INTEGRATED AND ADAPTIVE TRAFFIC SIGNAL CONTROL FOR DIAMOND INTERCHANGE

A thesis presented in partial fulfilment of the
requirements for the degree of

Doctor of Philosophy
in
Mechatronics Engineering

at
Massey University, Albany
New Zealand

Cao Van Pham

February 2017

ABSTRACT

New dynamic signal control methods such as fuzzy logic and artificial intelligence developed recently mainly focused on isolated intersection. Adaptive signal control based on fuzzy logic control (FLC) determines the duration and sequence that traffic signal should stay in a certain state, before switching to the next state (Trabia et al. 1999, Pham 2013). The amount of arriving and waiting vehicles are quantized into fuzzy variables and fuzzy rules are used to determine if the duration of the current state should be extended. The fuzzy logic controller showed to be more flexible than fixed controllers and vehicle actuated controllers, allowing traffic to flow more smoothly. The FLC does not possess the ability to handle various uncertainties especially in real world traffic control. Therefore it is not best suited for stochastic nature problems such as traffic signal timing optimization. However, probabilistic logic is the best choice to handle the uncertainties containing both stochastic and fuzzy features (Pappis and Mamdani 1977)

Probabilistic fuzzy logic control is developed for the signalised control of a diamond interchange, where the signal phasing, green time extension and ramp metering are decided in response to real time traffic conditions, which aim at improving traffic flows on surface streets and highways. The probabilistic fuzzy logic for diamond interchange (PFLDI) comprises three modules: probabilistic fuzzy phase timing (PFPT) that controls the green time extension process of the current running phase, phase selection (PSL) which decides the next phase based on the pre-setup phase logic by the local transport authority and, probabilistic fuzzy ramp-metering (PFRM) that determines on-ramp metering rate based on traffic conditions of the arterial streets and highways. We used Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network (AIMSUN) software for diamond interchange modeling and performance measure of effectiveness for the PFLDI algorithm. PFLDI was compared with actuated diamond interchange (ADI) control based on ALINEA algorithm and conventional fuzzy logic diamond interchange algorithm (FLDI). Simulation results show that the PFLDI surpasses the traffic actuated and conventional fuzzy models with lower System Total Travel Time, Average Delay and improvements in Downstream Average Speed and Downstream Average Delay.

On the other hand, little attention has been given in recent years to the delays experienced by cyclists in urban transport networks. When planning changes to traffic signals or making other network changes, the value of time for cycling trips is rarely considered. The traditional approach to road management has been to only focus on improving the carrying capacity relating to vehicles, with an emphasis on maximising the speed and volume of motorised traffic moving around the network. The problem of cyclist delay has been compounded by the fact that the travel time for cyclists have been lower than those for vehicles, which affects benefit–cost ratios and effectively provides a disincentive to invest in cycling issues compared with other modes. The issue has also been influenced by the way in which traffic signals have been set up and operated. Because the primary stresses on an intersection tend to occur during vehicle (commuter) peaks in the morning and afternoon, intersections tend to be set up and coordinated to allow maximum flow during these peaks. The result is that during off-peak periods there is often spare capacity that is underutilised. Phasing and timings set up for peaks may not provide the optimum benefits during off-peak times. This is particularly important to cyclists during lunch-time peaks, when vehicle volumes are low and cyclist volumes are high. Cyclists can end up waiting long periods of time as a result of poor signal phasing, rather than due to the demands of other road users being placed on the network.

The outcome of this study will not only reduce the traffic congestion during peak hours but also improve the cyclists' safety at a typical diamond interchange.

ACKNOWLEDGEMENTS

Firstly, I would like to thank my supervisors Prof. Peter Xu and Dr. Fakhrul Alam for mentoring and providing me with knowledge in this field. I am really grateful for the invitation to do my Doctor of Engineering. Both gave me the great feeling of being an essential part of intelligent transportation system research group. I am especially thankful for providing me with the opportunities in attending and presenting papers at conferences during my study as well as support for my thesis work.

Secondly, I would like to thank Ass. Prof. Johan Potgieter and Ass. Prof. Clara Fang for their valuable comments and supporting during my time of study.

Gratitude is also expressed to the staffs of Auckland Traffic Operation and Management Unit, North Shore City Council, Transit NZ for their valuable data and support.

I am very grateful to Massey University, for providing the scholarship during my study.

Furthermore, I would like to thank my best friend Xuefeng Yu (Aaron) for his support.

Finally, I would like to thank my parents and my sister who have been behind all my achievements in life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ADIFM	Actuated Diamond Interchange Control with Fuzzy Ramp Metering
AIMSUN	Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network
ATOM	Auckland Traffic Operation and Management Unit
DAD	Downstream Average Delay
DAS	Downstream Average Speed
FLC	Fuzzy Logic Control
FLDI	Fuzzy Logic Diamond Interchange Control
GTE	Green Time Extension
HOV	High Occupancy Vehicle
SH1	State Highway 1
MF	Membership function
MOE	Measures of Effectiveness
MOTORWAY	Freeway (US) / Highway
NZMOT	New Zealand Ministry of Transportation
NZTA	New Zealand Transport Agency
PFLDI	Probabilistic Fuzzy Logic Diamond Interchange Control

PFLDIBC	Probabilistic Fuzzy Logic Diamond Interchange Control Incorporating Bicycle Signal
PFPT	Probabilistic Fuzzy Phase Timing
PFRM	Probabilistic Fuzzy Ramp Metering
STTT	System Total Travel Time
TFL	Traffic Light
TTT	Total Travel Time

CHAPTER 1

INTRODUCTION

1.1 DIAMOND INTERCHANGES PROBLEMS

There are many types of interchanges; the most common types are the diamond interchanges which are made of two intersections, and each connecting to a motorway direction. Most of these intersections are **signalised** when the demand flows are high (peak hours). The **signalised** intersections connecting to the arterial cross street are often the key operational element within the interchange system (Fang, 2004). The distance between the two intersections varies from less than 120 meters in densely developed urban areas to 240 meters or more in suburban areas (Messer et al, 1997). The short distance between the two intersections limits the storage available for queued vehicles on the arterial cross street. The signal timing is very important in controlling the traffic flow for the intersection during the peak hours, if the signal timing is not set properly, the queue from the downstream intersection will spill back and block the upstream approaches (Fang, 2004). Highway Capacity Manual (2000) points out an affect known as demand starvation which occurs when portions of the green at the downstream intersection are not used because conditions prevent vehicles at the upstream intersection from reaching the downstream stop-line.

1.2 ADAPTIVE SIGNAL CONTROL

There are three types of control in practice includes pre-timed, actuated and adaptive control. The first one, pre-timed control, operates based on fixed intervals and phase timings with fixed number of cycle length and phase sequence. The disadvantage of this type of control, is they become less effective when traffic condition changes substantially. Advanced from the pre-timed is the actuated control which is based on the gap –seek logic to respond to traffic fluctuations during the day. The controller extends the green beyond a minimum time until a gap in the traffic flow on the approaches currently with green has been detected or until the pre-defined maximum green time has been reached. Actuated signal control has certain limitations including tendency to extend green inefficiently under low traffic flow conditions, and great sensitivity to incorrectly set maximum green times

(Fang 2004). Adaptive control can be defined as any signal control strategy that can adjust signal operations in response to fluctuating traffic demand in real time according to certain criteria (Lin and Vijayayumar 1988). Adaptive control makes signal timing decisions based on detected or identified current and short-term or even long-term future flow. Since Miller's pioneer work (Miller 1963), considerable research has been done to develop adaptive systems. Systems such as SCOOT (Hunt 1982), SCATS (Luk 1984, Charles 2001), OPAC (Gartner 1983, Gartner and Pooran 2002) and RHODES (Mirchandani and Head 2001) are among the best known. The dynamic programing (DP) algorithm (Fang and L. Elefteriadou 2004) was only devoted to the adaptive signal operation of a diamond interchange. Beside the surface street traffic management, ramp-metering has become popular (Muhurdarevic et al. 2006) to reduce congestion on the motorway and allow street traffic to join the motorway more safely. A field trial in New Zealand in 2005 showed that motorway traffic throughput (network wide) and travel speed increased significantly within the vicinity of the ramp-metering site (Brown et al. 2005). At an interchange, however, the intersection signals and the ramp-metering signals are mainly separately controlled. An integrated algorithm for both the signals was developed for diamond interchange and simulated in VISSIM. (Tian et al. 2002, Tian 2007).

1.3 FUZZY LOGIC SIGNAL CONTROL

Human decision making and reasoning in general and in traffic and transportation in particular, are characterized by a generally good performance. Even if the decision-makers have incomplete information, and key decision attributes are imprecisely or ambiguously specified, or not specified at all, and the decision-making objectives are unclear, the efficiency of human decision making is unprecedented. As another example, the most advanced central management systems rarely outperform the traffic operators: although the operators control network traffic mainly based on their expert knowledge and experience, they seem to be able to efficiently control traffic, using qualitative and vague information about the traffic conditions. Although a driver can only very roughly estimate the gap to the preceding vehicle, he is still able to maintain a sufficient distance in most circumstances. Crucial in all these cases is the ability of humans to reason using vague variables and their ability to reconcile different conflicting objectives, although they may be slow, sub-

optimal, and inconsistent in their decision-making. Fuzzy logics are approaches that are much closer to real human observation, reasoning and decision making than other (traditional) approaches, such as ALINEA, Green-wave algorithms. These fuzzy approaches have been applied successfully in a wide range of industrial processes (cement kilns, incineration processes, waste water treatment) and products (e.g. camera's digital image processing, washing machines). Applications in the field of traffic engineering have only recently emerged in larger numbers, and in many cases seem very promising. Most of these applications have an experimental and preliminary nature, whereas real-life applications of fuzzy sets and fuzzy logic in the field of traffic engineering are rare. There also have been studies on the fuzzy ramp-metering and fuzzy logic signal control for an intersection. Bogenberger et al. (2000) described a fuzzy traffic responsive ramp-metering that can automatically adjust the ramp-metering rate based on the current traffic conditions, detected by loop detectors. Pappis and Mamdani (Pappis and Mamdani 1977) presented a fuzzy logic control (FLC) for a traffic junction of two one-way streets. The input variables are the passed time of the current interval, the number of vehicles crossing an intersection during the green phase, and the length of queuing from the red direction. The extension time inferred by the FLC is the output. Hoyer and Jumar (1994) developed a FLC that controls an intersection involving 12 main directional traffic flows using 10 inputs and 2 outputs. The inputs are traffic density of different lanes and elapsed time since the last state change. The outputs are the extension time of the current phase and the selection of the next phase.

Adaptive signal control based on fuzzy logic control (FLC) determines the duration and sequence that traffic signal should stay in a certain state, before switching to the next state (Trabia et al. 1999, Pham 2013). The amount of arriving and waiting vehicles are quantized into fuzzy variables and fuzzy rules are used to determine if the duration of the current state should be extended. The fuzzy logic controller showed to be more flexible than fixed controllers and vehicle actuated controllers, allowing traffic to flow more smoothly.

1.4 PROBABILISTIC FUZZY LOGIC SIGNAL CONTROL

The FLC does not possess the ability to handle various uncertainties especially in real world traffic control. Therefore it is not best suited for stochastic nature problems such as

traffic signal timing optimization. However, probabilistic logic is the best choice to handle the uncertainties containing both stochastic and fuzzy features (Pappis and Mamdani 1977). For a typical traffic light control system, the variables considered as stochastic is the arrival traffic. The variability of traffic demand is inherent to the traffic system. This is due to the random nature arrival at an intersection and also induces the stochastic variability into the degree of saturation. There have been research works (Colubi et al. 2003, Liang and Song 1996) on the relationship of randomness and fuzziness. Meghdadi and Akbarzadeh-T studied the integration probability theory and fuzzy logic (Meghdadi and Akbarzadeh-T 1996). Recently, Liu and Li (2005) developed a systematic framework and methodology of probabilistic fuzzy logic system for robotic system and this methodology can be applied into various fields such as signal processing, power system, networking etc.

1.5 THE IMPORTANCE OF BICYCLE SIGNAL CONTROL AND SAFETY ENHANCEMENT AT A DIAMOND INTERCHANGE

Today, in some European cities – such as Amsterdam or Copenhagen – two-thirds of all road users are cyclists. In other words, it is perfectly feasible for a majority in a metropolis to ride a bike and not travel by car. Not everybody can ride a bike every day, however, which is why the bike should not be seen as a competitor, but rather as complementary to public transport. Especially on the way to and from work there is a lot of potential. Just to compare: in Germany 35 percent of commuters use a bike on their way to work, but as yet far less than 1 percent do so in New Zealand. In recent years, local councils invested heavily in building and upgrading the infrastructure to encourage people moving from driving to cycling to work, bus station. So clearly there is a need to study this mode of transport and integrate them into the operation of the diamond interchange. Looking at a broader image, creating a more efficient signal control for cyclists not only improves the bicycle flow rate but also increases the safety of the cyclists when traveling across the interchange. In long term, there will be more cyclists on the road therfore fewer cars and other private mode of transport. This will eventually decrease the congestion pressure during peak periods.

1.6 OBJECTIVES OF THE THESIS

This study aims at developing a methodology and implementing the algorithm to provide adaptive signal control of diamond interchanges using Probabilistic Fuzzy Logic (PFL). To achieve this, two major objectives have been identified:

- To develop a methodology for PFL signal control at a diamond interchange and the implementation of the algorithm

The methodology will be based on PFL because it can handle various uncertainties therefore PFL best suited for signal timing optimization. The proposed PFL methodology in this study will be structured and formulated to suit diamond interchange timing plans and will in turn lead to an efficient traffic signal plan. The framework and procedure to implement the proposed PFL algorithm will also be developed as part of this thesis.

- To evaluate the proposed PFL signal control for diamond interchanges using micro-simulation

A fully calibrated diamond interchange model will be built at the microscopic level using Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network (AIMSUN). This model is used to simulate the proposed algorithm and the signal plans from other models with various traffic scenarios and operational conditions. The proposed PFL algorithm will be evaluated and compared with other adaptive signal models using various performance measures.

- To develop a Probabilistic Fuzzy Bicycle Signal (PFBS) for diamond interchanges using micro-simulation and enhance cyclist safety during movement across the interchange.

The proposed PFBS will be structured, formulated and evaluated using a microscopic simulator. The bicycle lane planning, signal design and integration procedure into a diamond interchange will also be developed as part of this thesis.

The primary output of this thesis will be a new adaptive signal control methodology for a typical diamond interchange using Probabilistic Fuzzy Logic. The secondary output is the corresponding implementation procedure and algorithms. The final output of this study is a new methodology for cyclist signal control not only improving the traffic flow but also reducing the cyclist accidents.

1.7 THESIS OVERVIEW

A critical review on intersection traffic signal control, other adaptive diamond interchange signal control, cyclist safety, policy making and planning is included in Chapter 2. Chapter 3 presents the proposed PFL methodology for adaptive signal control of diamond interchanges and the model for green time extension along with phase selection logic. This is followed by a discussion on the algorithm implementation and the evaluation of the algorithm using simulation and comparing the performance of PFL algorithm with other models under various demand scenarios. Chapter 4 looks into cyclist safety and awareness at traffic intersection. The bicycle traffic design and planning is also discussed in this chapter. Chapter 5 presents the proposed PFBS methodology for diamond interchanges integrated with the cyclist signal control. The performance of PFBS is evaluated and analysed. Chapter 6 summarises the contributions, findings and conclusion drawn from the research.

CHAPTER 2

LITERATURE REVIEW

2.1 TRAFFIC SIGNAL CONTROL FUNDAMENTAL

Interchanges are designed to improve traffic flow in urban areas; they are carefully designed to enable the smooth flow of traffic from arterial streets to and from the highways. There are many types of interchanges and the diamond interchange is one of the most well-known interchanges. A diamond interchange consists of two closely spaced intersections, being less than 250 meters apart and each connecting directly to a highway direction (Fang 2004). The short distance between the two intersections limits the storage available for queued vehicles on the arterial cross street. The signal timing is very important in controlling the traffic flow for the intersection during the peak hours. If the signal timing is set improperly, the queue from the downstream intersection will spill back and block the upstream approaches (Fang 2004). One of the known effects of the diamond interchange is called demand starvation which occurs when portions of the green at the downstream intersection are not used because conditions prevent vehicles at the upstream intersection from reaching the downstream stop-line (HCM 2000). Recently, a fuzzy logic control for a diamond interchange (FLDI) was developed to improve the coordination between ramp metering and the two signals of intersections (Pham 2013).

Traffic signal control in practice includes pre-timed, actuated and adaptive control. Pre-timed control operates based on fixed intervals and phase timings with fixed number of Bicycle length and phase sequence. The disadvantage of this type of control is that they become less effective when traffic condition changes substantially. Actuated control which is based on the gap –seek logic to response to traffic fluctuations during the day is more sophisticated than the pre-timed control. The controller extends the green beyond a minimum time until a gap in the traffic flow on the approaches currently with green has been detected or until the pre-defined maximum green time has been reached. Actuated signal control has certain limitations including tendency to extend green inefficiently under low traffic flow conditions, and great sensitivity to incorrectly set maximum green times

(Fang 2004).Adaptive control can be defined as any signal control strategy that can adjust signal operations in response to fluctuating traffic demand in real time according to certain criteria (Lin and Vijayayumar 1988). Adaptive control makes signal timing decisions based on detected or identified current and short-term or even long-term future flow. Since Miller's pioneer work (Miller 1963), considerable research has been done to develop adaptive systems. Systems such as SCOOT (Hunt 1982), SCATS (Luk 1984, Charles 2001), OPAC (Gartner 1983, Gartner and Pooran 2002) and RHODES (Mirchandani and Head 2001) are among the best known. They are SCOOT - Split, Bicycle and Offset Optimization Technique developed by Transportation and Road Research Laboratory (TRRL), UK (Hunt, et al, 1982), and SCATS – Sydney Coordinated Adaptive Traffic System developed by New South Wales Roads and Traffic Authority (RTA) (Luk, 1984; Charles, 2001). Both systems attempt to select the best parameters to maintain a pre-defined degree of saturation for the critical intersection in the network (Fang, 2004). Across New Zealand, more than 500 traffic signals are coordinated and managed with SCATS.

Based on the predictions using the data from detectors upstream, the SCOOT program firstly establishes and models the development of queues of vehicles at the stop-line and so adjusts phase lengths and offsets to minimize delay. The split optimizer selects the phase that will be closest to achieving a 90 percent degree of saturation without changing the phase starting time by more than 4 seconds. The offset optimizer changes the offset of an intersection from its adjacent intersection once every Bicycle. The demand profiles obtained from the upstream and downstream of each intersection are used to assess whether a change in offset would minimize the overall disutility of delay for that Bicycle. The Bicycle length optimizer tries to maintain the degree of saturation for the heaviest intersection at 90 percent. A change in Bicycle length can only be made in every 5 minutes. The optimization model used by SCOOT is the hill-climbing procedure used in TRANSYT- 7F, which is a network signal timing optimization program developed by TRRL. The hill-climbing procedure makes changes to signal timings (such as splits and offsets) and determines whether or not the performance index (a combination of stops and delay, fuel consumption or operating cost) is improved. The hill-climbing is an iterative, gradient search technique. Firstly an initial signal timing plan and the initial performance index is given. Then an optimized parameter (i.e., offset) is increased by one pre-specified

"step size". The new performance index is compared with the previous value: if the new performance index is less than the previous value, the program continues to increase the offset by the same amount, as long as the performance index continues to decrease; if the new performance index is greater than the previous value, the program will decrease the offset by the same amount and continue to decrease the offset by this amount as long as the performance index continues to decrease. The above procedure will be repeated for all pre-specified optimization "step size". The disadvantage of the hill-climbing procedure is that it still requires extensive numerical computations, and it provides no guarantee that the global solution will always be found (Fang, 2004).

Sydney Coordinated Adaptive Traffic System (SCATS) uses information from vehicle detectors located in each lane ahead of the stop bar to adjust signal timing in response to variations in traffic demand and system capacity. It determines suitable signal timing based on average traffic conditions. It anticipates the arrival of vehicles at a certain intersection from traffic data collected at the preceding intersection and responds accordingly. Flow and occupancy are calculated during the green phase on each intersection approach. This information is transmitted to a control centre where SCATS attempts to maintain a user-specified degree of saturation on the intersection downstream of the collected traffic data. Signal phases can be set to the equalize degree of saturation on all approaches or they can be arranged to give priority to a particular direction (Pearce, 2001).

The dynamic programing (DP) algorithm (Fang and L. Elefteriadou 2004) was only devoted to the adaptive signal operation of a diamond interchange. Apart from surface street traffic management, ramp-metering has become a popular method (Muhurdarevic et al. 2006) to reduce congestion on the motorway and to allow the street traffic to join the motorway more safely. A field trial in New Zealand in 2005 showed that motorway traffic throughput (network wide) and travel speed increased significantly within the vicinity of the ramp-metering site (Brown et al. 2005). At an interchange, however, the intersection signals and the ramp-metering signals are mainly separately controlled. An integrated algorithm for both the signals was developed for diamond interchange and simulated in VISSIM. (Tian et al. 2002, Tian 2007).

