanes resulted in a better combination for the system to work.
4. Each phase, once assigned must continue for at least 30 seconds. This is necessary in order to avoid too frequent changes in the light status. This was implemented because most real world traffic control systems require a built in minimum time for a phase. Additionally, if the phase was allowed to switch arbitrarily fast, it might result in situations where the traffic didn’t have time to cross the junction, or even start moving before the phase changed.

8. STAGE III: BENCHMARKING AGAINST TRANSYT

Benchmarking was done against the well accepted standard TRANSYT signal control. TRANSYT typically uses the concept of platoon dispersion to model flow progression along a link. In this case, it was implemented on the same micro-simulated traffic model as the AIS system was based on. The TRANSYT system was simplified to fit within the framework of this research simulation.

8.1 Signal Control

As shown in figure 4, TRANSYT uses a concept of Split and Offset to describe the Cycle Time. Cycle time is the total duration of one phase of signals. One phase refers to one particular configuration of traffic flow (where all traffic in a few particular directions is flowing and the rest is stationary). The values of signal control timings were changed to introduce steps in the hill climbing algorithm that TRANSYT requires. The simulation was allowed to run for a thousand time steps in order to allow the system to reach equilibrium before the TRANSYT system was started. This is necessary in order to eliminate the problem of randomness in the starting situation. Also, the system at start does not look like a real traffic system due to the lack of vehicles on the road.

8.2 TRANSYT Description

TRANSYT has been much modified and developed since its first description, but for the purpose of this study, only the basic model was used. The steps involved in TRANSYT controlled optimization can be described as follows.

1. Initially, the system settings include pre-specified configurations of phase and stage.
2. The system is allowed to run through the simulation starting from a given set of conditions of traffic. The performance index (PI) is calculated for this run of the test.
3. The initial conditions are then modified by a simple hill climbing optimization algorithm. A hill climbing algorithm changes the system parameters by a small

amount and runs the simulation again. It does so for all possible changes to the system parameters.

4. The best change is said to be the one that gives maximum improvement in the performance index.
5. This change is then applied to the system and is used as the starting point for a new search.
6. The search continues until no further improvement is seen.
7. This final staging is then implemented into the real world.

As can be seen, the algorithm optimizes for a given set of traffic conditions. Hence in this case, the random behavior of the drivers and traffic had to be removed in order to best reach the optimum for a given set of conditions. This also meant that the traffic simulation model had to be updated in order to allow for it to be run multiple times. The simulation system had to also be modified in that the cars could no longer be introduced at random *each* time the simulation ran. Doing so would have introduced unnecessary randomness into the TRANSY system.

Where this system fails is that it is optimized only for one set of conditions. Hence better optimizations mean running the simulations for different conditions (like night and day) and storing different strategies for each. This was also done, with 2 different sets of conditions stored (night and day traffic). Weekend and special event traffic (carnivals, big games, etc.) was not simulated for training TRANSYT.

8.3 Benchmarking Results

The benchmarking performed as expected. Since TRANSYT is optimized for one particular situation in the traffic simulation – it performs really well in the beginning. Eventually, its performance starts degrading because it is not adaptable in real time. The AIS on the other hand, learns from experience, and its performance improves over time.

It can be seen in Table 1 that within a thousand time steps it had started outperforming the TRANSYT system.

Table 1: Benchmarking results for TRANSTY vs. AIS

Average P.I. value improvement over ‘dumb’ controllers (10 runs)		
Time Steps	TRANSYT	AIS
300	20%	12%
1000	15%	16%
5000	13%	16%

Also, another important aspect of AIS that were tested were the predictive abilities of the AIS. As such this does not classify under benchmarking because TRANSYT is not a predictive tool for future traffic problems. Nor is it capable of classifying traffic problems. However this was an important area where AIS was found to perform much better. The secondary Ab were successfully able to stimulate neighboring junctions into taking predictive action. They were also able to sound alerts ahead of time.

The alerts discussed in this case are messages to human operators that something is about to go wrong. The AIS was able to handle the traffic flow on its own but this is a useful tool to have which can possibly be integrated into GPS systems to better plan the best route for travel for vehicles equipped with the system. As such, this was a proof-of-concept and a detailed study was not conducted into this feature of the AIS

implementation. It is however believed that this property caused the AIS to perform better than the TRANSYT system over longer durations of testing times.