2.2 FUZZY CONTROL IN TRAFFIC MANAGEMENT

Human decision making and reasoning in general and in traffic and transportation in particular, are characterized by a generally good performance. Even if the decision-makers have incomplete information, and key decision attributes are imprecisely or ambiguously specified, or not specified at all, and the decision-making objectives are unclear, the efficiency of human decision making is unprecedented. The most advanced central management systems rarely outperform the traffic operators. Although the operators control network traffic mainly based on their expert knowledge and experience, they seem to be able to efficiently control traffic, using qualitative and vague information about the traffic conditions. Although a driver can only very roughly estimate the gap to the preceding vehicle, he is still able to maintain a sufficient distance in most circumstances. Even though humans may be slow, sub-optimal, and inconsistent in their decision-making, their ability to reason using vague variables and their ability to reconcile different conflicting objectives is quite remarkable. Fuzzy logics are approaches that are much closer to real human observation, reasoning and decision making than other (traditional) approaches, such as ALINEA, Green-wave algorithms. These fuzzy approaches have been applied successfully in a wide range of industrial processes (cement kilns, incineration processes, and waste water treatment) and products (e.g. cameras, washing machines). Applications in the field of traffic engineering have only recently emerged in larger numbers, and in many cases seem quite promising. Most of these applications are experimental and are often rudimentary. Real-life applications of fuzzy sets and fuzzy logic in the field of traffic engineering are rare. There have been studies on the fuzzy ramp-metering and fuzzy logic signal control for an intersection. Bogenberger et al. (2000) described a fuzzy traffic responsive ramp-metering that can automatically adjust the ramp-metering rate based on the current traffic conditions, detected by loop detectors. Pappis and Mamdani (1977) presented a fuzzy logic control (FLC) for a traffic junction of two one-way streets. The input variables are the passed time of the current interval, the number of vehicles crossing an intersection during the green phase, and the length of queuing from the red direction. The extension time inferred by the FLC is the output. Hoyer and Jumar (1994) developed a FLC that controls an intersection involving 12 main directional traffic flows using 10 inputs and 2 outputs. The inputs are traffic density of different lanes and elapsed time since the

last state change. The outputs are the extension time of the current phase and the selection of the next phase.

Adaptive signal control based on fuzzy logic control (FLC) determines the duration and the sequence that traffic signal should stay in a certain state, before switching to the next state (Trabia et al. 1999, Pham 2013). The amount of arriving and waiting vehicles are quantized into fuzzy variables and fuzzy rules are used to determine if the duration of the current state should be extended. The fuzzy logic controller has been shown to be more flexible than the fixed controllers and the vehicle actuated controllers, allowing traffic to flow more smoothly.

The FLC does not possess the ability to handle various uncertainties especially in real world traffic control. Therefore it is not suitable for stochastic problems such as traffic signal timing optimization. Probabilistic logic is better equipped to handle the uncertainties containing both stochastic and fuzzy features (Pappis and Mamdani 1977). For a typical traffic light control system, the traffic demands are stochastic in nature. The variability of traffic demand is inherent to the traffic system. This is due to the random nature of arrival at an intersection which also induces stochastic variability into the degree of saturation. There have been research works (Colubi et al. 2003, Liang and Song 1996) on the relationship of randomness and fuzziness. Meghdadi and Akbarzadeh-T (1996) investigated the integration of probability theory and fuzzy logic. Recently, Liu and Li (2005) developed a systematic framework and methodology of probabilistic fuzzy logic system for robotic system and this methodology can be applied to various fields such as signal processing, power system, networking etc.

2.3 BICYCLE LITERATURE REVIEW

This literature review covers three main aspects of bicycle research which are planning, policy making, bicycle modelling and lastly bicycle signal.

It is generally understood from a study of cyclists' behaviour that not everyone complies with traffic laws. Accidents involving cyclists in crosswalks at intersection including when crossing freeway interchanges are a common cause of road fatalities. In some urban areas the cyclist fatalities are up to 40 to 50 percent of all the traffic fatalities.

According to the National Highway Transportation Safety Administration Safety facts for 2007 it is reported that 4,464 cyclists were killed in the United States and another 70,000 were injured in traffic crashes. On average, a cyclists is killed in a traffic crash every 113 minutes and injured in a traffic crash every 8 minutes. There are a number of factors influencing the crossing behaviour of cyclists like crossing speed, gap acceptance and signal compliance with relation to age and gender. Also it was found that bicycle speed varies by age, gender, distance and group size and that cyclists tended to walk faster after longer wait times. Also most cyclist fatalities occurred at night between 6PM and 6AM (64%). Currently there are different traditional facilities and treatments that are installed at the intersections for providing a safer crossing. Bicycle signals and signs are basic facilities applied for bicycle safely crossings. Some of the treatments already in use are high visibility signs, in road signs, coloured pavements, overpass, underpass, bicycle boulevards etc., but these conventional methods are not sufficient to provide safer crossings to the intersections.

In Japan and some other countries such as The Netherland, United States, cyclists and cyclists are treated as a single mode. In these countries bicycle roads have been constructed in urban and suburban areas to accommodate recreational uses. In Europe and China, bicycle is considered as a mode of transportation while other countries such as New Zealand, United States, Japan and Australia bicycle is used for recreational. Most bicycle roads around the world are constructed as Bicycle and cyclist roads known as shared-use paths. In United States, other countries including New Zealand and Japan bicycles are permitted on roads. With the increasing of fuel cost and population in major cities, governments and local authorities are paying attention to the Bicycle policy and infrastructures because its use can reduce the automobile usage and relieve environmental problems. In New Zealand, Auckland council has allocated funding for improvements for cycling in Auckland region. According to the Gravitas's report, in 2007 over $100 million was planned to be invested in building over 50% of the Regional Bicycle Network by 2015. By mid-2009, 21% of the Regional Bicycle Network had been built across North Shore, Auckland CBD and Waitaikere. On the other hand, taking Japan as an example, huge number of Bicycles left around railway, bus stations cause serious congestion. Therefore, some decentralization of Bicycle feeder demand should be considered. If adequate road network plans are introduced along with optimised Bicycle signal controls at intersections,

the centralization problem will be solved. New Zealand Transport Strategy (NZTS) released in December 2002 supports cycling and the provision of cycling infrastructure*:*

"The negative social and environmental impacts of transport must be reduced. In land transport, the government is determined to see that the transport system supports access and environmental outcomes through improving public transport, reducing congestion, improving safety for all, supporting alternatives to travel (such as teleworking and local provision of services), and providing infrastructure for walking and cycling."(GTEP14, 2008)

According to GTEP14 (2008), initiatives that support the NZTS and the Road Safety Strategy 2010 are:

- Land Transport Safety Authority prepared a Bicycle Network and Route Planning Guide that was released in 2004. It provides guidance on the planning of cycling facilities.
- Land Transport NZ is supporting the development of cycling strategies by requiring road controlling authorities to have a cycling strategy as a condition of its financial support for individual cycling projects.
- Road controlling authorities throughout New Zealand have or are preparing cycling strategies for their jurisdictions.

2.3.1 Policy

Along with walking, cycling is an integral part of the transportation network. It is an environmentally sustainable mode of travel, and is also one of the most common forms of transportation. Research shows that 15% of people regularly cycling and a further 32% seriously think about cycling. In a nationwide study, potential cyclists strongly stated that they wanted to travel separately from motor vehicles and to be able to cross safely at intersections. Improving the visibility of cycling and providing good cycling facilities is needed to achieve high levels of cycling.

Encouraging a greater number of people to consider cycling as part of their daily routine can lead to benefits for public health and can lead to a substantial reduction in number of vehicle trips made over short distances and therefore reduce the pollution in

urban city. The New Zealand Transport Strategy 2008 includes several components that are relevant for cycling planning. The most important of these is the target of increasing walking, cycling and other modes of activities from 18% to 30% of the total trips in urban areas. And of the way of improving the desirability of cycling trips is to reduce delays created by traffic signals which improve the safety of the cyclist. This is because the primary concern regarding high level of delays for cyclists is the safety issues that result from frustrated ignoring traffic signals and making their own gap-acceptance judgements. The research has found that it is possible to achieve this without significant capital cost and, in many cases, without adversely affecting other mode choices.

New Zealand has tended to favour vehicles with cyclists/cyclists often considered only in terms of delay to vehicles and as an impediment to intersection efficiency, and cycling capacities often being marginalised.

The New Zealand 2009 Government Policy Statement on Land Transport Funding (GPS) sets out the level of funding for walking/cycling facilities, and included the following short to medium term impacts:

- *Improvements in the provision of infrastructure and services that enhance transport efficiency and lower the cost of transportation through better use of existing transport capacity.*
- *Reductions in deaths and serious injuries as a result of road crashes.*
- *More transport choices, particularly for those with limited access to a car, where appropriate.*

In 2012-2013, the GPS scopes still remain the same with few a amendments such as:

- *Investment in walking and cycling is also expected to make a contribution to economic growth and productivity. To achieve this, funding should be directed to reducing congestion and/or improving cyclist and cyclist safety.*
- *Roading programmes could, where appropriate, include provision for treatments that improve cycling safety on roads that are part of the Bicycle trails network*

Much of the investment that goes into walking and cycling projects occurs at a local government level, coordinated by regional councils and regional land transport strategies. Local government is also responsible for operating traffic signals and maintaining the (non-state highway) road network. Recent regional land transport strategies, and regional and local government policies, have tended to place increasing emphasis on stimulating active

transport modes and integrating modes. At both central and local government levels, there is a desire to implement improvements without significant additional cost. The most effective means of keeping costs down is to look for increased performance efficiencies and purely operational changes, rather than expensive infrastructure investments. Making the most of existing infrastructure is a cost-effective approach, and a key aim of the GPS 2009. The optimisation of traffic signals is an area that could be given more policy and funding emphasis and achieve results with relatively little cost.

2.3.2 Safety and Compliance

2.3.2.1 Safety

The relationship between cyclist delay and cyclist safety is a complex one. Cyclists who feel they are faced with unreasonable delays will use their own judgement as to when it is safe to cross. However, an increase in delay does not necessarily result in an increase in non-compliance, as high volumes of traffic can act as a deterrent to non-compliance at signalised intersections. Conversely, low cyclist waiting times do not guarantee universal compliance with traffic signals. It is therefore difficult to draw a linear relationship between delay and resulting non-compliance. It is also difficult to find an exact correlation between non-compliance and injury, as cyclists unlawfully crossing the road will use their judgement to determine a safe crossing, and it is only when that judgement is in error that an injury or fatality might occur. Only a small portion of non-compliant activity result in an injury or fatality. However, as traffic volumes increase, any non-compliant behaviour becomes inherently riskier.

2.3.2.1 Compliance

The Manual on uniform traffic control devices (Federal Highway Administration 2009) observes that traffic control signals are often considered a ‘panacea’ for all traffic problems at intersections; however, simply installing signals does not guarantee a favourable outcome, particularly for low vehicle volumes. Cyclist signals, if poorly operated, can create unnecessary delay. Various studies have found that cyclists are more flexible in their

regard for road rules than other mode types. Cyclists tend to use traffic signals as a guide, but if they become frustrated by long delays, they will likely ignore the signals entirely and cross when they perceive the risk to be acceptable, rather than accept continued delay. Thus, cyclist signals have a higher non-compliance rate than vehicle traffic signals (and potentially, a much lower enforcement rate). Therefore, it is possible to infer that the primary measure of whether a set of signals is functioning adequately for cyclist traffic would be the rate of non-compliance. Non-compliance to traffic signals presents a risk to the cyclists and other road users, and as a result, frustration at cyclist delay quite quickly translates into a road safety issue. Much of the literature reviewed considered cyclist delay entirely from a compliance/safety perspective, rather than as a factor in overall cyclist travel times. Noland and Ishaque (2007) found that:

Cyclist non-compliance behaviour is encouraged by signal timings that are not favourable to them. This is the case both when a disproportionately large amount of time is made available to vehicular traffic and when cyclist volumes are such that they do not fit into the time provided for by the cyclist phase. Long signal Bicycles may pose a safety hazard for cyclists and therefore one of the most effective measures to increase cyclist safety and compliance is to make traffic signals as good as possible for cyclists and that is by minimising their waiting times.

They also reported on several European studies on the topic of cyclist perception of delay. One two-year study in London found that at controlled crossings, 30–40% of cyclists felt annoyed when the delay was in the range of 6–22 seconds, but more than 70% felt annoyed when the delay was longer than 26 seconds. Another study on children and adults showed that 30 seconds was the maximum that both children and adults were willing to wait at a signalised intersection, and similar findings in Germany had led to the German Highway capacity manual (Institute for Traffic Engineering, Ruhr-University 2000) specifically recommending that signal Bicycle times longer than 90 seconds should be avoided. Ishaque and Noland found that cyclist non-compliance was encouraged by timings that were unfavourable to them, and that long Bicycle times could therefore pose a safety hazard that could be avoided through reducing signalised delays. This is significant in New Zealand, because the intersections observed, had Bicycle times longer than 90 seconds, and average cyclist delays of more than 30 seconds.

The UK Traffic advisory leaflet 5/05 (Department for Transport 2005) also raises the issue of signal compliance and reasons for some cyclists being more willing to take risks:

Cyclists compliance with the Red signal is thought to be generally poor. Cyclists are more likely to disregard the Red signal if they consider the distance they have to cross, or the time they have to wait, is unreasonable. (When waiting at a junction in bad weather, a driver may be frustrated but is generally warm and dry. A frustrated, cold and/or wet cyclist is more likely to take what otherwise they would consider an unacceptable risk).

Part 7 in the Ausroads (1994) identifies that drivers and cyclists will disobey a red signal if delays are abnormally long. The guidelines suggested that a maximum waiting time of 20 seconds would be tolerated in light traffic, and 120 seconds in heavy traffic. This is much longer than the time suggested in literature from other sources, and longer than what was considered acceptable by cyclists interviewed during this research. As a general rule, signals engineers try to aim for a total Bicycle of 120 seconds or less, where possible.

Australian research conducted by Daff et al (1991) showed that heavier traffic flows in more recent times seem to have forced more compliance at intersection signals, though this is likely to be due to perceived safety issues rather than a greater desire to comply with signalised delays. In an effort to improve compliance, some signal operators have introduced countdown timers. The literature reviewed universally found that excessive delay would lead to cyclist frustration.

High traffic volumes reduce the risk of non-compliant behaviour, as the perception of risk is greater. Where volumes are low and delays are long (e.g. outside of vehicle 'peak' periods) then cyclists are more likely to ignore the signals. The dangers associated with this can be aggravated by a number of factors, including:

- The visibility of cyclists versus other visual distractions (such as oncoming traffic)
- The presence of heavy vehicles, which can have significant blind spots in their field of vision and also are harder to slow down than cars, and may therefore be less able to avoid cyclist than the cyclist expects).

It is important, therefore, for road controlling authorities to consider cyclist delay, and in particular to consider off-peak conditions when setting up signal phasing and timing.

2.3.3 Bicycle Operation Fundamental

Miller and Ramey (1975) came up with the fundamental diagram of bicycle operation (Figure 2.1) in their comprehensive study about bicycle flow. They used radar gun to measure the speed and the bicycle flow is obtained by counting the bicycles on a selection bike's paths around Davis and American River in Sacramento, USA. The density is then calculated by using the following equation (1):

$$k = \frac{q}{u - width} \quad (1)$$

where:

q is flow rate (bikers per hour)

u is average speed (miles per hour)

k is density (bikes per feet^2)

$width$ is path width (ft)

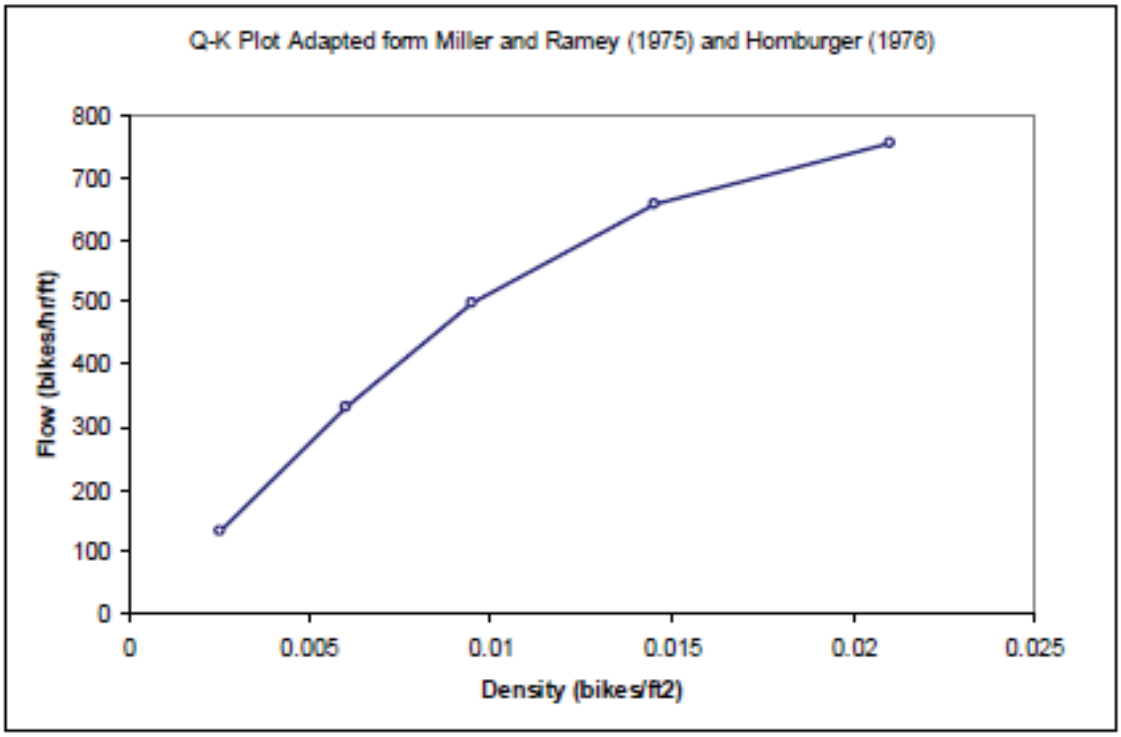


Figure 2.1 Fundamental diagram for bicycle flow (Miller and Ramey, 1975)

2.3.4 Cyclist Crash Prediction Modelling

A substantial amount of work has been undertaken, both in New Zealand and throughout the world, in understanding road safety issues for cyclists. The most recent cyclist/cyclist safety research in New Zealand can be found in Land Transport NZ report 389: Predicting accident rates for cyclists and cyclists (Turner et al 2009). An interesting finding of this research was a comparison between cyclist and cycling incidents reported to the official Crash Analysis System (CAS) database, and those recorded in the databases of the Accident Compensation Corporation (ACC) and St John Ambulance. This comparison found that there was significant under-reporting of cyclist and cycling incidents in CAS. This has direct implications for safety research and also for developing day-to-day engineering solutions to safety issues. There is a need to consider this under-reporting while developing crash prediction models.

Turner (2000) developed a series of APMs for accidents at intersections using generalised linear models. At traffic signals for example, he developed four models, to describe the four major accident types. These were developed for each of the three main centres in New Zealand, Auckland, Wellington and Christchurch over six time periods. Turner (2000) produced models that allow accidents to be predicted from traffic volumes. The modelling was produced using macros that were developed for the statistical package, Minitab. Models were developed using accident data from throughout New Zealand for a five-year period, where motor vehicles were involved. The typical Turner model took the form (2):

$$A = b_0 Q_a^{b1} Q_b^{b2} \tag{2}$$

where:

A is accidents in five years;

Q_A and Q_B are approach flows;

$b0$, $b1$ and $b2$ are parameters.

Turner et al (2005) also found that the people most likely to be injured on New Zealand roads were those aged 10–20 years and the elderly. In the case of those aged 10–

20, the number of incidents, although high, matched the percentage of cyclist trips by this age group. The high rate of incidents could therefore be attributable to the over-representation of this age group as cyclists, rather than any particular behavioural trait in this age group. People over 80 were the largest group of cyclist casualties per kilometre travelled. Contributing factors included the likelihood of reduced mobility, and the increased seriousness of their injuries meaning the injury was more likely to be reported. The crash prediction models developed by Turner et al (2005) considered traffic signals and commercial midblock sites. International research reached similar conclusions on a 'safety in numbers' effect. Essentially, they found that an increase in cyclists (or cyclists) does not result in a linear increase in crashes. Although there may be more injuries, in absolute terms, an increase in cyclist/cyclist numbers appears to lead to an improvement in safety on a per-cyclist/per-cyclist basis.

Roozenberg and Turner (2004) developed two models for cyclist–motor vehicle conflicts: one for cyclists travelling parallel to the flow of traffic and crashing with stationary or parallel moving vehicles, and one for cyclists crashing with turning vehicles. Both models had a pronounced 'safety in numbers' effect in that as the number of cyclists increased the crash rate per cyclist decreased. Turner et al.'s following research (2009) added a variable for Bicycle lanes and found that their presence increased the crash rate for parallel crashes. This crash model was developed over only 21 intersections, and further research is currently being carried out in this area across both Australia and New Zealand.