9. COMMONLY USED TRAFFIC SENSORS

This section describes some of the common sensing technologies available to sense traffic characteristics on urban roads. Although AIS based control is more or less independent of the type of sensor used, a survey of these sensors is pertinent because not all sensors can measure all the types of data required for most control systems to perform ideally.

9.1 Inductive Loops

By far the most common method for sensing road traffic is Inductive Loops. Inductive loops are installed usually after the road has been covered with asphalt. A loop is dug into the road and metal wires places along it. These are covered by rubber compound *[19]* and act on the principle of magnetic induction. The ferrous material in the body of the vehicles that pass over the loops changes their magnetic induction. Measuring this change in induction can give an idea about the size of the vehicle that passes and gives approximate queue lengths and occupancy data about the amount of traffic on the road.

Inductive loops can also be used to measure the speed of cars if two inductive loops are used. By measuring the time between subsequent activations of the loops and knowing how far apart they are, the loops can give an approximation of the speed of the vehicle that last passed. These are amongst the cheapest sensors to install and are unaffected by weather conditions. These advantages are offset however by the inaccuracy inherent in the induction detection system. They can be thrown off by vehicles that do not contain a lot of metal and are not very good at differentiating individual vehicles.

9.2 Visual Surveillance

CCTV cameras are often used on traffic signals to monitor traffic conditions in conjunction with existing sensors. The advantage of visual surveillance is the amount of data it is able to gather in a very short period of time. However there are not many

system where the camera itself is able to do all the necessary calculations. Usually, they cameras are hard-cabled to a central transport authority where a human operator oversees their operation. With the decrease in cost of cameras and computers, and increasing computational power, these systems are becoming a more and more viable alternative to older sensors because of the amount of information they can gather. Currently though, they usually suffer from the disadvantages of being costlier and being affected by weather conditions.

9.3 Pressure Tubes

Pressure tubes are installed much like inductive loops and offer the same benefits of low cost. Additionally, its measurements are unaffected by the existing weather conditions. On the flip side, installation of these systems is labor intensive and might disrupt traffic flow. Also, they require somewhat more maintenance than inductive loop sensors would.

9.4 Passive Acoustic Sensors

These respond to the sound of traffic and use it to measure the traffic conditions. These sensors are capable of measuring the speed, volume and occupancy data. As is expected from sound based sensors, these require good filtering software in order for them ignore the noise that is inherent in the environment.

10. ADVANTAGES OF AIS OVER CONVENTIONAL SYSTEMS

10.1 Incident Detection/Classification

The problem of incident classification is to form abstraction at the highest level do describe an incident. Current systems in place like SCOOT, RHODES etc. are unable to do so as shown in Table 2. AIS on the other hand, starts off by classifying and is an inherently strong classifier/pattern recognition system.

Table 2: Incident detection capabilities of existing systems

System	**Description**	**Automatic Incident ID/Classification**
SCOOT	Feed forward loop – predict arrival of traffic at junctions.	No
SCATS	Feed back loop – predict emptying of queues at junctions.	No
OPAC	Rolling horizon type	No
RHODES	Multi-tier optimization based on TRANSYT	No

The current system used was able to provide some level of abstraction in recognizing similar sets of conditions that arose when an attempt was made to simulate abnormally high traffic flow in along three lanes.

10.2 Training Time and Human Intervention

All agencies agree that successfully operating and maintaining the system is a substantial time investment and requires at least a year with the system before the person becomes proficient in its use. *[20]* Very often the systems require full time engineers working to second guess and tweak the optimization process. A lot of the efficacy of this approach then depends on the knowledge and experience of the engineer. There has not been any effective way to translate this knowledge/experience into algorithms. AIS

implementations on the other hand require minimal human intervention. Also, there is practically no training time involved.

10.3 Scalability

AIS implementations allow for ad-hoc placement of the traffic signal/controller and little or no changes need be made at a central location. This will ensure that they system can be put up in any location. Once an AIS based controller is created with the initial set of Ab, the controller becomes ‘plug and play’. Once installed in a new location, it can borrow secondary Ab from neighboring junctions while it developed Ab of its own. These borrowed secondary Ab along with its own predefined Ab should suffice to keep the system working without failure. Also, since the system is independent of the type of sensor used, it ensures the controllers are compatible with all types of junctions and city systems. Adding a new controller to the junction will also not add to the overall complexity of the system except in a linear manner.

10.4 Distributed Control

The fact that AIS offers distributed control is very important in a complex environment like UTC. Central computers are often not capable of planning out an entire city’s traffic flow optimally in real time and have to rely on semi-real time updates, which is to say, they use updates at fixed time intervals. For example, SCOOT updates at 4 second intervals.

AIS offers distributed control with linear scalability over the local area. Addition of new nodes neighboring another will not appreciably change its performance speed. Also, since most of the operations done are memory lookups, the performance is much faster than systems than actually spend time calculating the flow of traffic.

11. FUTURE WORK: HARDWARE IN LOOP TESTING

The current work was limited in scope because even though it benchmarked against an accepted standard algorithm, it did not base itself on an accepted standard simulation of the urban traffic it was trying to control. It is expected that the system will be able to overcome this limitation by using hardware in loop (HIL) testing available at the Texas Transportation Institute at Texas A&M University.

TTI offers two distinct approaches to using a standardized algorithm. One would be to use the VISSIM simulator software. VISSIM offers a simple programming interface akin to the language BASIC. It was built with the intention that any program capable of running on the simple processors that are available to install with traffic junctions should be programmable in a simple language. VISSIM offers the advantage of being completely within a simulation. Hence ‘real time’ simulations would run much faster than they would with a real hardware in loop system.

The other alternative is to use the CORSIM simulation package which has been set up with a hardware in loop controller interface device (CID). The CID acts as the bridge between the simulation package and the user’s program as shown in Figure 18. The current hardware installed in the TTI complex consists of CID’s built by the Louisiana State University. *[21]*

These can be connected to any of several micro-simulation packages available, the most commonly used one being CORSIM. These packages, together with the CIDs offer a very good simulation base to start the next stage in the AIS research. The CID’s can be connected over local area networks to the various control computers. This offers the advantage that they can be accessed remotely from anywhere in the world over the Internet. The current interface however is designed for serial port access in order to increase accessibility from laptops. Even though serial ports are slower in

communication, this offers a very good way to simulate real world conditions because almost every simple control device supports serial communication.

While implementing HIL testing, it becomes much easier to ensure that the system is be able to perform in real time. If the control system is unable to send the traffic signal within the second that it was supposed to, the simulation just continues as it were and the performance index changes appropriately.

Programming for the HIL testing can be done in any language that supports serial communication. It is expected that the current program can be modified to serve as a stand alone application able to communicate with and control the CID signals. Since C++ is a powerful language and the current program has already been developed in it, it is recommended that this same language be used to implement HIL testing.

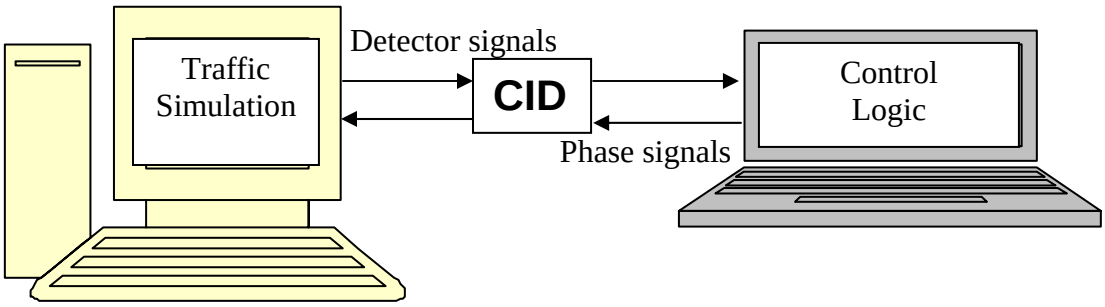

Figure 18: The Controller Interface Device (CID) forms an interface between the simulation and the controller.