This safety in numbers effect is consistent with other research, such as Jacobsen (2003), who explored several aspects of the safety in numbers effect for walking and cycling. Jacobsen found that cities with a higher walking or cycling mode share had comparatively fewer accidents per kilometre travelled. Jacobsen also found the correlation worked with both increasing and decreasing mode shares, with a reduction in walking or cycling kilometres travelled increasing the per-person/per-km risk. Jacobsen concluded that driver behaviour played a strong part in the likelihood of a cyclist or cyclist being injured, and that this aspect was influenced by numbers. Presumably this is because a more visible presence of cyclists or cyclists on the road space leads to greater awareness on the part of motorists of the possible conflicts that may occur.

2.3.5 Bicycle Intersection Control

Past work in the area of intersection control is concentrated around signalised intersections Wachtel et al. (1995) analysed the minimum green time required for cyclists stopped on a red. Several researchers, including Forester (1994) have concluded through analysis of crash data that inadequate clearance intervals (i.e., the combination of the yellow change interval and any all-red clearance interval) exist for bicycles at some intersections. Taylor (1993) and Wachtel et al. (1985) use standard deterministic equations to show that this is the case if the design clearance point (i.e., the point cyclists must reach) for safety is much inside the stop bar. The limitations of this approach include (a) the fact that the location of the design clearance point varies over intersections and is unknown, (b) the lack of data on perception-reaction times, and (c) the small sample of data on which to base the required design deceleration. If a designer decides extra clearance is warranted, Taylor (1970) and Wachtel et al. (1985) suggest various ways to provide it and various ways to convey the signal change warning to cyclists while limiting impacts to automobiles.

Taylor (1970) developed a probability-of-stopping model from roadside observations of cyclists' behaviour at the onset of yellow and found several factors that affected this behaviour. He shows how such models provide insights for the design of clearance intervals and how they can be used to evaluate the effects of different intervals and different methods of providing the signal change warning. He suggests that the main limitation is a lack of observational study of the impacts of clearance intervals longer than those normally provided for automobiles and that the logical next step is such analysis using "before and after" probability-of-stopping models.

2.3.6 TRAFFIC OPERATION BICYCLES AFFECT

Wei et. al. (2003) reported on the interferences of a bicycle flow to the vehicle flow moving in the same movement direction are identified with the following features of the affected motor vehicle platoon:

- Vehicle start reaction time
- Vehicle platoon progression
- Stable vehicle headway (i.e. minimum running headway)

Wei et. al. (2003) believed that in a typical signalised urban intersection, during a red indication, bicycles are stopped behind the stop line forming bike queues in the most

intense manner. When signal turns green, the bicycles begin to move forward, accelerate, and pass the intersection with a very dense bicycle flow. Since its flexible manoeuvre and much quick response to the change in the signal indication, bicyclists are able to start crossing the stop line ahead of motor vehicles. Without physical banned facilities within the intersection and intensive and dense bicycles coming into the intersections, many bicycle riders, especially those want to compete to overtake ahead bicycles, will occupy the fringe area of non-motorized lanes, and even encroach into the motorized right of way nearby. As a result, the "bicycle disturbance areas" are formed, as shown in Figure 9. Vehicles close to this area have to slow down in order to avoid any possible collision with bicycles. We have observed that in most cases, bicyclists occupied the space of disturbance area so quickly that the vehicles have no choice except yielding to the bicycles.

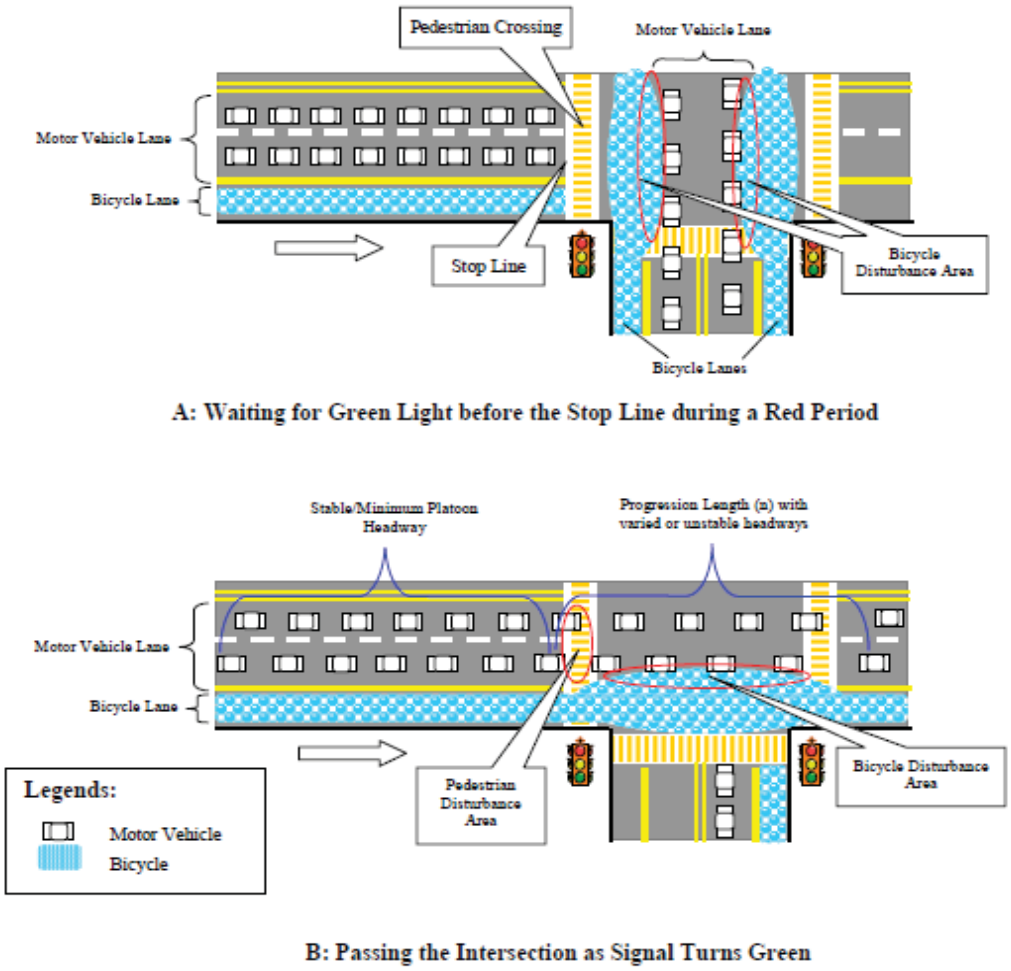


Figure 2.2 Illustration of Bicycle's Affects on Vehicle Traffic Operation in an At-grade Intersection (Wei et. al. 2003)

2.4 SUMMARY

Several fuzzy logic controls have been developed for isolated intersections and multi-phase intersections but there is no fuzzy logic control for Diamond Interchanges. Three-phase and Four-phase plans are used for Diamond Interchanges control; however they are not common in New Zealand and they are only optimised for some situations and may not provide the globally optimum solution (Fang, 2004). On the other hand, these phase plans are based on off-line demand and do not response to real-time demand fluctuation. In contrast, conventional FLC does not possess the ability to handle various uncertainties especially in real world traffic control. Therefore it is not suitable for stochastic problems such as traffic signal timing optimization. Hence, there is a need to develop a Probabilistic Fuzzy Logic control for the entire diamond interchange including the on-ramp.

Most of the fuzzy logic controllers mentioned in this literature review are designed to control the length of the green time according to the traffic conditions. In the traffic lights controller two fuzzy input variables are chosen: the quantity of the traffic on the arrival side and the quantity of traffic on the queuing side. Therefore, based on the current traffic conditions the fuzzy rules can be formulated so that the output of fuzzy controller will extend or not the current green light time. If there is no extension, the state lights will change to another state allowing the traffic from the alternate to flow.

Bicycle movements play a vital role in the daily operation of the diamond interchange. It is important to take into account this mode of transportation in this study. A comprehensive study about bicycle user has been carried out covering a wide range of topics including planning, policy making, fundamental of bicycle operation along with bicycle signal control at intersection and crash prediction models.

CHAPTER 3

PROBABILISTIC FUZZY DIAMOND INTERCHANGE

3.1 METHODOLOGY

3.1.1The Interchange and Detectors Placement

The diamond interchange at the Upper Harbour Interchange in Auckland (Figure 3.1) was chosen to investigate the performance of PFLDI. The distance from north to south of the diamond interchange is 3 kilometers and the distance between the two intersections is 120 meters. Its geometric layout was obtained from a high-resolution JPEG image from Google Earth and its additional information on the widths and number of lanes were supplied by Transit NZ. The interchange is equipped with various detectors for its traffic actuated signal control. The detectors for the intersection signals are placed at 1 m before the stop line and 40 to 60 m away from the stop line for each movement (Figure 3.2). The detectors for ramp-metering are set up as follows: Ramp Queue detector at 70 m from on-ramp entrance, Ramp Check-In detector at 413 m from on-ramp entrance; Up-stream detector at 160 m from on-ramp exit on the motorway; Down-stream detector at 200 m before on-ramp exit; and Interchange Queue Occupancy detector at 40 m away from the stop-line.

Figure 3.1 Aerial view of Upper Harbour Interchange in Auckland (Google Earth)

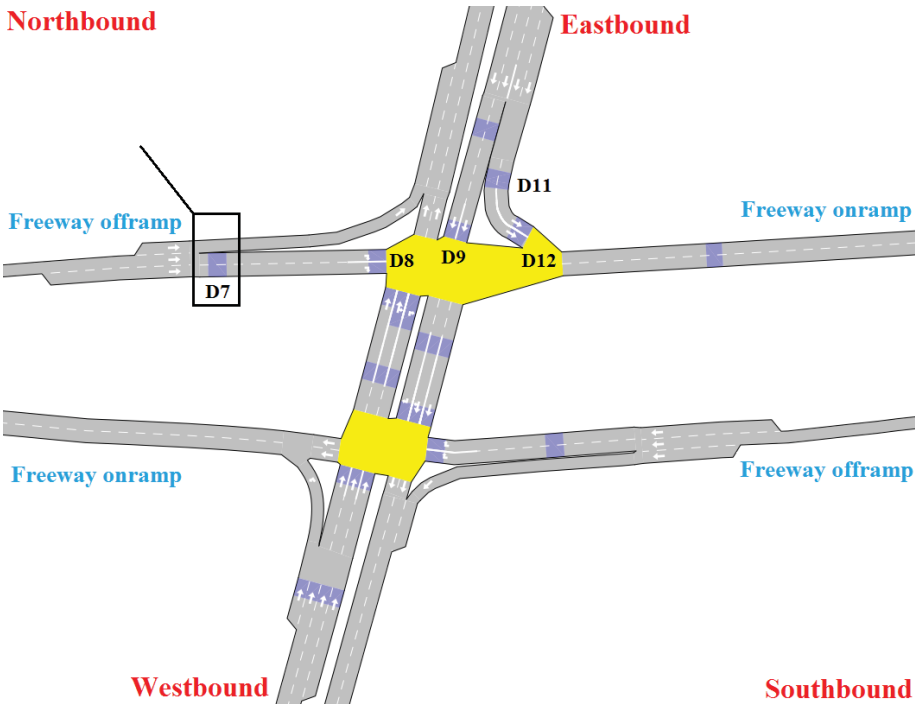


Figure 3.2 Upper Harbour Interchange Detectors

3.1.2 The Interchange Signal and Phasing

At the study site (Upper Harbour Diamond Interchange), the existing traffic signal control reacts to the traffic condition by triggering different pre-defined Bicycle time.

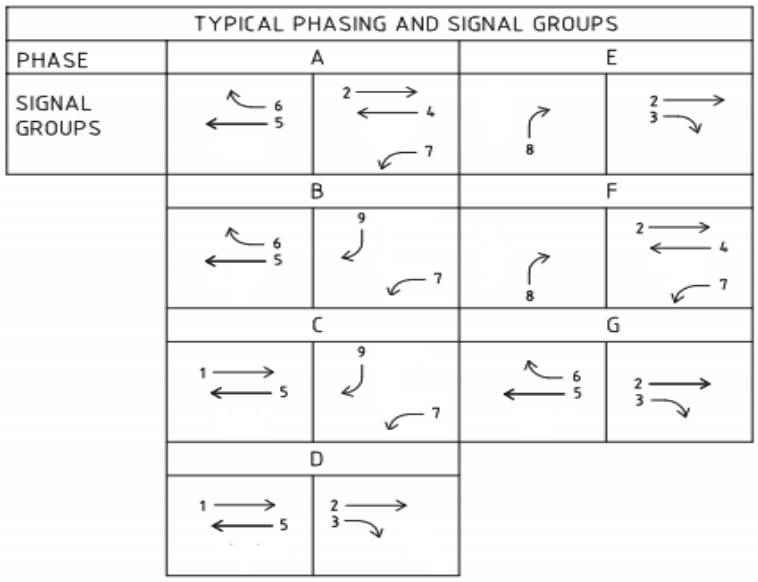


Figure 3.3 Typical Phasing and Signal Groups at Upper Harbour Interchange

The phase sequence (Figure 3.4) for all the current available phases (A, B, C, D, E, F and G) is defined as (Pham et. al., 2013)

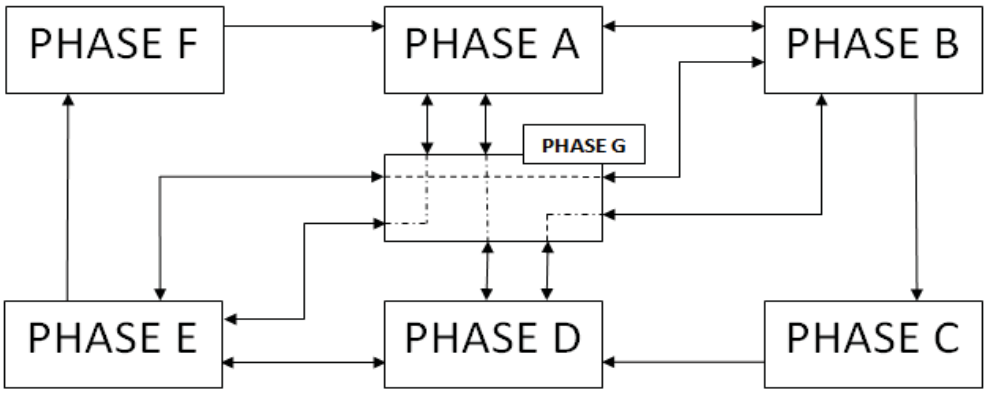


Figure 3.4 Phase Sequence for Upper Harbour Interchange

Running from Phase A

Phase A→Phase B (if there are cars waiting on any of movement 9's detectors) **→Phase C** (if there are cars waiting on any of movement 1's detectors) **→Phase D** (unconditional)

Phase A→Phase G (if there are cars waiting on any movement 8's detectors and no car on any of movement 9's detectors) **→Phase D** (unconditional)

Phase A→Phase G (if there are cars on movement 8's detectors and no car on any of movement 1, 2 and 9's detectors) **→Phase D** (unconditional)

Phase G logic

Phase G is a transitional phase providing mid-block clearance when phases are skipped during a signal Bicycle. This phase effectively provides a flexible early cut-off period for a congested scenario, and a demand for this phase will simultaneously lodge for another phase. Once in Phase G, exit must follow the predefined path. All calls for Phase G are cancelled when the demand conditions are no longer met.

Phasing operation during peak hours

During peak hours from 6:30–9:30 am on weekdays, only three phases (Phase B, F and D) are available and they operate in a fixed sequence, from Phase B to F and to D. Each phase has a green time of 6 s at minimum and, 28, 48 and 17 s at maximum for Phase B, Phase D and Phase F, respectively. Table 3.1 gives the signal timing data used for ADI, FLDI and PFDI in this study

Table 3.1 Signal timing data for ADI, PFLDI and FLDI

Actuated Control			
Phase	Min. Green time (s)	Max. Green time (s)	Yellow (s)
B	6	28	4
D	6	48	4
F	6	17	4
PFLDI and FLDI			
B	6	6-30	4
D	6	6-48	4
F	6	6-18	4

3.1.3 Probabilistic Fuzzy Signal

For a typical traffic signal control system, the variables that are considered as stochastic in nature are effective green time, saturation flow rate and degree of saturation (Kamarajugadda and Park 2003). The traffic demand and conditions are expected to follow a Poisson distribution. The Poisson distribution can be visualised as a limiting form of the binomial distribution, and is also used widely in queuing models (Juang 2000). The Poisson model assumes that the number of sources is finite and the traffic arrival pattern is random. Poisson processes have been used widely in traffic modelling. Probabilistic modelling is a good approximation to real world problem when random uncertainty governs the phenomenon.

Fuzzy logic applies the same tools as probability theory. But it is not able to capture the essential property of meaning (partial knowledge) like probability theory. FLC does not possess the ability to handle various uncertainties especially in real world traffic control. Therefore it is not best suited for stochastic nature problems such as traffic signal timing optimization. As a result probabilistic fuzzy logic (PFL) which combines probability theory with Fuzzy logic is the best choice to handle the uncertainties containing both stochastic and fuzzy features. The detailed concept of probabilistic fuzzy set is defined in (Liu and Li 2005).

In this research on diamond interchange, the arrival of vehicles from one arterial street follows the Poisson distribution (3). For saturated traffic flow, the probability, $P(x)$, that x vehicles arrive at any interval in the two-way intersection is calculated by

$$P(x) = \frac{m^x e^{-m}}{x!} = \frac{I}{x!}\left(\frac{Vt}{3600}\right)^x e^{-\frac{Vt}{3600}} \quad (3)$$

where:

t is the length of time interval in seconds

V is hourly volume (AIMSUN output for our simulation)

m is the average number of vehicles per interval (Vt/3600).

3.1.4 Design of Probabilistic Fuzzy Logic for a Diamond Interchange

The developed PFLDI model consists of three main modules: Phase Selection Logic (PSL), Probabilistic Fuzzy Phase Timing (PFPT) and Probabilistic Fuzzy Ramp-metering (PFRM), as shown in Figure 3.5.

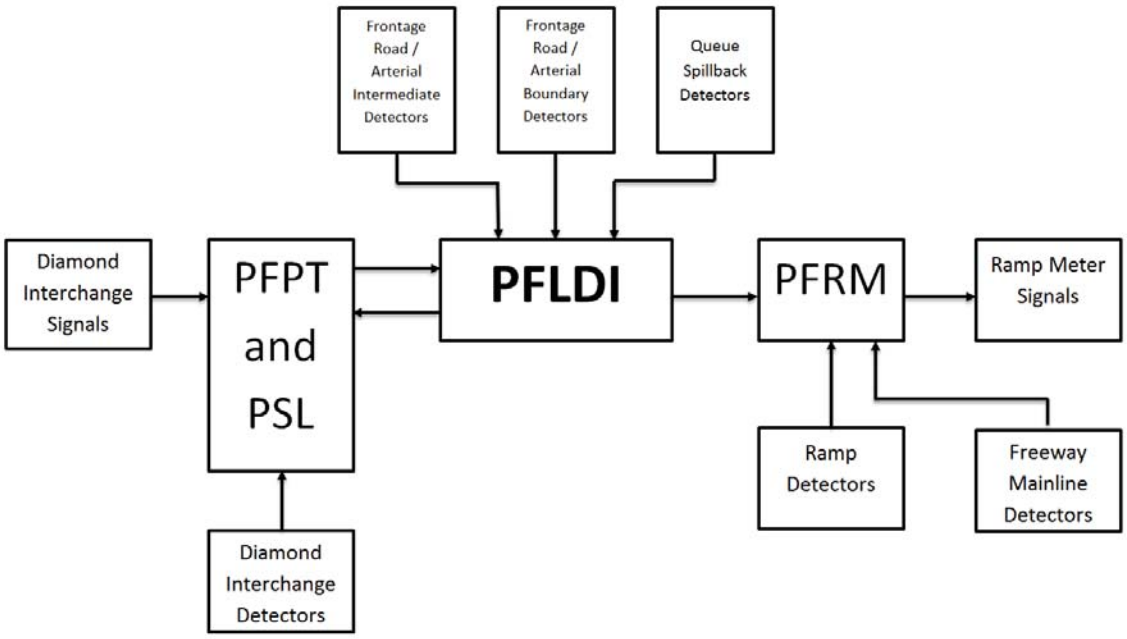


Figure 3.5 PFLDI model illustrations

The framework of the PFLDI algorithm is shown in Figure 3.6.

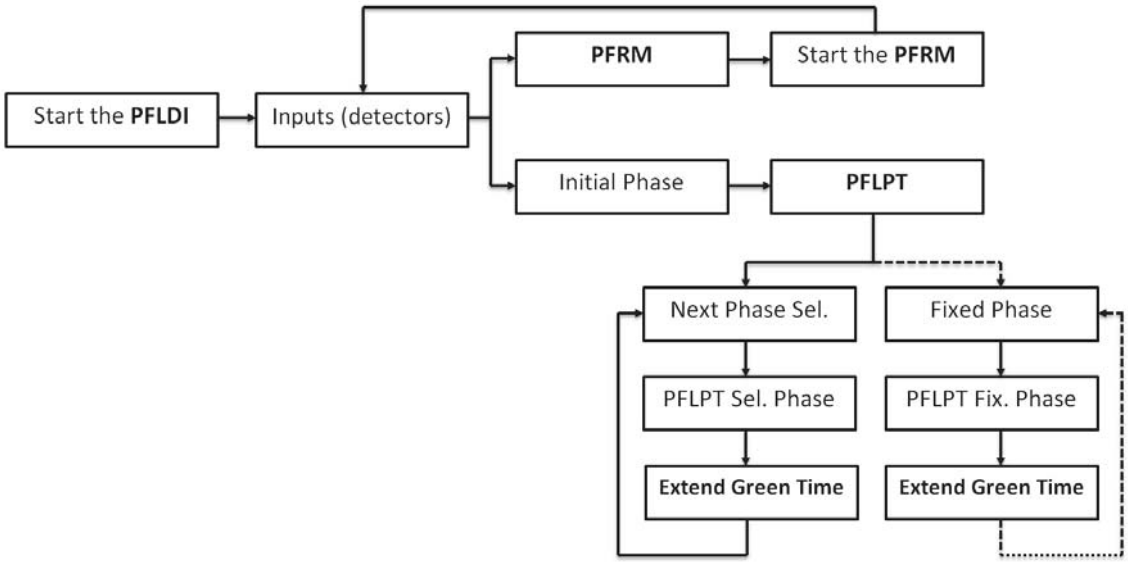


Figure 3.6 The overall framework of the PFLDI algorithm

The historical data, traffic plan, and design layout of the specified diamond interchange are based on the information provided by New Zealand Transportation Agency (NZTA) and our site surveys. The detectors can record vehicle's speed, vehicle types, occupancy, headway, traffic flow, etc. and feed the data into the SCATS and Traffic Management System. Traffic Engineers can extract this information from SCATS at any time for any interval. Fuzzification is a process that converts each numerical (analogue) input into a set of degrees of membership by membership functions. These input data are acquired by numerous detectors on the interchange, ramp and freeway. These detectors were installed and managed by NZTA.

a. The upstream detector is designed to collect information including local speed, local traffic flow and local occupancy of the mainline. In the SH1 Upper Highway Diamond Interchange the detector is placed 140 meters (on the freeway) before the Constellation Drive off-ramp exit.
b. The downstream detector is designed to detect the downstream speed and flow rate (volume). The downstream volume/capacity-ratio is used to measure the bottleneck behaviour. The detector is placed (on the motorway) 210 metres after the South-bound on-ramp exit.

c. The detector at the end of the ramp storage, also called the queue detector, is to detect the queue occupancy. It is placed 180m from South bound on ramp entrance.
d. The check-in detector is located at the ramp metering stop bar to detect the check-in occupancy. The location of the detector is 470 meters from the South-bound on-ramp entrance.
e. Stop-line and advanced detectors are installed on every movement within the interchange.

We are able to analyze and compute the total and average traffic demand for the freeway and arterial roads from the historical data. The membership functions are built based on this analysis. For example: the two mid-block arterial roads, each with a length of 56 meters can hold a maximum of 12 cars on each lane.

3.1.5 Probabilistic Fuzzy Ramp-metering (PFRM)

The PFRM module optimizes the ramp-metering rate considering the traffic flows. The input to this module is acquired by four loop detectors installed at different locations, as stated in the preceding section. Membership functions were selected using trial and error method, however the foundation of these functions are based on the Bogenberger's model (Bogenberger and Keller 2000). The Up-stream detector supplies input variables *local speed, local traffic flow and local occupancy* of the motorway, each of which is described by three fuzzy sets, Low, Medium and High. The downstream detector detects the *downstream speed* and flow rate (volume) for *volume/capacity ratio* (or *v/c ratio*) that is used to measure the bottleneck behavior. The queue detector at the end of the ramp detects the input variable *queue occupancy*. The check-in detector detects the input variable *check-in occupancy.* One fuzzy set is used to describe each *downstream speed, v/c ratio, queue occupancy, check-in occupancy*. The input variables (Table 3.2) of the PFRM are fuzzy and probabilistic, described by membership functions as shown in Figure 3.7, where the symbol "*P*" is the probability of a fuzzy linguistic variable. Thus, a variable in the PFRM is characterized by a quintuple shown in formula 4 below.

$$(x, T(x), U, G, M) \tag{4}$$

where:

x is the name of a variable

$T(x)$is the term set of x;

U is the set of names of linguistic values of x defined

G is the rule for generating the names of values x

M is the semantic rule for associating its meaning with each value.

For example, the variable *local speed* is characterized by a quintuple, *("local speed", {(<"low","0.2">, <"medium","0.2">, <"high", "0.2">)}, [0, 100 kilometers per hour)], G, M).*

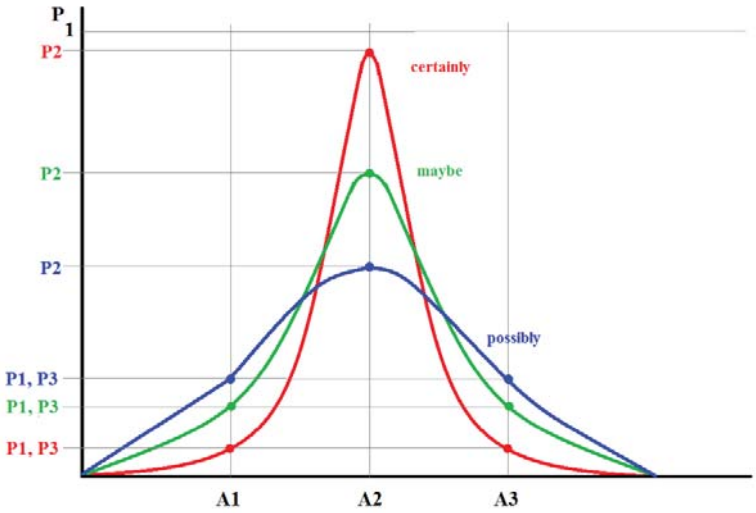


Figure 3.7: Probabilistic fuzzy membership functions of local speed, flow, occupancy, downstream speed, v/c ratio, and check-in and queue occupancy

Table 3.2 Terms of the fuzzy sets for inputs and outputs for PFRM module

Local Speed (0-100 km/h)	Low	Medium	High
Local Flow Rate (0-4000 veh/h)	Low	Medium	High
Metering Rate (240-900 veh/h)	Low	Medium	High
Local Occupancy (0-30%)		High	
Downstream v/c (0-1)		High	
Downstream Speed (0-100 km/h)		High	
Check-in Occupancy (0-50%)		High	
Queue Occupancy (0-50%)		High	
Interchange Queue Occupancy (0-90%)		High	

The membership functions of local speed, flow, occupancy, downstream speed, v/c ratio, check-in and queue occupancy are probabilistic variables with fuzzy rules which can be represented as (5)

$$Tv = \{A_1 / P_1, A_2 / P_{2,}, A_3 / P_3\} \tag{5}$$

where:

A_1, A_2, A_3 are fuzzy variables (corresponds to Low, Medium and High) with probabilities P_1, P_2, P_3 so

$$P_1 + P_2 + P_3 = 1 \tag{6}$$

The output to the PFRM (7) is the *metering rate* that is described by three fuzzy sets, Low, Medium and High from 240 – 900 vehicles / hour. The metering rate is being scaled down (Figure 3.8) by using this equation

$$Scaled\_metering\_rate = \frac{metering\_rate - 900}{900 - 240} \tag{7}$$

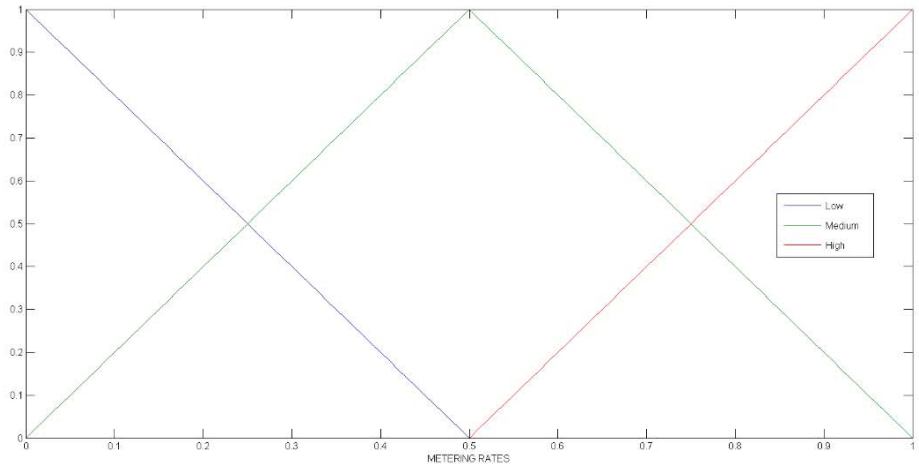


Figure 3.8 Scaled fuzzy ramp-metering Rate

B. Probabilistic Fuzzy Phase Timing (PFPT)

The PFPT is designed to calculate the green time extension for each individual phase of the interchange. Vehicular detectors are installed on “Up-stream-line” and “stop-line”. The number of approaching vehicles for each approach during any given time interval can be estimated using the detectors data regarding the quantity of traffic on the arrival side

(Arrival) and the quantity of traffic on the queuing side (Queue).The vehicle arrivals are calculated based on all the Green signals which include many movements from arterial streets of the interchange. The total vehicle arrival is random. As the signal control is critical during peak hours, this study considers this scenario only. Suppose the PFPT runs Phase B initially with a minimum green time of 6 seconds. The PFPT needs to determine whether to extend or terminate the current green phase when the minimum green time is served. If Phase B is running green, the movements 5, 6, 7, and 9 are considered as the arrival side, whose total number of vehicles is denoted by a variable *Arrival*. The movements 1 to 4, and 8 are queued side, whose total number of vehicles is denoted by a variable Queue. If Phase D is running green then the movements 1 and 5 (since 2 and 3 are linked to 1) are on arrival side and the movements 4, 6, 7, and 8 are queue side. Phase F is running green then the movements 2, 4, 7 and 8 on the arrival side and the movements 1, 5, 6 and 9 on the queue side.

The input variable Arrival (Figure 3.9) is expressed in four fuzzy sets, Few (over 0~14), Small (over 0~28), Medium (over 14~42) and Many (over 42). The input variable Queue (Figure 3.10) is expressed in four fuzzy sets, Few (over 0~19), Small (over 0~38), Medium (over 19~38) and Many (over 38~57). The output variable green extension is fuzzified in four fuzzy sets, Very Short (over 0~10 s), Short (over 0~20 s), Medium (over 10~30 s) and Long (over 20~30 s).

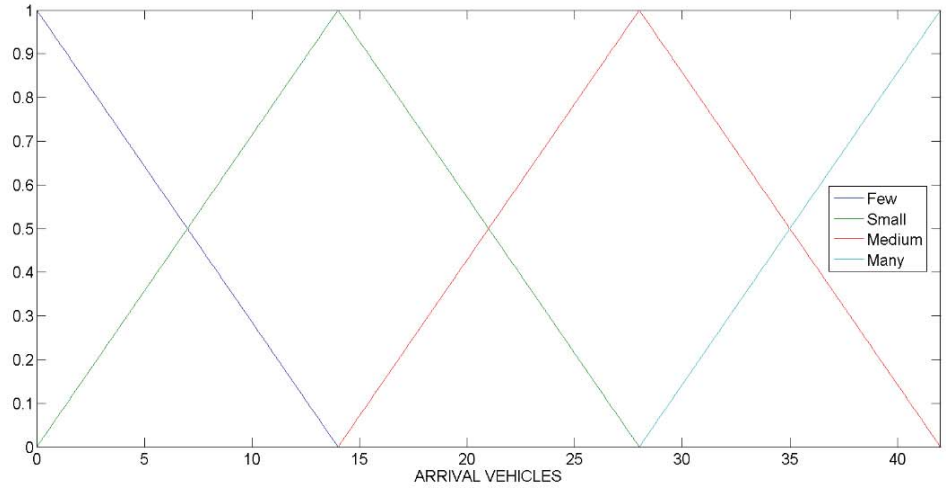


Figure 3.9 Total Arrival membership functions (Phase B)

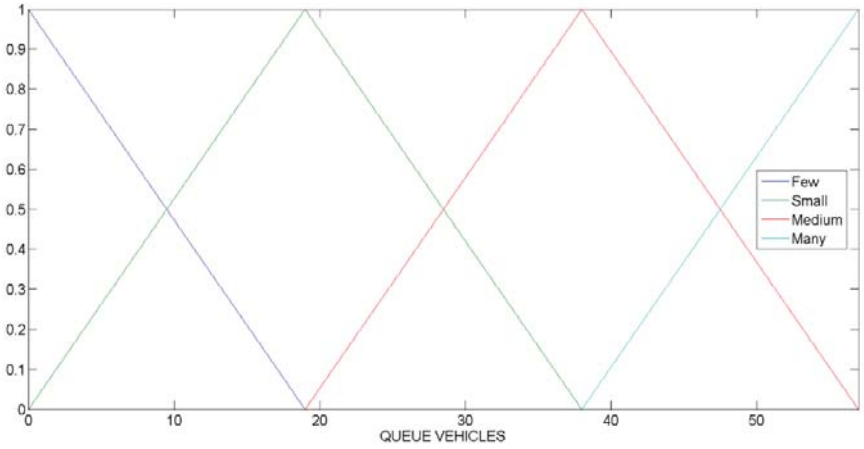


Figure 3.10 Total Queue membership functions (Phase B)

The input variable Arrival and Queue are converted to probabilistic value using the same process as mentioned in probabilistic fuzzy ramp metering (Figure 3.9 and 3.10). Green time extension, which is the extension needed for the green light on the arrival side, and also the output of the fuzzy variable. In this model, the minimum green time is first given for arriving traffic flows. After that the extension process is stopped and the next phase is selected according to the Phase Selection Logic (PSL). For each phase there is a minimum run of 6 seconds based on Auckland Traffic Operations and Management (ATOM) timing plan (Figure 3.11).

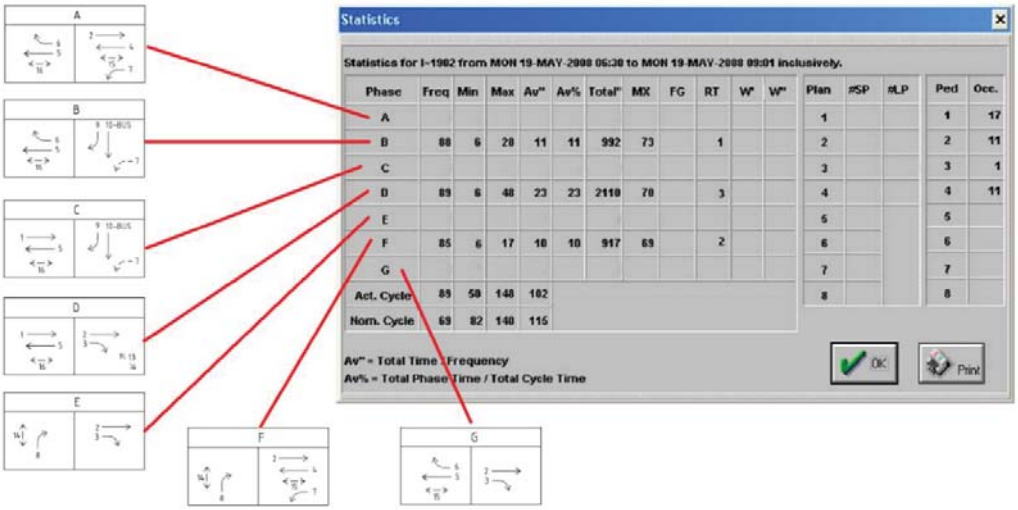


Figure 3.11 Auckland Traffic Operations and Management (ATOM) timing plan for Upper Harbour Interchange

During this process (Figure 3.12), the PFLC infers the output Green Time Extension for the running phase which has a range from 6 to 30 seconds (Table 3.1). Once this is complete, the PFLC moves on to the next phase.

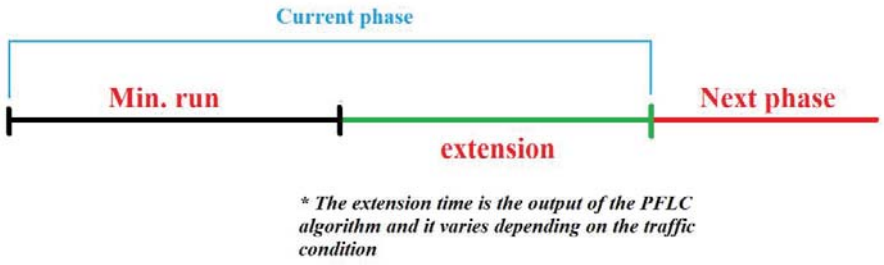


Figure 3.12 Fuzzy green time extensions in current running phase

C. Rule bases of the PFLDI model

There are two rule bases for the PFLDI. One is with the PFRM module and the other with the PFPT module. They are built on the combination of input and output variables. The configuration of the fuzzy rules in this study is based on the traffic data collected from the Upper Harbour Diamond Interchange site survey. By gathering the data we try our best to replicate the operation of this diamond interchange during the simulation. While the PFLDI algorithm in this study is designed based on the Upper Harbour Diamond Interchange, it can be easily applied to any typical diamond interchange.

The resulting PFRM rule base (Table 3.3) was formed by 10 rules. The first three rules (Rule 1, 2 and 3) are to ensure that at least one rule will be triggered as all the occupancy range has been covered. Rule 8 is designed to prevent the formation of downstream congestion. Volume/capacity ratio (v/c ratio) is calculated from the historical measured maximum flow rate of downstream and could be seen as a prediction of the downstream bottleneck behavior. Each rule has weight, ranging from 1.5 to 3.0, that sets the priority of each rule (Table 3.3).

Table 3.3 Rules and its weighting for PFRM module

RULE	IF		THEN	WEIGHT
1	Local Occupancy = Low		Metering Rate = High	1.5
2	Local Occupancy = Medium		Metering Rate = Medium	1.5
3	Local Occupancy = High		Metering Rate = Low	2.0
4	Local Speed = Low	Local Flow Rate = High	Metering Rate = Low	2.0
5	Local Speed = Medium	Local Occupancy = High	Metering Rate = Medium	1.0
6	Local Speed = Medium	Local Occupancy = Low	Metering Rate = High	1.0
7	Local Speed = High	Local Flow Rate = High	Metering Rate = High	1.0
8	Downstream speed = Very Low	Downstream v/c = Very High	Metering Rate = Low	3.0
9	Check-in Occupancy = Very High	Queue Occupancy = Very High	Metering Rate = High	3.0
10	Interchange Queue Occupancy = High	Queue Occupancy = High	Metering Rate = High	3.0

The choice of the weights is based on our best understanding of the traffic condition at the study site (survey and historical data). For example: if Downstream Speed is VERY LOW and Downstream v/c is VERY HIGH, then Metering Rate is LOW. The rule weight for this rule is 3.0 (highest priority). Rules 1 and 6 are designed to restrict the metering rate when the vehicles are unable to merge onto the motorway. When the motorway is highly congested, a secondary queue of metered vehicles may form. If a secondary queue persists, ramp-metering is no longer providing any benefit. Rule 9 and 10 are used for the situation when there is a long queue on the ramp. The PFRM module has to extend the metering rate to the maximum, in order to avoid the long queue on the ramp as well as preventing the traffic spillback to the connected intersection. If this occurs, the metering rate will be set to high allowing more cars to be merged onto the motorway/freeway. The interchange queue occupancy detectors are placed in various places at the connected intersections.

The rule base for PFPT is developed with respect to the three phases B, D and F used during the peak hours. For each phase, there are 16 rules (Table 3.4). Similar to a conventional fuzzy system, the final output needs to be in a non-fuzzy form (green time extension). This conversion is done by using the continuous centroid formula (Pappis and Mamdani 1977).

Table 3.4 Rules and its weighting for PFPT module

RULE	IF		THEN	WEIGHT
	ARRIVAL	QUEUE	EXTENSION	
1	Few	Few	Very Short	1
2	Few	Small	Very Short	1
3	Few	Medium	Very Short	1
4	Few	Many	Very Short	1
5	Small	Few	Short	2
6	Small	Small	Short	2
7	Small	Medium	Very Short	1
8	Small	Many	Very Short	1
9	Medium	Few	Medium	3
10	Medium	Small	Medium	3
11	Medium	Medium	Short	2
12	Medium	Many	Short	2
13	Many	Few	Long	4
14	Many	Small	Medium	3
15	Many	Medium	Medium	3
16	Few	Many	Short	2

The defuzzification operation in probabilistic fuzzy logic is associated with probabilistic fuzzy set instead of the normal fuzzy set so that the mathematical expectation of *G* (green time extension) is the crisp output of the PFLC in formula 8.

$$G = E\left[X_{(G)} \right] \tag{8}$$

where:

G is the crisp output

E is the mathematical expectation.

Table 3.5 shows the probabilistic fuzzy green time extension value with various traffic condtions.