The simulation package (say, CORSIM) will simulate the flow of traffic in real time. The detectors simulated inside of CORSIM will send information to the CID. The CID in turn, sends information to the controller which can be a hardware controller or a virtual controller simulated on a laptop. All of this should ideally work in real time especially if a hardware controller is used. However with simulated controllers, it might be feasible to speed up the time scale. The data acquired from the CID by the controller can be analyzed by the control algorithm and phase sequences can be fed back to the controller. Figure 19 shows the flow of control and data in this setup.

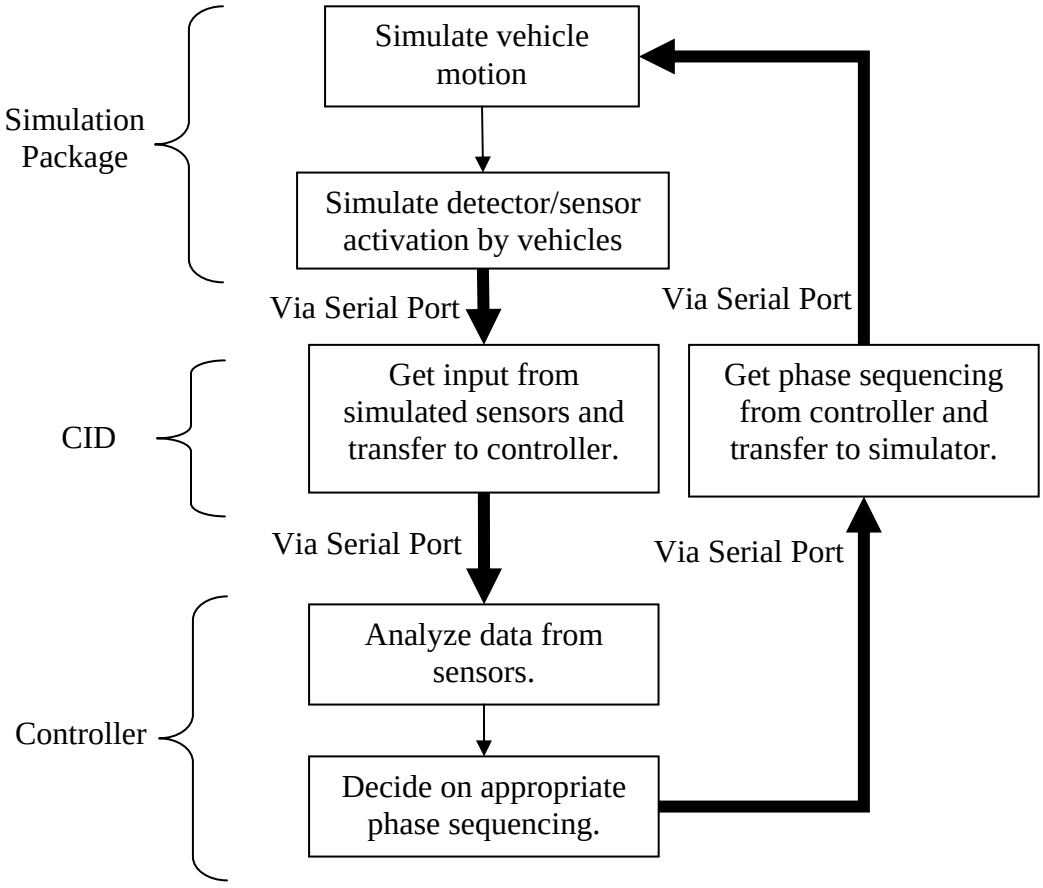

Figure 19: Data and control flow in the HIL implementation.

Communication with the LSU CID package occurs over standard serial cables. This can be achieved via C++ which was the original programming language.

The first step in serial communication would be to create a variable of data type HANDLE for communication using the *CreateFile* command. Serial I/O in visual c++ will work pretty much exactly like file I/O. A typical format for the CreateFile command is described below:

```
HANDLE CreateFile(
  LPCTSTR lpFileName,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile
);
```