Table 3.5: Probabilistic Fuzzy Green Time Extension

PHASE D			PHASE B			PHASE F		
No. of Arrival Vehicles	**No. of Queue Vehicles**	**Green Time Extension (0-48 sec)**	**No. of Arrival Vehicles**	**No. of Queue Vehicles**	**Green Time Extension (0-30 sec)**	**No. of Arrival Vehicles**	**No. of Queue Vehicles**	**Green Time Extension (0-18 sec)**
47	12	33.00	10	11	8.75	24	43	8.05
8	56	10.78	14	58	4.89	16	69	3.41
13	52	8.45	19	60	7.66	12	9	5.50
22	18	19.04	27	15	20.23	39	64	8.63
2	17	8.49	18	62	7.37	35	43	9.88
54	8	35.00	38	6	23.21	25	55	5.29
49	26	28.61	37	70	12.60	10	49	2.29
44	35	23.68	5	28	6.32	41	51	11.16
35	22	26.21	16	13	13.16	28	43	7.25
8	9	12.02	32	22	20.25	32	67	7.10
23	38	9.04	17	18	13.72	25	36	7.95
6	35	7.61	16	45	8.48	5	39	3.38
21	18	18.29	35	41	18.06	10	43	3.02
24	24	18.01	36	28	18.75	21	2	9.00
37	45	15.47	16	3	12.84	41	54	11.04
48	23	29.63	21	19	15.81	23	39	7.00
29	32	17.80	38	12	21.84	36	54	8.74
41	56	16.89	28	6	20.42	9	38	3.56
40	52	17.52	7	14	7.89	18	26	7.38
10	33	8.47	10	51	4.44	8	32	4.04
9	20	11.73	4	71	7.07	17	17	7.80
30	8	24.68	36	69	12.60	13	50	2.44
6	13	11.29	29	20	20.44	15	71	2.76
29	14	24.28	7	47	3.82	22	52	4.85
40	50	17.52	11	20	9.11	6	65	3.68
9	44	8.31	34	51	15.47	33	63	7.58
39	35	19.81	17	26	12.49	18	12	8.18
34	24	24.10	14	36	9.00	13	68	2.44
56	37	31.24	28	54	11.33	31	43	8.67
Average Total number of arrival vehicles		***28 veh***	Average Total number of arrival vehicles		***21 veh***	Average Total number of arrival vehicles		***22 veh***
Average Total number of queue vehicles		***30 veh***	Average Total number of queue vehicles		***33 veh***	Average Total number of queue vehicles		***45 veh***
Average Probabilistic Fuzzy Green Time Extension		***18.52 sec***	Average Probabilistic Fuzzy Green Time Extension		***12.69 sec***	Average Probabilistic Fuzzy Green Time Extension		***6.28 sec***

3.2 ALGORITHM IMPLEMENTATION

This section describes the implementation of PFLDI algorithm. Section 3.2.1 provides the guidelines on how to determine several parameters using an example. A framework of algorithm implementation is provided in Section 3.2.2.

3.2.1 Parameters for implementing PFLDI Algorithm

To implement the PFLDI method developed in Section 3.13, it is essential to determine the value of some parameters such as the distance between the stop-line and the check-in detectors and the detection range. This distance depends on the length of the arterial roads, on/off ramps, average vehicle speed, and queue length at the stop-line. This study uses Upper Harbour Diamond Interchange as an example illustrated in Figure 3.14 to provide guidelines on how to determine these basic parameters for implementing the PFLDI algorithm. A typical New Zealand diamond interchange is about 3 kilometers long and the distance between the two intersections is around 120 meters. The interchange is equipped with various detectors for its traffic actuated signal control. The detectors for the intersection signals are placed at 1 m before the stop line and 40 to 60 m away from the stop line for each movement (Figure 3.13).

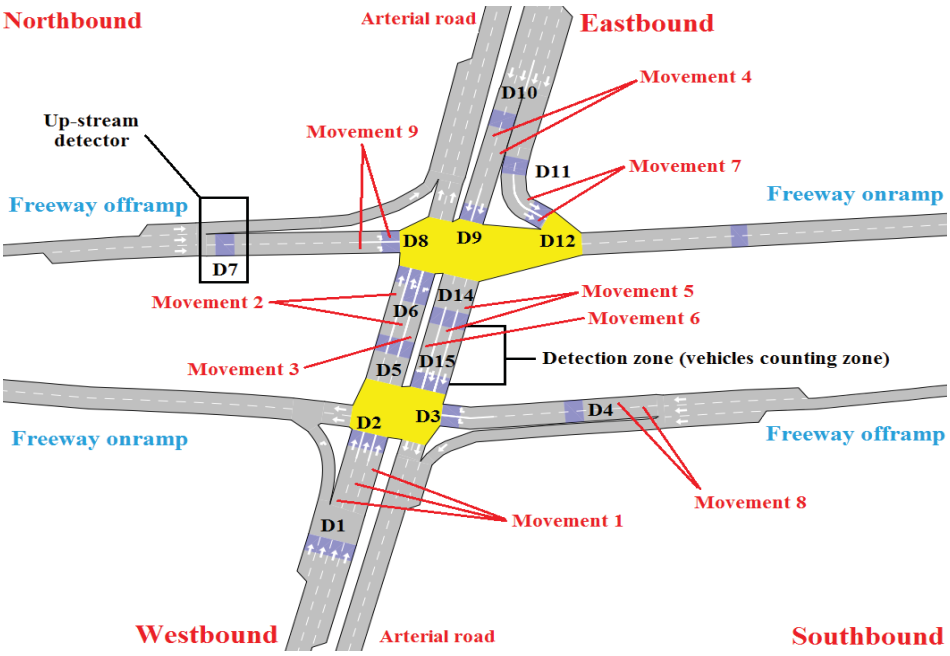


Figure 3.13 Detectors placement in Upper Harbour Diamond Interchange's two middle intersections

The detectors for on ramp-metering are set up as follows: Ramp Queue detector at 70 m from on-ramp entrance, Ramp Check-In detector at 413 m from on-ramp entrance; Up-stream detector at 160 m from the on-ramp exit on the motorway; Down-stream detector at 200 m before the on-ramp exit; and Interchange Queue Occupancy detector at 40 m away from the stop-line (Figure 3.14).

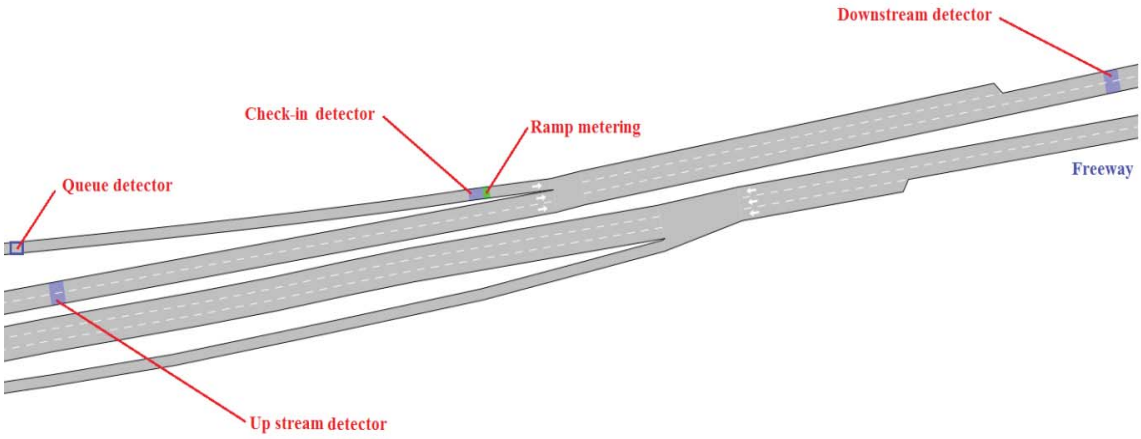


Figure 3.14 Detectors placement along the on-ramps and motorway mainstreams

3.2.2 Framework of PFLDI Algorithm Implementation

The framework and procedure for implementing the PFLDI control is presented in Figure 3.15. Inputs include initial signal phase, initial queue length, and continuously undated vehicle information from the detectors. Initial queues in the beginning of each movement can be calculated using the counting function. For example, the PFLDI counts the number of arrival and queue vehicles for the first 6 seconds of the current running phase. This counting process only happens once for each phase. The Probabilistic Fuzzy Phase Timing for each phase will generate the appropriate green time extension based on these inputs.

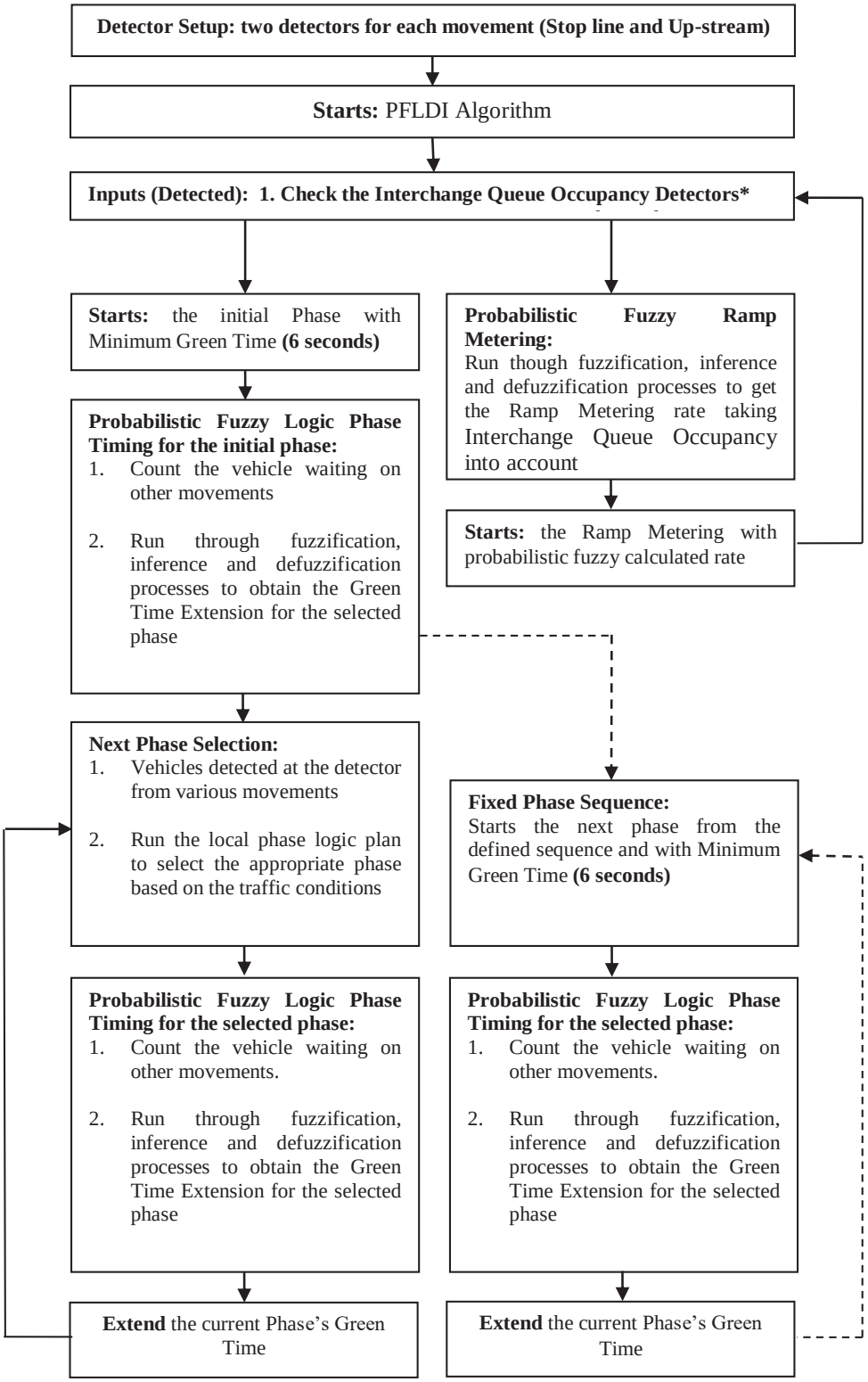


Figure 3.15 Framework of PFLDI Algorithm

3.3 USING SIMULATION TO EVALUATE THE PFLDI ALGORITHM

3.3.1 Simulation Evaluation Procedure

Due to the nature of this study and the limited funding, the simulation of diamond interchange operation provides a cost-effective way to evaluate the performance of the algorithm and its strategy. Several signal controls methods are evaluated in the same environment and results are compared accordingly. The simulation evaluation procedure is shown in Figure 3.16. The values of calibrated parameters are the same for all comparisons cases such as driver behaviour assumptions, vehicle characteristics, traffic demands, phasing, etc. This study compares the performance of the entire diamond interchange from three signal control algorithms (PFLDI, FLDI and Actuated).

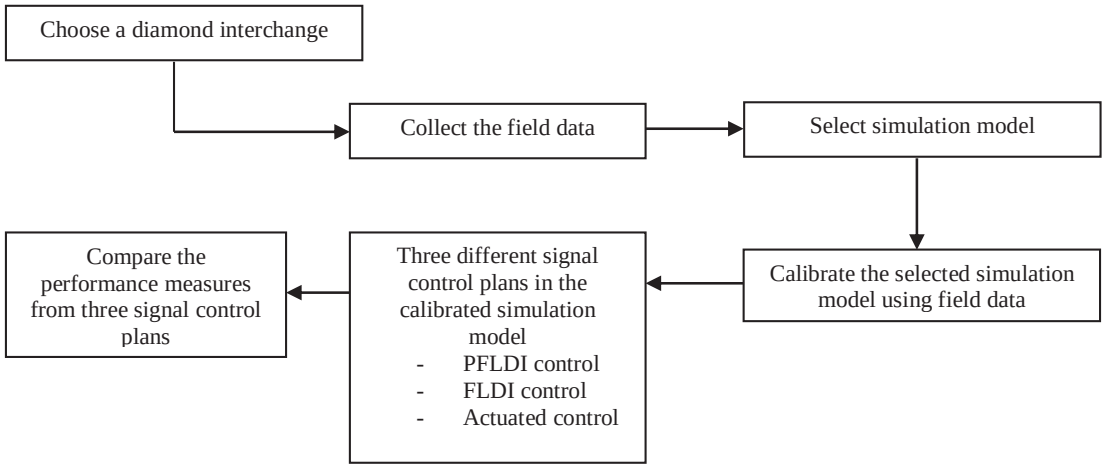


Figure 3.16 Simulation evaluation procedures

3.3.2 Microscopic Traffic Simulation Model

With the state-of-the-art computing technology, microscopic simulation models have been widely used in transportation issues such as ramp-metering, diamond interchange, intersection applications (Hassan et al, 1998). While various traffic simulation models are available, Paramics and AIMSUN (Fang, 2004) are two widely used in microscopic simulation models for studying traffic problems. They provide users with the flexibility of

developing special signal control logics and algorithms, which are essential in evaluating new control strategies and algorithms (Tian, 2004).

AIMSUN 6 (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network version 6) is a microscopic, stochastic model for simulating the traffic of the road networks. The microscopic traffic simulation suite AIMSUN reproduces traffic flow by modelling the behaviour of every single vehicle, the interactions between different vehicles and the network model. The wide variety of possible settings leads to a very realistic one-to-one reproduction of a network's traffic. It includes an animated simulation display, which shows vehicles moving through the network. The model can simulate a range of traffic management features including incident detection and surveillance systems, variable message signs and wide area traffic control strategies. Simulating predictive control and guidance strategies are also potentially feasible (AIMSUN, 2008). One of the most important features for AIMSUN is being able to connect to common adaptive control interfaces such as SCATS, VS-PLUS, UPTOPIA as shown in Figure 3.17 (Pham, 2009).

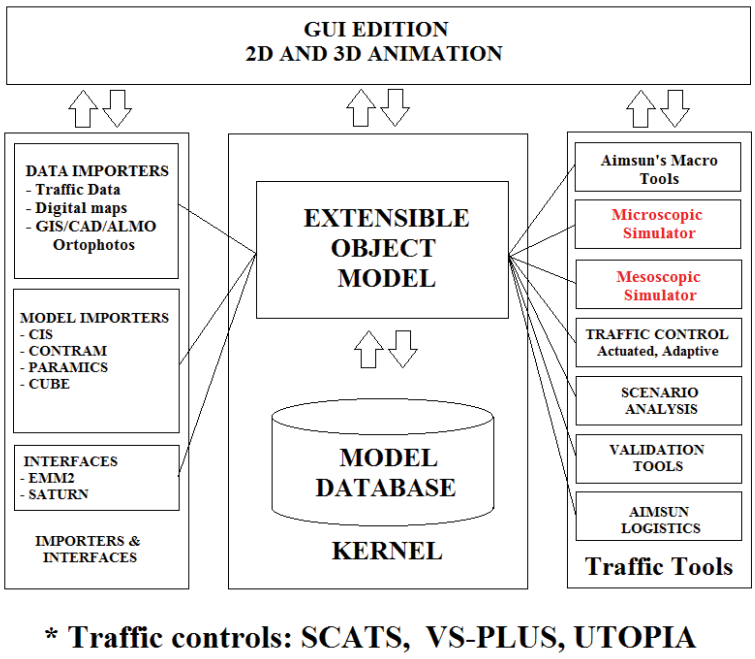


Figure 3.17 AIMSUN 6 environment (Excerpt from: TSS-GETRAM Extension User Manual, 2002)

3.3.3 Calibration of Selected Simulation Model

The selected simulation model, in this case Upper Harbour Diamond Interchange, has been calibrated using field data surveyed and historical data obtained from ATOM, New Zealand Transportation Agency and North Shore City Council. According to Elefteriadou et al. (1997), calibration is the process of quantifying model parameters using real-world data in the model logic so that the model can realistically represent the traffic environment being analysed. AIMSUN traffic simulation model provides default values of some parameters including vehicle characteristics, drivers, traffic model etc. These values will be adjusted with the values obtained from field surveys and historical data provided by the government transportation agency. This to ensure the calibrated model outputs have the same performance measures results as field surveys and observations.

The calibration of the diamond interchange for evaluating the PFLDI algorithm and comparison study is based on the field data of a site obtained from New Zealand Transportation Agency and North Shore City Council. The data used in this study includes

- Video of vehicle arrivals, queues, on/off ramps and motorways
- Geometric layout as shown in Figure 3.18
- Signal timing plan
- Traffic demand
- Other relevant information

The average flow rates, speed, occupancy and heavy occupancy vehicles data were obtained from ATOM. The turning rates were obtained from video recording of the study site.

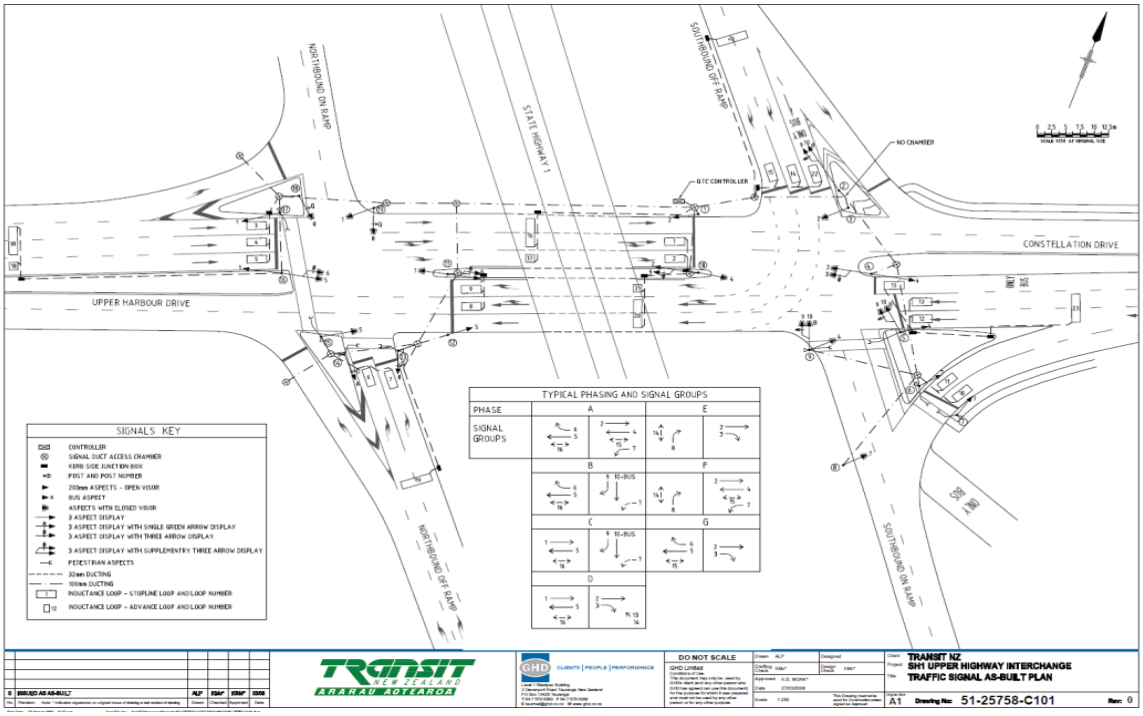


Figure 3.18 Geometric layout of Upper Harbour Diamond Interchange (Excerpt from: NZTA)

(The high resolution image above can be found in the Appendix)

Parameters below are modified from their default values in AIMSUN for calibration:

Table 3.6 Calibration data for AIMSUN simulation model

Parameter in AIMSUN	Maximum (calibrated / default)	Minimum (calibrated / default)
Vehicle maximum speeds (kph)	50 / 60	40 / 50
Vehicle maximum acceleration (m/s)	2.2 / 2.5	2.2 / 2.5
Arterial roads speed limit (kph)	50 / 60	40 /50
Ramp speed limit (kph)	100 / 120	80 / 100
Motorway speed limit (kph)	100 / 120	80 / 100
Arterials turning speed limit (kph)	50 / 60	40 / 50
Ramps turning speed limit (kph)	50 / 60	40 / 50

Aside from the parameters in Table 3.5, other modified parameters are considered:

- Driver's reaction time: decreased from 0.75s to 0.5s
- Response time at stop: decreased from 1s to 0.75

3.3.4 Traffic Turning

The traffic turning information in this simulation is based on the field data during the peak hours from 08:00 to 09:00AM on 14/07/2008. Three cameras were placed around the diamond interchange, the first video camera was placed at the South-bound on-ramp exit, the second video camera was placed on the Constellation Drive traffic junction next to the Constellation Bus Station's entrance, and the last one was placed near the cyclist crossing by the Upper Harbour Highway (SH18) traffic junction. The turning traffic is set up as in Figure 3.19 and 3.20.

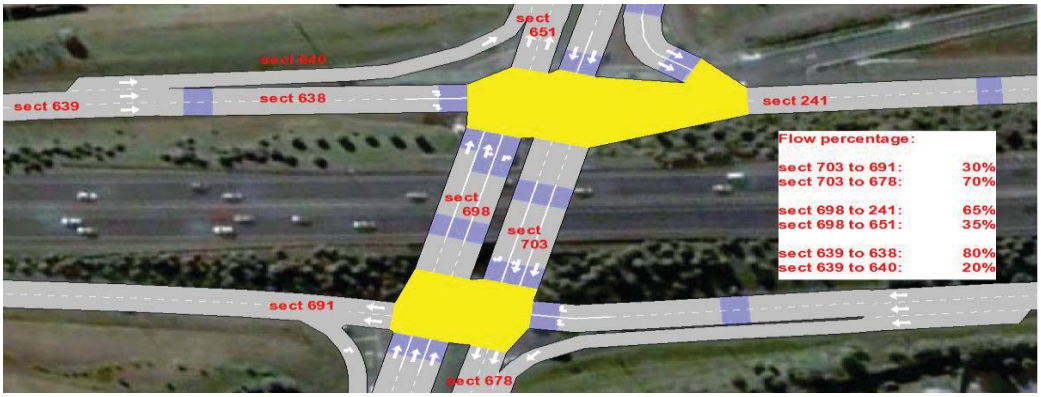
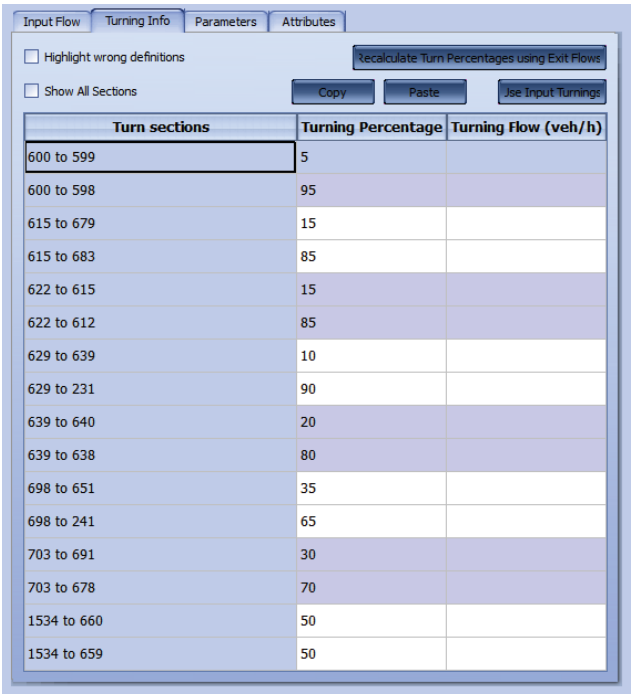


Figure 3.19 Turning traffic observed



Turn sections	Turning Percentage	Turning Flow (veh/h)
600 to 599	5	
600 to 598	95	
615 to 679	15	
615 to 683	85	
622 to 615	15	
622 to 612	85	
629 to 639	10	
629 to 231	90	
639 to 640	20	
639 to 638	80	
698 to 651	35	
698 to 241	65	
703 to 691	30	
703 to 678	70	
1534 to 660	50	
1534 to 659	50	

Figure 3.20 Turning traffic values in AIMSUN

3.3.5 Simulation Assumption

In this study, Upper Harbour Diamond Interchange is used as a testing site for the PFLDI model implementation. The simulation model is based on the following assumptions:

- No bus is used in this simulation (starting from mid-2007, buses travelling to and from Auckland CBD operate in their own lanes which are independent from the motorway)
- No cyclist in the simulation
- The simulation will mimic the real-world peak hour period from 8:00-9:00 on Monday morning

3.3.6 Driver and Vehicle Information

Vehicles and drivers have a range of characteristics that affect the way they travel through a road network. These are made of by two attributes, which are mechanical attributes of the vehicle and the aspects of driver behaviour. For each vehicle type, the traffic flows must be entered separately rather than as a percentage of the total flow. The number of vehicle types in this model has been limited to two: cars, heavy vehicles, or high occupancy vehicles. In AIMSUN, this type of information is input as parameters pertaining to vehicle types, any number of which can be defined by the user. These include desired speed, acceleration, normal and emergency acceleration, maximum yield time and minimum vehicle spacing when stop in a queue. According to Hughes (2000), the queuing up and queue leaving speeds control whether or not a vehicle will enter an intersection that contains vehicles which are "queued", as defined by these parameters. In a traffic stream these data may vary stochastically between vehicles. For each data item (desired speed, acceleration etc.), the values attributed to individual vehicles are considered normally distributed. The main characteristics (list below) for New Zealand vehicles are outlines in Hughes's master thesis (2000) and New Zealand Ministry of Transportation travel survey. For heavy vehicles, no accurate field data exist so default values were used.

- Capacity on ramps: maximum 1700 vehicles/hour/lane
- Capacity on arterials: maximum of 1000 vehicles/hour/lane
- Capacity on motorway: max 2500 vehicles/hour/lane

3.3.7 Detector Type and Placement

Single loop inductive sensor (detector) is used in this simulation accompany by the single lane metering control system. The length for each detector is 4.5 meters. Detector's measurements are traffic counts, density, presence, occupancy and headway.

Detectors are placed according to the layout plan shown in Figure 3.2, 3.13 and 3.21. These detectors generate data for vehicle counts, speed, occupancy (percentage of time step the detector is pressed) and presence (determine whether a vehicle is on the detector or not). They are the required data for implementing the proposed algorithm.

Aside from the detector values, simulation step 0.5s is used to collect vehicle detection information. The smaller interval means more accurate detected data which helps the PFLDI to generate accurate green time extension.

i) Ramp Detectors Setup

Detector 1 (Ramp Queue detector): 70m from on-ramp entrance.

Detector 2 (Ramp Check-in detector): 413m from on-ramp entrance.

Detector 3 (Upstream Detector): 160m from South-bound on-ramp exit on SH1.

Detector 4 (Downstream Detector): 200m before South-bound on-ramp exit on SH1.

Detector 5 and 6 (Interchange Queue Occupancy/Connected Intersection Advanced Loop): 40m away from the stop-line mark on Upper Harbour Drive Eastern direction.

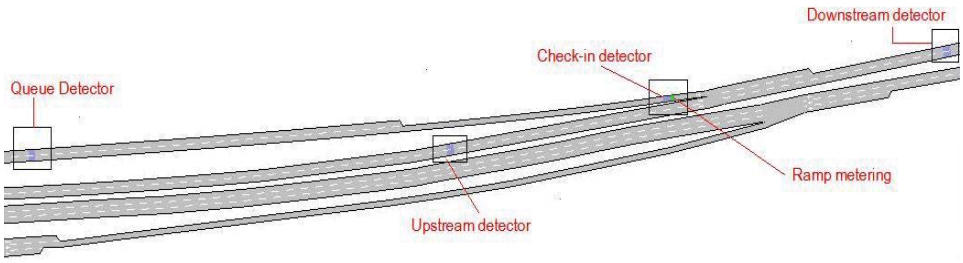


Figure 3.21 Detectors setup for PFR system in AIMSUN

ii) Interchange Detectors Setup

Detectors 843, 1571, 1572: are placed 1m before the stop line mark on Upper Harbour Drive Eastern direction. These are called stop-line detectors.

Detectors 1574 and 1573: are placed 46m away from the stop line mark on Upper Harbour Drive Eastern direction. These are called up-stream detectors.

Detectors 1578 and 737: are placed 1m before the stop line mark on North-bound off-ramp

Detector 744 is placed is 40m away from the stop line mark on North-bound off-ramp.

Detectors 841, 1579 and 1593: are placed 1 m before the stop line mark on Upper Harbour Drive Western direction.

Detectors 1541, 1592 and 740: are placed 40m away from the stop line mark on Upper Harbour Drive Western direction.

Detectors 1590, 1585 and 845: are placed 1m away from the stop line mark on Upper Harbour Drive Easter direction.

Detectors 741, 1589 and 1540: are placed 40m away from the stop line mark on Upper Harbour Drive Eastern direction.

Detectors 1583, 841: are placed 1m away from the stop line mark on South-bound off-ramp.

Detector 742: is place 40m from the stop line mark on South-bound off-ramp.

Detectors 745, 1582: are placed 1m away from the stop line mark on Constellation Drive Western direction.

Detector 743: is placed 60m away from the stop line mark on Constellation Drive Western direction.

Detector 1580 and 1581: are placed 1m away from the stop line mark on South-bound on-ramp.

Detector 1587: is placed 20m away from the stop line mark on South-bound on-ramp.

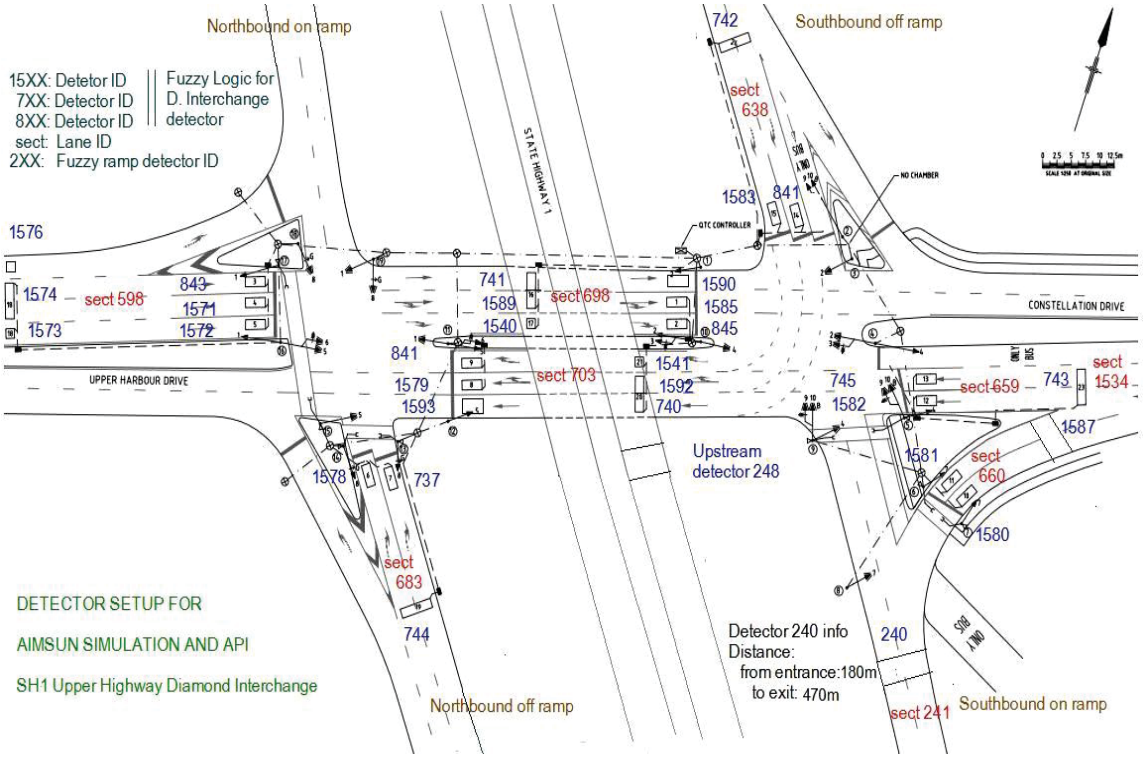


Figure 3.22 Detectors setup for SH1 Upper Highway Diamond Interchange in AIMSUN

3.3.8 Dynamic scenario setup

- Simulation type: microscopic simulator
- Detection interval: 6 seconds
- Statistics recording: every minute
- Traffic demand: 08:00 – 09:00 AM
- Simulation day: Tuesday
- Weather: dry
- Season: winter
- Control plans: actuated and fuzzy (API)
- Car following model: Deceleration estimation (Leader deceleration)
- Lane changing model:
- Percent overtake: 90%
- Percent recover: 95%
- On ramp model: cooperative mode looking gaps upstream
- Route choice model: fixed using Travel Time in Free Flow models

3.3.9 Simulation of the PFLDI Algorithm in AIMSUN

AIMSUN is well known for its powerful GETRAM extension module (part of the Application Programming Interface) which can be used to implement advanced traffic control applications such as PFLDI, FLDI, etc.

i) GETRAM Extension Module

Since AIMSUN is unable to implement adaptive traffic control with the standard software pack, the AIMSUN API module has been used to enable the communication between the AIMSUN simulation model and a user-built control algorithm. Figure 3.23 illustrates the conceptual structure of how AIMSUN working with user application by means of AIMSUN API module:

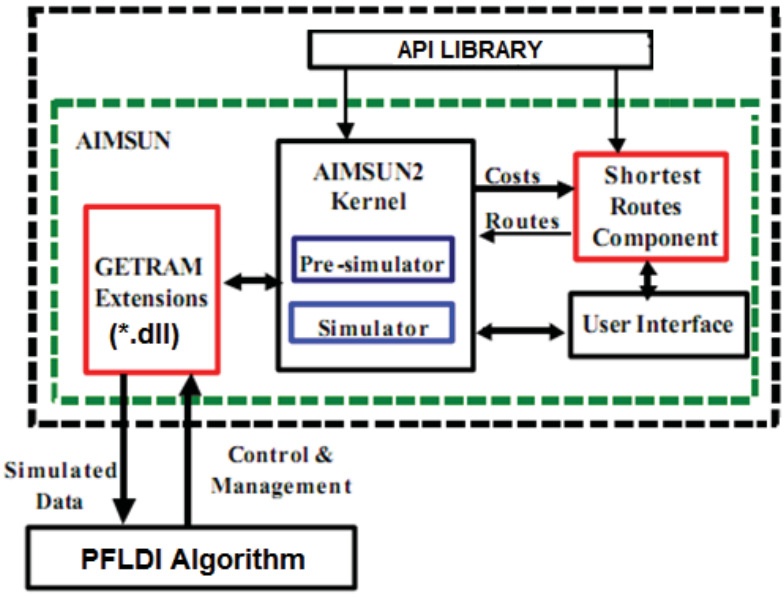


Figure 3.23 Simulations of the PFLDI / FLDI Algorithm in AIMSUN via GETRAM Extension

The AIMSUN API module provides a set of functions to collect the required data (e.g. flow, occupancy, etc.) from traffic simulation. Based on the collected information, the EXTERNAL APPLICATION (user-built control algorithm) makes some control decisions which will be applied to the simulation. Such a process

completes the communication between the AIMSUN simulation model and a user-built control algorithm.

The communication process (Figure 3.24) is guaranteed by eight high level functions defined in AIMSUN API module: AAPILoad, AAPIInit, AAPIManage, AAPIPostManage, AAPIFinish, AAPIUnLoad, AAPIEnterVehicle and AAPIExitVehicle.

a. AAPILoad (): It is called when the module is loaded by AIMSUN.

b. AAPIInit (): It is called when AIMSUN starts the simulation and can be used to initialise whatever the module needs.

c. AAPIManage (): This is called in every simulation step at the beginning of the Bicycle, and can be used to request detector measures, vehicle information and interact with junctions, metering and VMS in order to implement the control and management policy.

d. AAPIPostManage (): This is called in every simulation step at the end of the Bicycle, and can be used to request detector measures, vehicle information and interact with junctions, metering and VMS in order to implement the control and management policy.

e. AAPIFinish (): It is called when AIMSUN finish the simulation and can be used to finish whatever the module needs.

f. AAPIUnLoad (): It is called when the module is unloaded by AIMSUN.

The scheme of how AIMSUN interacts with AIMSUN API is shown in Figure 3.11.

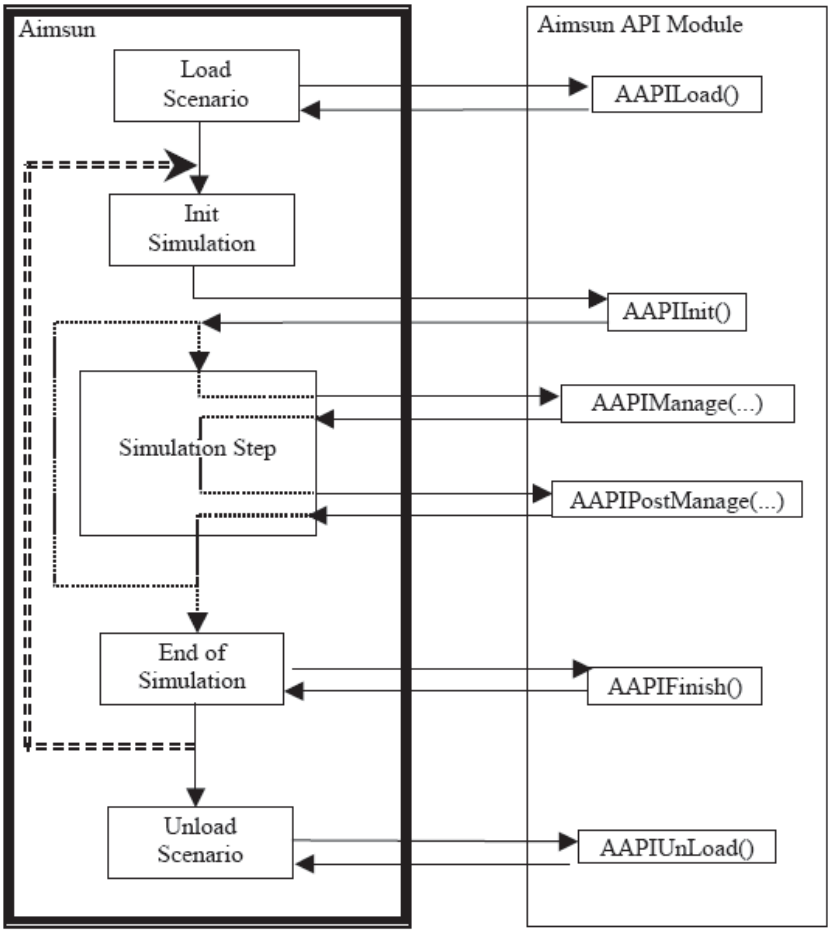


Figure 3.24 Interaction between AIMSUN and its API module
(Excerpt from: TSS-GETRAM Extension User Manual, 2002)

The proposed ramp metering algorithms programmed in Microsoft Visual C++ is implemented in AIMSUN simulator through AAPIManage () and AAPIPostManage () functions using Microsoft Visual Studio 2008, where a Dynamic Link Library (DLL) will be generated and integrated to the simulator.

ii) AIMSUN Ramp-metering Control

AIMSUN also incorporates ramp-metering control. This type of control is used to limit the input flow to certain roads or freeways in order to maintain certain smooth traffic conditions. The objective is to ensure that entrance demand never surpasses the capacity of the main road. AIMSUN considers three types of ramp

metering depending on the implementation and the parameters that characterize it (Barceló et al., 1995):

a. Green time metering, with parameters green time and Bicycle time. It is modelled as a traffic light.

b. The flow metering is automatically regulated in order to permit the entrance of a certain maximum number of vehicles per hour.

c. Delay metering, with parameters mean delay time and its standard deviation. It is used to model the stopped vehicles due to some control facility, such as a toll or a customs checkpoint.

The EXTERNAL APPLICATION can modify this modelling by different actions. It can:

a. Change the parameters of a metering: the EXTERNAL APPLICATION can dynamically modify the parameters that define a ramp metering.

b. Disable the control structure: EXTERNAL APPLICATION disables the structure of the ramp metering and completely controls the state changing.

c. Change the state of a metering: the EXTERNAL APPLICATION can change the current state to another. If the metering has not disabled the control, AIMSUN6 programmes the next changing of state taking into account the parameters, which define the control. Otherwise, AIMSUN2 holds the new state until the EXTERNAL APPLICATION changes it to another.

3.4 COMPARISONS OF DIFFERENT SIGNAL CONTROL METHODS

This section presents comparisons of the performance of PFLDI, FLDI and Actuated control. The comparisons is conducted using the calibrated Upper Harbour diamond interchange ranging from low to extreme traffic demand. The performance measures from three algorithms are compared with respect to each scenario. Lastly, the summary of the comparison is provided.