Translated into source code, the initialization would look something like the code segment described on the following page. The first step would be to declare a HANDLE type variable, followed by initializing it using the *CreateFile* command.

```
HANDLE hCommPort;
hCommPort = CreateFile( szPort,  // null terminated string //name of
                                                  port
                    GENERIC_READ | GENERIC_WRITE,
                    0,         // signfies exclusive access
                    0,         // signifies no security
                    OPEN_EXISTING,
                    FILE_FLAG_OVERLAPPED, // whether overlapped or not
                    0);
if (hComm == INVALID_HANDLE_VALUE)
   // if above condition is true, it signals an error in opening the
    port


DCB dcb = {0};
if(!GetCommState(hCommPort, &dcb))
        // error: GetCommState failed
else
        // set values for the comm port
```

The communication port values can be set using the following sample code.

```
dcb.BaudRate = CBR_57600;     // set the baud rate
dcb.ByteSize = 8;             // data size, xmit, and rcv
dcb.Parity = NOPARITY;        // no parity bit
dcb.StopBits = ONESTOPBIT;    // one stop bit
```

Reading from the serial port can be accompalished using the Read command or the ReadFile command as shown.

```
unsigned long read_val = 0;
char charbuff[128];
ReadFile(hCommPort, charbuff, 128, &read_val, 0);
```

The above sample code can be used to start off serial communication with Win32 using multithreading and overlapped operation. Such operation might be useful when the program expands beyond simple data acquisition and control. For a more complicated system, this approach might end up being more useful. Additionally, this system is useful as it hides some of the hardware from the programmer. Programmers with little or no experience with hardware programming would find this approach simpler.

In case the programmer has some experience with hardware programming, we can start off with a lower level approach that refers back to the original C/C++ implementation for COM port communication. A different compiler than the .Net version of Visual Studio might have to be used. This would offer significant simplification in the C++ code from the view of a programmer who does not have much experience with Visual programming. A code snippet explaining serial programming using this approach is given below.

```
#define SERPORT1 0x3F8 //define as base address for serial port

outportb(SERPORT1 + 1 , 0); // write zero to the interrupt enable bit
outportb(SERPORT1 + 0 , 0x03);  // Set a baud rate of 38,400 byte/sec
char ch = inportb(SERPORT1); // Read a character from the serial port
```

The above code sequence explains how one can open the serial port and set values for read/write. It also gives samples on how to read and write to it without using the file reading commands from windows programming. The programming method used would depend on the comfort level of the programmer with Windows or with hardware programming.

12. SUMMARY AND CONCLUSIONS

The control of urban traffic over a two junction road network was developed and analyzed. The control was based on a branch of artificial intelligence known as Artificial Immune Systems. In order to study the system, first a micro-simulation of the traffic flow over the two junction road network was developed. Though a simple implementation, the simulation was expected to perform reasonably well and within expectations as a real road network. A graphical user interface was developed to study the performance. The system was benchmarked against the accepted standard TRANSYT traffic management system. TRANSYT is a fixed timing sequence control and was expected to perform better during the initial conditions that it was trained for. The Artificial Immune System is an adaptive control strategy and was expected to perform better as and when it learned to optimize. The results can be summarized as follows.

The traffic simulation was examined visually after being represented by the graphical user interface and performed within expectations. It was able to simulate real traffic well and there were no noticeable discrepancies that stood out.

The Artificial Immune System was trained per the plan, with each link acting as an identifier in the self cells along with specific traffic data (flow and occupancy) which are easily measurable quantities. It also included temporal information that was used to classify the data into different days between weekdays and weekends. The system once trained was able to distinguish between self and non self as expected. It was also used to generate control cells to decide which signal phase to activate.

The system was benchmarked against TRANSYT and the performance was as expected. Initially TRANSYT gave a performance boost of 20% over unintelligent timed control while AIS gave a 12% improvement. Over 5000 generations though, the performance of

the AIS improved to give 16% improvement while TRANSYT gave 13% improvement on average.

REFERENCES

[1] D. I. Robertson, “TRANSYT method for area traffic control,” *Traffic Eng. Control*, vol. 10, pp. 276–281, 1969.

[2] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, “Review of road traffic control strategies,” in *Proc. IEEE*, vol. 91, pp. 2043 – 2067, Dec. 2003.

[3] J. D. C. Little, M. D. Kelson, and N. H. Gartner, “MAXBAND: A program for setting signals on arteries and triangular networks,” *Transportation Res. Rec. 795*, pp. 40 – 46, 1981.

[4] D.I. Robertson and R.D. Bretherton, “Optimizing networks of traffic signals in real time-the SCOOT method,” *IEEE Trans. Veh. Technol.*, vol. 40, pp. 11 – 15, Feb. 1991.

[5] P. Dell'Olmo, and P. B. Mirchandani, “REALBAND: An approach for real-time coordination of traffic flows on networks,” *Transportation Res. Rec. 1494*, pp. 106 – 116, 1995.

[6] K.L. Head, P.B. Mirchandani, and S. Shelby, “The RHODES prototype: a description and some results,” presented at the 77th Annual Meeting Transportation Res. Board, Washington D.C., 1998.

[7] K.L. Head, “An event-based short-term traffic flow prediction model,” *Transportation Res. Rec. 1510*, pp. 45 – 52, 1995.

[8] M. Patel and N. Ranganathan, “IDUTC: An intelligent decision-making system for urban traffic-control applications,” *IEEE Trans. Veh. Technol.*, vol. 50, pp. 816 – 829, May 2001.

[9] F. Boillot, J.M. Blosseville, J.B. Lesort, V. Motyka, M. Papageorgiou, and S. Sellam, “Optimal signal control of urban traffic networks,” presented at the Sixth Int. Conf. Road Traffic Monitoring, pp. 75-79, London, 1992.

[10] F. Busch, and G. Kruse, “MOTION for SITRAFFIC - a modern approach to urban traffic control,” in *Proc. Intelligent Transportation Systems*, pp 61 – 64, Aug. 2001.

[11] M. Ayara, J. Timmis, R. D. Lemos, L. N. D. Castro, and R. Duncan, “Negative Selection: How to generate detectors,” in *Proc. 1st Int. Conf. on Artificial Immune Systems (ICARIS)*, pp. 89 – 98, Sep. 2002.

[12] L.N. Castro, and F.J.V. Zuben, “An evolutionary immune network for data clustering,” in *Proc. IEEE Brazilian Symposium on Artificial Neural Networks (SBRN),* pp. 84-89, Nov. 2000.

[13] S. Forrest, A.S. Perelson, L. Allen, and R. Cherukuri, “Self-nonself discrimination in a computer,” in *Proc. IEEE Sym. Res. Security and Privacy*, pp 202-212, May 1994,.

[14] N.K. Jerne, “Towards a network theory of the immune system,” *Ann. Immunol. (Inst. Pasteur)* 125C, pp. 373-389, 1974.

[15] S. A. Hofmeyr and S. Forrest, “Immunity by design: an artificial immune system,” in *Proc. Genetic and Evolutionary Computation Conference (GECCO),* pp. 1289-1296, July 1999.

[16] S.P.N. Singh, S.M. Thayer, and W.P. Thayer, “A foundation for kilorobotic exploration,” in *Proc. Congress on Evolutionary Computation,* vol. 2, pp. 1033-1038, May 2002.

[17] M. Araujo, J. Aguilar, and H. Aponte, “Fault detection system in gas lift well based on artificial immune system,” in *Proc. Int. Jt. Conf. Neural Networks,* vol. 3, pp 1673 - 1677, July 2003.

[18] P. Negi, and R. Langari, “Artificial immune networks for multi-sensor fault detection”, presented at the American Control Conference, Houston, Texas, Jun. 2005.

[19] Howstuffworks, "How does a traffic light detect that a car has pulled up and is waiting for the light to change?,” http://auto.howstuffworks.com/question234.htm, (accessed Nov. 16, 2006)

[20] P. Martin, “Adaptive control systems survey,” *Workshop on Signal Control Priority for Transit and Adaptive Signal Control,* Washington, D.C., Jan. 2004.

[21] R.J. Engelbrecht, C.M. Poe, and K.N. Balke. "Development of a distributed hardware-in-the-loop simulation system for transportation networks," in *Proc. Transportation Res. Board Annual Meeting,* Washington, D.C., January 1999.

VITA

Name:	Pallav Negi
Address:	Texas A&M University, Department of Mechanical Engineering,
	3123 TAMU, College Station TX 77843-3123
Email:	pallav@tamu.edu
Education:	B.E., Manufacturing Processes and Automation, University of Delhi, 2003
	M.S., Mechanical Engineering, Texas A&M University, 2006