3.4.1 Overview

The signal plans are obtained for each demand scenario before running the AIMSUN simulation. These signal plans are then input to the AIMSUN interchange (in this case Upper Harbour Diamond Inter change). The signal plans of the PFLDI algorithm are not pre-determined, they generated based on real-time demand scenario. The simulated Upper Harbour Diamond Interchange is then used to comparing the performance of three different signal controls under the same scenarios. The simulation time is 1 hour on a typical Monday morning (Pham, 2013).

3.4.2 Performance Measure of Effectiveness

The PFLDI performance is measured by using the following Measures of Effectiveness **(MOE)**. Travel delay is defined as the time difference between the desired and actual travel time of a vehicle. In this definition, desired travel time is determined (by the AIMSUN) from both the motorway speed limit and driving characteristics of each individual vehicle.

i) Motorway / Freeway MOE

Total travel time: is the total time accumulated by all the vehicles while travelling on the motorway mainline (vehicle-hours) within a specified period of analysis.

Average flow rate: is the average hourly rate of vehicles passing a point on the motorway during a given time interval. These flow rates are recorded by the up-stream and down-stream loop detectors located on the motorway (vehicles/hour).

Average delay: is the time per vehicle while travelling on the freeway mainline (minutes/vehicle) within a specified period of analysis.

Average speed: is the space mean speed for vehicles serviced by the freeway mainline (kilometres/hour) within a period of analysis.

ii) Ramp delay / Queue MOE

Total travel time: is the time accumulated on all metered ramp (vehicle-hours).

Average delay: is the Ramp Total Delay averaged over ramp volumes (minutes/vehicle).

Average flow rate: is the average hourly rate of vehicles passing the check-in detector located on the on-ramp (vehicles/hour).

iii) System performance MOE

Total travel time: is the time accumulated by all vehicles (vehicle-hours).

Average delay: is the total delay time accumulated by all vehicles (vehicle-kilometres).

Average flow rate: is the total flow rate accumulated by all vehicles on the diamond interchange (vehicles/hour).

3.4.3 Simulation pictures

Figure 3.25 (a)

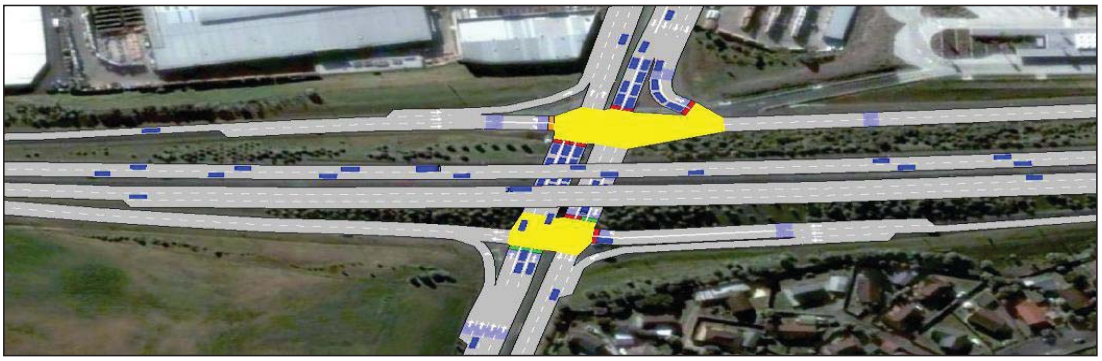


Figure 3.25 (b)

Figure 3.25 (c)

Figure 3.25 (a) (b) (c) State Highway 1 Upper Highway Interchange AIMSUN simulation pictures

3.4.5 Traffic Demand Information

AIMSUN allows two methods of traffic demand input. Firstly, the traffic demand can be entered by traffic flows on each entry link and subsequent turning percentages at each intersection. The drawback of this approach is that a vehicle does not know its final destination upon entering the network. Therefore, turning movements are carried out probabilistically and weaving patterns may be unrealistic. A more realistic lane changing behaviour for vehicles can be achieved using Origin-Destination (O/D) matrices for traffic demand input (Hass, 2001). This ramp model is using the traffic flows that include upstream traffic demand and ramp traffic demand. This is a one hour simulation with traffic demand changes at every 15 minutes.

In AIMSUN, the congestion usually happens when the total of the ramp traffic demand and upstream demand is equal to the downstream traffic capacity for an isolated on-ramp model. Since there are two lanes so the total road capacity will be around 5000 vehicles/hour, so to create the congestion the total traffic demand (sum of upstream and ramp demands) is set to 5000 vehicles per hour to test the performance of the fuzzy algorithm and find out the specific ranges of traffic flow conditions the fuzzy algorithm works best (Pham, 2013).

Totally six traffic demands are used in this simulation: 3348 vehicles/hour (Table 3.7), 3752 vehicles/hour (Table 3.8), 4592 vehicles/hour (Table 3.11), 6272 vehicles/hour (Table 3.12), 7952 vehicles/hour (Table 3.15) and 8783 vehicles/hour (Table 3.16). These demands are used to simulate the traffic congestion. The traffic demand for up-stream is

less than 3000 vehicles/hour is not relevant in this case because the average up-stream demand (SH1 Greville to Tristram – South-bound) and should be more than 4000 vehicles per hour to cause the downstream congestion (after the Constellation Drive on-ramp exit). According to the data obtained from Transit New Zealand (TNZ, 2008), the average upstream demand rarely exceeds the 4200 vehicles/hour. On the other hand, when average upstream demand is lower than 4000 vehicles/hour the motorway is under free flow condition as the freeway/motorway capacity is more than 5000 vehicles/hour (Pham, 2013).

3.4.6 Comparison for low traffic demand

Several low demand scenarios are designed to study the performance using PFLDI algorithm. As illustrated in Table 3.7 and Table 3.8 below, the demand changes every 15 minute period in order to determine whether or not the PFLDI algorithm has the capability to discharge the increased in volume on particular movements while maintaining the operation of the entire diamond interchange system.

Table 3.7 Traffic demand data for Scenario 1 (3348 vehicles per hour) - Low

Time (AM)		Constellation towards Upper Harbour		Upper Harbour towards Constellation		Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Car	HOV	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	379	19	278	14	1344	67	36	2
8:15	8:30	430	22	384	19	1582	79	96	5
8:30	8:45	481	24	490	25	1749	87	137	7
8:45	9:00	727	36	509	25	2216	111	254	13
Average Demand		504	25	415	21	1723	86	131	7
Total Demand		529		436		1809		138	

Table 3.8 Traffic demand data for Scenario 2 (3752 vehicles per hour) – Low

Time (AM)		Constellation towards Upper Harbour		Upper Harbour towards Constellation		Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Car	HOV	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	579	29	478	24	1544	77	236	12
8:15	8:30	630	32	584	29	1782	89	296	15
8:30	8:45	681	34	690	35	1949	97	337	17
8:45	9:00	927	46	709	35	2416	121	454	23
Average Demand		704	35	615	31	1923	96	331	17
Total Demand		739		646		2019		348	

Table 3.9 Measure of Effectiveness between models (Scenario 1)

	Downstream MOE				Ramp MOE			System MOE		
	TTT s/km	AD sec/veh	AFR veh/hr	AS km/hr	TTT sec/km	AD sec/veh	AFR veh/hr	TTT sec/km	AD sec/veh	AFR veh/hr
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDI (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
					Percentage of change					
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

Table 3.10 Measure of Effectiveness between models (Scenario 2)

	Downstream MOE				Ramp MOE			System MOE		
	TTT s/km	AD sec/veh	AFR veh/hr	AS km/hr	TTT sec/km	AD sec/veh	AFR veh/hr	TTT sec/km	AD sec/veh	AFR veh/hr
ADI (1)	18650	0.63	2315	86.14	144976	2.67	679	58.59	24	3314
FLDI (2)	17718	0.78	2147	84.45	20122	10.71	651	62.96	32.71	3221
PFLDI (3)	18058	0.66	2220	85.24	14208	5.34	577	48.46	7.17	3284
					Percentage of change					
(2) vs. (1)	-5.00	23.81	-7.26	-1.96	38.80	301.12	-4.12	7.46	36.29	-2.81
(3) vs. (1)	-3.18	4.76	-4.10	-1.04	-1.99	100	-15.02	-17.29	-70.13	-0.91
(3) vs. (2)	1.92	-15.4	3.40	0.94	-29.39	-50.14	-11.37	-23.03	-78.08	1.96

Scenario 1 and 2 are considered as low traffic with Total Traffic Demand ranges from 3348 to 3752 vehicles per hour. According to the results obtained from the AIMSUN's simulation summary (Table 3.9 and 3.10), PFLDI outperformed the ADI and FLDI models. The PFLDI model reduced the system's total travel time by 16.12% (Scenario 1), 17.29% (Scenario 2) along with a decrease in the system's average delay time by 75.75% (Scenario 1), 70.13% (Scenario 2) in comparison to the ADI model. Moreover, the PFLDI improved the Scenario 1's downstream average speed (from 86.99 km/hr to 88.40 km/hr) and the average delay (reduce by 6.67%).

In Scenario 2 the PFLDI performs very well even though there is a slight decrease in the downstream average speed (from 86.14km/hr to 85.24 km/hr), average flow rate (reduce by 4.10%) and a 4.76% increase in average delay because as the on ramp traffic demand increases the PFLDI increases the ramp metering rate enabling more vehicles getting into the motorway causing a slight congestion in the downstream movements. However, the PFLDI reduced the system total travel time by 17.29% (from 58.59 hours to 48.46 hours) and average delay time by -70.13%.

3.4.7 Comparison for medium traffic demand

Similar to the low traffic demand, these scenarios are designed to evaluate the performance of the PFLDI under medium traffic demand conditions. The demand varies every 15 minute-period.

Table 3.11 Traffic demand data for Scenario 3 (4592 vehicles per hour) - Medium

Time (AM)		Constellation towards Upper Harbour		Upper Harbour towards Constellation		Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Car	HOV	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	779	39	678	34	1744	87	436	22
8:15	8:30	830	42	784	39	1982	99	496	25
8:30	8:45	881	44	890	45	2149	107	537	27
8:45	9:00	1127	56	909	45	2616	131	624	33
Average Demand		904	45	815	41	2123	106	531	27
Total Demand		949		856		2229		558	

Table 3.12 Traffic demand data for Scenario 4 (6272 vehicles per hour) – Medium

Time (AM)		Constellation towards Upper Harbour		Upper Harbour towards Constellation		Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Car	HOV	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	1179	59	1078	54	2144	107	836	42
8:15	8:30	1230	62	1184	59	2382	119	896	45
8:30	8:45	1281	64	1290	65	2549	127	937	47
8:45	9:00	1527	76	1309	65	3016	151	1054	53
Average Demand		1304	65	1215	61	2523	96	931	47
Total Demand		1369		1276		2649		978	

Table 3.13 Measure of Effectiveness between models (Scenario 3)

	Downstream MOE				Ramp MOE			System MOE		
	TTT	**AD**	**AFR**	**AS**	**TTT**	**AD**	**AFR**	**TTT**	**AD**	**AFR**
	s/km	**sec/veh**	**veh/hr**	**km/hr**	**sec/km**	**sec/veh**	**veh/hr**	**sec/km**	**sec/veh**	**veh/hr**
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDI (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
	Percentage of change									
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

Table 3.14 Measure of Effectiveness between models (Scenario 4)

	Downstream MOE				Ramp MOE			System MOE		
	TTT	AD	AFR	AS	TTT	AD	AFR	TTT	AD	AFR
	s/km	sec/veh	veh/hr	km/hr	sec/km	sec/veh	veh/hr	sec/km	sec/veh	veh/hr
ADI (1)	18650	0.63	2315	86.14	144976	2.67	679	58.59	24	3314
FLDI (2)	17718	0.78	2147	84.45	20122	10.71	651	62.96	32.71	3221
PFLDI (3)	18058	0.66	2220	85.24	14208	5.34	577	48.46	7.17	3284
					Percentage of change					
(2) vs. (1)	-5.00	23.81	-7.26	-1.96	38.80	301.12	-4.12	7.46	36.29	-2.81
(3) vs. (1)	-3.18	4.76	-4.10	-1.04	-1.99	100	-15.02	-17.29	-70.13	-0.91
(3) vs. (2)	1.92	-15.4	3.40	0.94	-29.39	-50.14	-11.37	-23.03	-78.08	1.96

3.4.8 Comparison for high and extreme traffic demand

Similar to the low traffic demand, these scenarios are designed to evaluate the performance of the PFLDI under medium traffic demand conditions. The demand varies every 15 minute-period.

Table 3.15 Traffic demand data for Scenario 5 (7952 vehicles per hour) – High

Time (AM)		Constellation towards Upper Harbour		Upper Harbour towards Constellation		Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Car	HOV	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	1579	79	1478	74	2544	127	1236	62
8:15	8:30	1630	82	1584	79	2782	139	1296	65
8:30	8:45	1681	84	1690	85	2949	147	1337	67
8:45	9:00	1927	96	1709	85	3416	171	1454	73
Average Demand		1704	85	1615	81	2923	146	1331	67
Total Demand		1789		1696		1809		1398	

Table 3.16 Traffic demand data for Scenario 6 (8783 vehicles per hour) – Extreme

Time (AM)		Constellation towards Upper Harbour		Upper Harbour towards Constellation		Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Car	HOV	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	1779	89	1678	84	2744	137	1436	72
8:15	8:30	1830	92	1784	89	2982	149	1496	75
8:30	8:45	1881	94	1890	95	3149	157	1537	77
8:45	9:00	2127	106	1909	95	3616	181	1654	83
Average Demand		1904	95	1815	91	3123	156	1531	77
Total Demand		2000		1906		2019		1598	

Table 3.17 Measure of Effectiveness between models (Scenario 5)

	Downstream MOE				Ramp MOE			System MOE		
	TTT s/km	AD sec/veh	AFR veh/hr	AS km/hr	TTT sec/km	AD sec/veh	AFR veh/hr	TTT sec/km	AD sec/veh	AFR veh/hr
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDI (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
					Percentage of change					
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

Table 3.18 Measure of Effectiveness between models (Scenario 6)

	Downstream MOE				Ramp MOE			System MOE		
	TTT s/km	AD sec/veh	AFR veh/hr	AS km/hr	TTT sec/km	AD sec/veh	AFR veh/hr	TTT sec/km	AD sec/veh	AFR veh/hr
ADI (1)	18650	0.63	2315	86.14	144976	2.67	679	58.59	24	3314
FLDI (2)	17718	0.78	2147	84.45	20122	10.71	651	62.96	32.71	3221
PFLDI (3)	18058	0.66	2220	85.24	14208	5.34	577	48.46	7.17	3284
					Percentage of change					
(2) vs. (1)	-5.00	23.81	-7.26	-1.96	38.80	301.12	-4.12	7.46	36.29	-2.81
(3) vs. (1)	-3.18	4.76	-4.10	-1.04	-1.99	100	-15.02	-17.29	-70.13	-0.91
(3) vs. (2)	1.92	-15.4	3.40	0.94	-29.39	-50.14	-11.37	-23.03	-78.08	1.96

3.5 SUMMARY

The PFLDI was evaluated by comparing it with the ADI and FLDI models. The ADI model is controlled by AIMSUN with built in ALINEA algorithm. These values are collected from the various detectors placed on the entire diamond interchange. The performances of the diamond interchange were measured based on the individual performance of the downstream, ramp, intersections and overall system. The performance of the model is based on the actual performance of the on ramp linked with the freeway (% change in Downstream Average Speed and % change in Downstream Average Delay) and the performance of the two intersections were based on the Delay Time. The System Total Travel Time is the overall performance of the ramp and intersections. The PFLDI's performance is measured by using the following Measures of Effectiveness (MOE) guidelines from Fang (2004) PhD thesis.

There are six scenarios tests that can be carried out with Total Traffic Demanding ranges from 3348 to 8783 vehicles per hour with varied flow rate. These scenarios have different traffic demand distribution ratios. The varied flow rate allows complex traffic situation which reflects real-life conditions. The scenarios covered most of the real life traffic conditions from free flow to congestion. In each scenario, there were ten simulation runs, one for each model. The total traffic demand for the interchange is approximately 3000-4000 veh/h. The traffic demand was purposely setup to be higher than the actual capacity to test the model under the extreme conditions where traffic demand is higher than the capacity (scenario 4, 5 and 6). The total volume includes the through traffic in the main highway / motorway / freeway.

According to Table 3.19, the PFLDI model gives smaller delays than the FLDI and ADI models when the traffic volumes are below 5000 veh/h. However, if traffic volumes are more than 5000 veh/h, the PFLDI gives higher delays than the ADI but performs better than the PFLDI. In Upper Harbour diamond interchange the total traffic demand in any given day does not exceed 3800 veh/h. The reason for the draw back in the performance of the PFLDI when traffic demand is greater than the road capacity is because the algorithm itself was designed to give higher priority to the traffic on the highways, thus causing the internal congestion at the two inner intersections. This leads to the increase in average delay of the vehicles at the intersections and on-ramp which increases the overall average delay time of the whole system. Table 3.20 shows the comparison results of Total Travel Time (TTT) between the three models for six scenarios with Total Traffic Demand ranging from 3348 to 8783 veh/h. When the total traffic demand ranges from 4592 to 7952 veh/h, the TTT generated from the FLDI and PFLDI models are much higher than the ADI model. Similar to the average delay time results, when the traffic demand exceeds the actual road capacity, the ramp metering flushes all the on-ramp vehicles to the highway causing extra delay for the traffic condition. This has to be done in order to prevent a further queue spillback on the internal intersections. From the results above, PFLDI generates lower TTT comparing to FLDI model.

Table 3.19 Average delay time comparison (second per vehicle)

Scenario	Traffic demand	ADI Model	FLDI Model	PFLDI Model
1	3348	21.48	5.32	4.18
2	3752	23.92	7.71	6.50
3	4592	41.62	17.39	15.34
4	6272	71.03	135.27	121.15
5	7952	88.94	149.86	137.65
6	8783	135.62	152.79	151.49

Table 3.20 Total travel time comparison (hour)

Scenario	Traffic demand	ADI Model	FLDI Model	PFLDI Model
1	3348	42.78	36.74	35.82
2	3752	58.69	62.96	48.52
3	4592	82.58	122.66	68.69
4	6272	135.45	210.17	201.23
5	7952	183.23	263.88	248.66
6	8783	285.86	276.43	265.82

Table 3.21 Downstream average speed comparison (km/hr)

Scenario	Traffic demand	ADI Model	FLDI Model	PFLDI Model
1	3348	86.82	88.33	91.36
2	3752	87.14	84.65	89.43
3	4592	84.78	82.19	86.12
4	6272	78.03	79.57	80.72
5	7952	73.95	78.12	78.82
6	8783	41.74	76.18	77.23

Table 3.20 and Table 3.21 show that PFLDI outperforms the FLDI and ADI models in terms of downstream average speed and reducing the average delay time. This is because the probabilistic fuzzy generates more accurate metering rate based on the current traffic condition, thus creating a smooth flow of traffic into the mainline highway.

Probabilistic Fuzzy Logic Diamond Interchange (PFLDI), Fuzzy Logic Diamond Interchange Metering (FLDI), Actuated Diamond Interchange Control (ADI) were built and tested in AIMSUN 6 environment with six different traffic scenarios ranging from free flow to congestion. From the results, it was found that PFLDI outperforms the ADI and FLDI models in system total travel time, downstream average delay and average speed even when the traffic demand exceeds the road capacity (including arterial roads, ramp and motorway).

The ADI model becomes less effective once the total traffic demand is much higher than road capacity. The percentage of change in the system total travel time, downstream average delay and average speed (six scenarios) has shown the reason why PFLDI model has better performance the other models. The PFLDI and FLDI models were designed to avoid the formation of congestion and in extreme traffic conditions they also prevent the congestion from getting worse.

CHAPTER 4
CYCLIST SAFETY AND PLANNING

Little attention has been given in recent years to the delays experienced by cyclists in urban transport networks. When planning changes to traffic signals or making other network changes, the value of time for cycling trips is rarely considered. The traditional approach to road management has been to only focus on improving the carrying capacity relating to vehicles, with an emphasis on maximising the speed and volume of motorised traffic moving around the network. The problem of cyclist delay has been compounded by the fact that value of time figures for cyclists has been lower than those for vehicles, which affects benefit–cost ratios and effectively provides a disincentive to invest in cycling issues compared with other modes. The issue has also been influenced by the way in which traffic signals have been set up and operated. Because the primary stresses on an intersection tend to occur during vehicle (commuter) peaks in the morning and afternoon, intersections tend to be set up and coordinated to allow maximum flow during these peaks. The result is that during off-peak periods there is often spare capacity that is underutilised. Phasing and timing set up for peaks may not provide the optimum benefits during off-peak times. This is particularly important to cyclists during lunch-time peaks, when vehicle volumes are low and cyclist volumes are high. Cyclists can end up waiting long periods of time as a result of poor signal phasing, rather than due to the demands of other road users being placed on the network.

This research focuses primarily on signalised cyclist crossing with the intention of identifying how operational strategy changes could improve the level of service for cyclists.

4.1 Safety and Compliance

4.1.1 Safety

The relationship between cyclist delay and cyclist safety is a complex one. Cyclists who feel they are faced with unreasonable delays will use their own judgement as to when it is safe to cross. However, an increase in delay does not necessarily result in an increase in non-compliance, as high volumes of traffic can act as a deterrent to non-compliance at

signalised intersections. Conversely, low cyclist waiting times do not guarantee universal compliance with traffic signals. It is therefore difficult to draw a linear relationship between delay and resulting non-compliance. It is also difficult to find an exact correlation between non-compliance and injury, as cyclists unlawfully crossing the road will use their judgement to determine a safe crossing, and it is only when that judgement is in error that an injury or fatality might occur. Only a small portion of non-compliant activity results in an injury or fatality. However, as traffic volumes increase, any non-compliant behaviour becomes inherently riskier.

4.1.2 Compliance

The Manual on uniform traffic control devices (Federal Highway Administration 2009) observes that traffic control signals are often considered a 'panacea' for all traffic problems at intersections; however, simply installing signals does not guarantee a favourable outcome, particularly for low vehicle volumes. Cyclist signals, if poorly operated, can create unnecessary delay. Various studies have found that cyclists are more flexible in their regard for road rules than other mode types. Cyclists tend to use traffic signals as a guide, but if they become frustrated by long delays, they will likely ignore the signals entirely and cross when they perceive the risk to be acceptable, rather than accept continued delay. Thus, cyclist signals have a higher non-compliance rate than vehicle traffic signals (and potentially, a much lower enforcement rate). Therefore, it is possible to infer that the primary measure of whether a set of signals is functioning adequately for cyclist traffic would be the rate of non-compliance. Non-compliance to traffic signals presents a risk to the cyclist and other road users, and as a result, frustration at cyclist delay quite quickly translates into a road safety issue. Much of the literature reviewed considered cyclist delay entirely from a compliance/safety perspective, rather than as a factor in overall cyclist travel times. Ishaque and Noland (2007) found that:

Cyclist non-compliance behaviour is encouraged by signal timings that are not favourable to them. This is the case both when a disproportionately large amount of time is made available to vehicular traffic and when cyclist volumes are such that they do not fit into the time provided for by the cyclist phase. Long signal Bicycles may pose a safety hazard for

cyclists and therefore one of the most effective measures to increase cyclist safety and compliance is to make traffic signals as good as possible for cyclists and that is by minimising their waiting times.

They also reported on several European studies on the topic of cyclist perception of delay. One two year study in London found that at controlled crossings, 30–40% of cyclists felt annoyed when the delay was in the range of 6–22 seconds, but more than 70% felt annoyed when the delay was longer than 26 seconds. Another study on children and adults showed that 30 seconds was the maximum that both children and adults were willing to wait at a signalised intersection, and similar findings in Germany had led to the German Highway capacity manual (Institute for Traffic Engineering, Ruhr-University 2000) specifically recommending that signal Bicycle times longer than 90 seconds should be avoided. Ishaque and Noland (2007) found that cyclist non-compliance was encouraged by timings that were unfavourable to them, and that long Bicycle times could therefore pose a safety hazard that could be avoided through reducing signalised delays. This is significant in New Zealand, because the intersections observed, had Bicycle times longer than 90 seconds, and average cyclist delays of more than 30 seconds.

The UK Traffic advisory leaflet 5/05 (Department for Transport 2005) also raises the issue of signal compliance and reasons for some cyclists being more willing to take risks:

Cyclists compliance with the Red signal is thought to be generally poor. Cyclists are more likely to disregard the Red signal if they consider the distance they have to cross, or the time they have to wait, is unreasonable. (When waiting at a junction in bad weather, a driver may be frustrated but is generally warm and dry. A frustrated, cold and/or wet cyclist is more likely to take what otherwise they would consider an unacceptable risk).

Part 7 in the AUSROADS Guides to engineering practice – traffic signals (1994) identifies that drivers and cyclists will disobey a red signal if delays are abnormally long. The guidelines suggested that a maximum waiting time of 20 seconds would be tolerated in light traffic, and 120 seconds in heavy traffic. This is much longer than the time suggested in literature from other sources, and longer than what was considered acceptable by cyclists

interviewed during this research. As a general rule, signals engineers try to aim for a total Bicycle of 120 seconds or less, where possible.

Australian research conducted by Daff et al (1991) showed that heavier traffic flows in more recent times seem to have forced more compliance at intersection signals, though this is likely to be due to perceived safety issues rather than a greater desire to comply with signalised delays. In an effort to improve compliance, some signal operators have introduced countdown timers. The literature reviewed universally found that excessive delay would lead to cyclist frustration.

High traffic volumes reduce the risk of non-compliant behaviour, as the perception of risk is greater. Where volumes are low and delays are long (e.g. outside of vehicle 'peak' periods) then cyclists are more likely to ignore the signals. The dangers associated with this can be aggravated by a number of factors, including:

- The visibility of cyclists vs. other visual distractions (such as oncoming traffic)
- The presence of heavy vehicles, which can have significant blind spots in their field of vision and also are harder to slow down than cars, and may therefore be less able to avoid cyclist than the cyclist expects).

It is important, therefore, for road controlling authorities to consider cyclist delay, and in particular to consider off-peak conditions when setting up signal phasing and timing.

4.2 New Zealand Bicycle-vehicle crashes statistics (2004-2008)

Table 4.1 shows the number of Bicycle–motor vehicle crashes, by type, at the selected intersections (Turner et al 2012)

Table 4.1 Bicycle–vehicle crashes at selected sites (2004–2008)
(Excerpt from: Turner et al 2012)

Crash type	Number of crashes (NZ Sites)	Number of crashes (Melbourne sites)
Right-angle (NZ type HA)	15	1
Right-turn-against (cyclist going straight) (NZ type LB)	28	2
Right-turn-against (cyclist turning right) (NZ type LB)	5	0
Left-turn side swipe (NZ type CB, AC)	8	N/A
Other	38	12
Total injury crashes	**94**	**15**

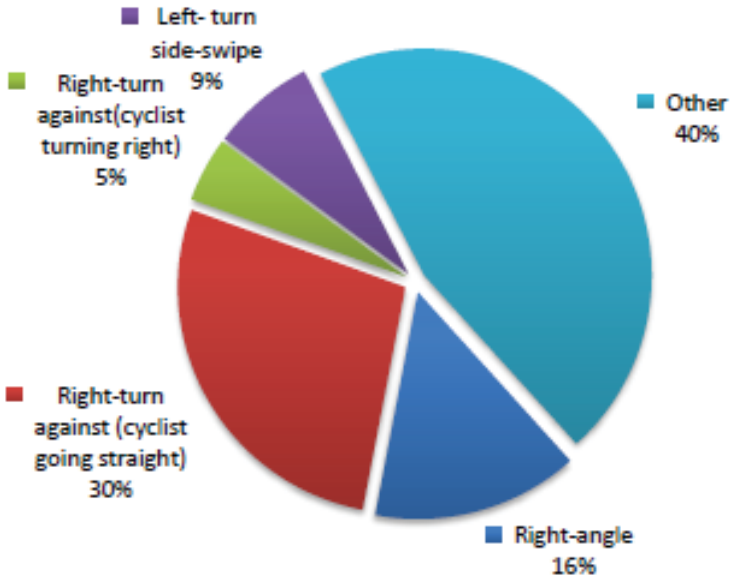


Figure 4.1 Proportion of Bicycle-motor vehicle crashes, by type (2004-2008)
(Excerpt from: Turner et al 2012)

4.3 New Zealand cyclist safety statistics

In the ten-year period from 1993 to 2002 there were 7354 cyclists reported as injured and 139 killed (LTSA 2003a).

As shown in Table 3.3, since 1970, cyclist injuries have climbed to a peak in 1988, then declined, with some increases recently. In 2001 the number of cyclists injured per 100,000 population increased to 18.1 from 14.6 in 2000 (LTSA 2002). It increased further to 19.6 in 2002.

Table 4.2 New Zealand pedal cyclist casualties and population statistics – historical, year ending 31 December (Excerpt from: LTSA 2002)

				Per 100,000 Population	
Year	Population	Injured	Killed	Injured	Killed
1983	3,269,500	900	19	27.5	0.6
1984	3,299,500	958	31	29.0	0.9
1985	3,311,200	1106	21	33.4	0.6
1986	3,316,700	1012	22	30.5	0.7
1987	3,349,100	1051	18	31.4	0.5
1988	3,356,200	1081	20	32.2	0.6
1989	3,384,510	1051	20	31.1	0.6
1990	3,429,100	1054	27	30.7	0.8
1991	3,449,700	1000	22	29.0	0.6
1992	3,485,400	941	17	27.0	0.5
1993	3,524,800	910	17	25.8	0.5
1994	3,577,200	882	15	24.7	0.4
1995	3,643,200	813	15	22.3	0.4
1996	3,717,400	754	13	20.3	0.3
1997	3,761,100	724	12	19.2	0.3
1998	3,790,900	626	16	16.5	0.4
1999	3,810,700	619	8	16.2	0.2
2000	3,830,800	559	19	14.6	0.5
2001	3,850,100	696	10	18.1	0.3

As shown in Table 4.2, since 1983, cyclist injuries have climbed to a peak in 1988, and then declined, with some increases recently. In 2001 the number of cyclists injured per 100,000 population increased to 18.1 from 14.6 in 2000 (LTSA 2002). Cyclist deaths have generally been declining (Figure 4.2). The death and injury rate per 100,000 population has been dropping since 1983 (as shown in Table 4.2), although there have been increases in 2000 and 2001. However, the decreasing injury rate may be related to decreasing cycling distances which would reduce the exposure per head of population.

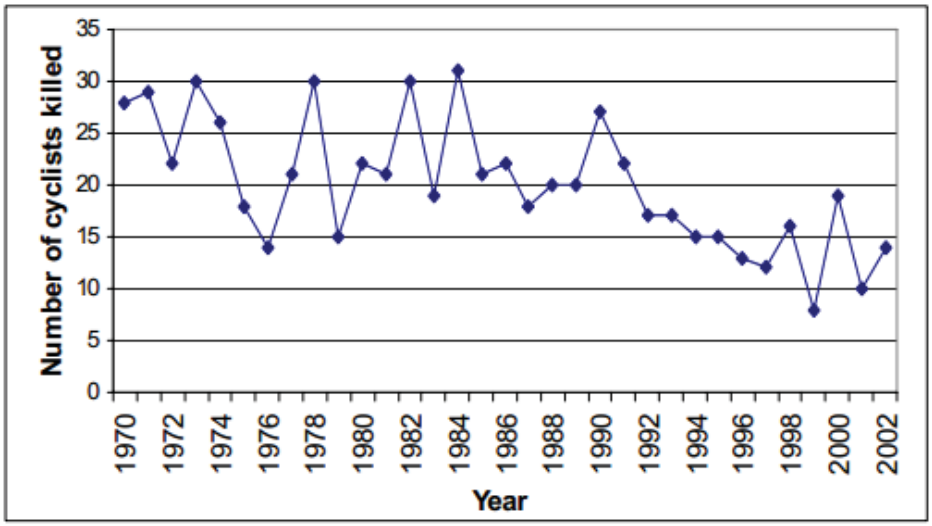


Figure 4.2 Number of cyclists killed in NZ per year, 1970 – 2002 (LTSA 2003a)

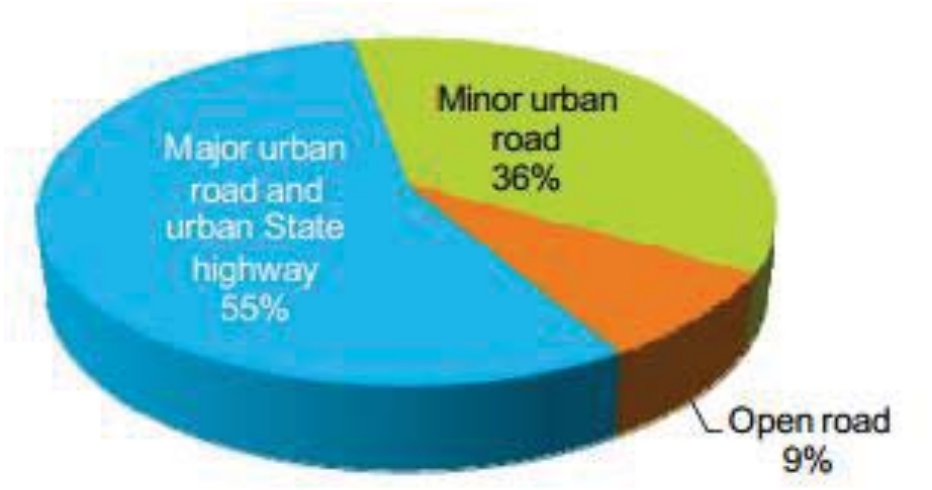


Figure 4.3 Percentage of cyclist deaths and injuries in motor vehicle crashes by road type (2007-2011)

Approximately nine in every ten reported cyclist casualties (2007–2011) occurred on urban roads (roads with a speed limit of 70km/h or less). Furthermore, over half of all cyclist casualties occur on major urban roads (typically busy arterials), rather than on the minor urban roads that usually provide access to abutting properties. Over half of all cyclist fatalities occur on the open road, due to the high impact speeds associated with crashes on the open road.

4.4 Types of crash

In New Zealand, Three specific crash movements each account for more than 10% each of all cyclists deaths or injuries in police-reported crashes involving motor vehicles.

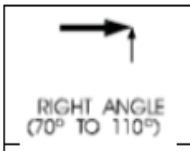
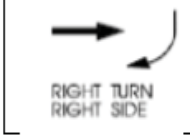
	Crossing (No Turns)	13.8%	This crash type involves a collision at a right angle, typically when both parties involved are moving straight through an intersection.
	Right Turn Against	15.5%	Approximately 88 percent of this crash type involves another vehicle turning in front of the cyclist.
	Crossing (Vehicle Turning)	11.3%	Approximately 81 percent of this crash type involves another vehicle turning in front of the cyclist while crossing an intersection.

Figure 4.4 Types of crash in New Zealand (Excerpt from: MOT Bicycle crash fact sheet 2012)

Cyclists have primary responsibility in only 23% of all cyclist-vehicle crashes in which they are injured or die. As the severity of the crash increases, there is a slight increase in the proportion of cyclists found to have the primary responsibility. Of the cases where the cyclists are found to have primary responsibility, 40% of the at-fault cyclists failed to give way and 27% of the at-fault cyclists did not see the other party. In the cases where the vehicle drivers are found to have primary responsibility in a crash involving a cyclist, 61% of the drivers in fatal or injury crashes failed to give way or stop and 59% did not see the other party. Fifteen per cent were inattentive or their attention was diverted. This rises to 36% of the at-fault drivers in fatal crashes involving cyclists being inattentive or their attention being diverted.

The faster drivers are going, the more difficult it is for them to avoid hitting a cyclist in their path. An alert driver travelling at 50km/h will travel 37 metres after reaction/braking before coming to a complete stop. The same driver travelling at 100km/h will move 5 metres further than this before even reacting and, once braking has started, will travel a further 69 metres before coming to a complete stop. The speed at which cyclists are struck is important in determining the likelihood of death 4. The risk that speed poses to more vulnerable cyclists, such as the elderly and children, is likely to be even higher due to their natural fragility 5.

4.5 Bicycle Planning

According to GTEP14 (2008):

1. Roads with higher traffic speed and traffic volumes are more difficult for cyclists to negotiate than roads with lower speeds and volumes. The threshold for comfort and safety for cyclists is a function of both traffic speed and volume, and varies by cyclist experience and trip purpose.

2. When school cyclists are numerous or the route is primarily used for recreation then path treatments may be preferable to road treatments.

3. Provision of a separated Bicycle path does not necessarily imply that an on-road solution would not also be useful, and vice-versa. Different kinds of cyclists have different needs. Family groups may prefer off-road Bicycle paths while racing or training cyclists, or commuters, tend to prefer Bicycle lanes. Figure 4.5 illustrated the choice of facility type for cyclist and Figure 4.6 shows how the vehicle positions on the road with Bicycle lanes.

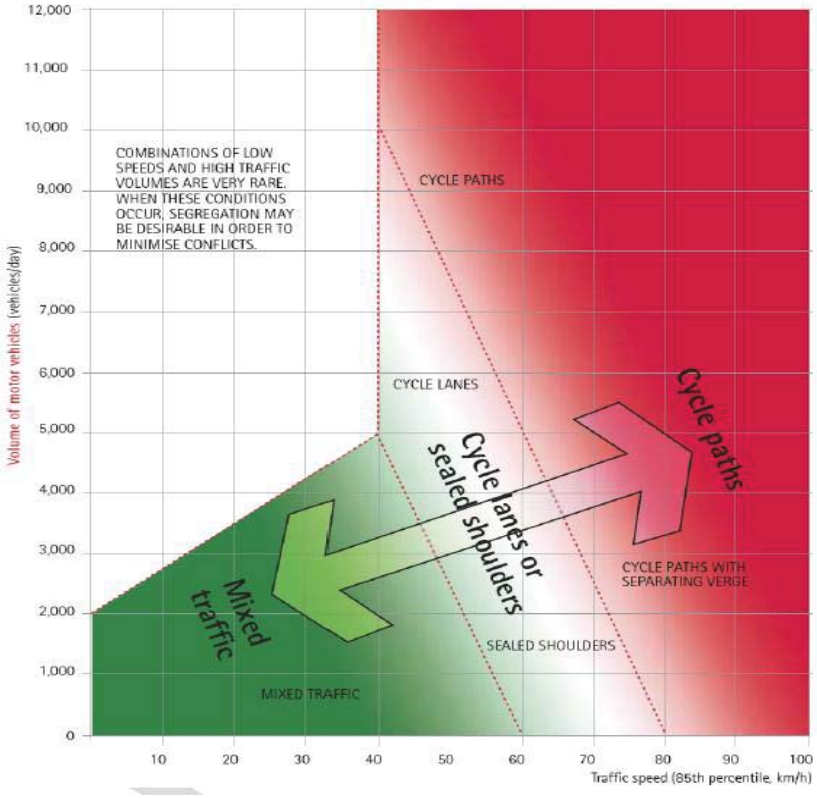


Figure 4.5 Guide to Choice of Facility Type for Cyclists (Excerpt from: GTEP14)

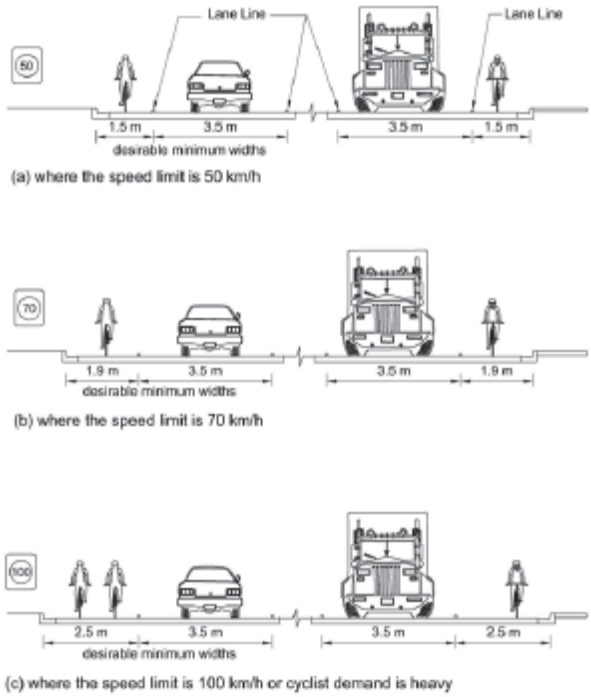


Figure 4.6 Vehicle positions on road carriage way associated with Exclusive Bicycle Lanes (Excerpt from: GTEP14)

4.6 Current Bicycle Infrastructure in Auckland Region

Figure 4.7 Cyclist holding area (Albany)

Figure 4.8 Typical Bicycle lane design at intersection (Albany)

Figure 4.9 Bicycle lane – car park design (Albany)

Figure 4.10 Bicycle with protected lanes at Custom Street, Auckland CBD

Figure 4.11 Bicycle with protected lanes overlaping with vehicle left-hand-turning at Custom Street, Auckland CBD

Figure 4.12 Cyclist traveling across the intersection at Custom Street, Auckland CBD

4.7 BICYCLE DESIGN GUIDELINES

4.7.1 Major Bicycle way through intersections with Barnes Dance layout

According to Christchurch Bicycle Design Guidelines 2013 (CCDG 2013), Controlled intersections and T-intersections are challenging for cyclists. Major Bicycle-ways that cross these intersections need to be designed to protect the cyclist and provide a greater level of comfort. Where possible, roundabouts should be avoided. A Dutch intersection or a Bicycle Barnes dance offers the highest level of protection and comfort at intersections.

Figure 4.13 Dutch intersection with Bicycle Barnes dance (CCDG 2013)

These designs (Figure 4.13) are a new concept in New Zealand and will need to be trialled before wider use. In the meantime, other intersection treatments such as protected Bicycle lanes provide a level of increased safety and comfort for cyclists.

4.7.2 Major Bicycle way through intersections with adapted Dutch design

The Dutch intersection (Figure 4.14) is a new approach to intersection design in Christchurch and potentially offers the highest level of protection to cyclists. The design principles are:

- Dutch intersection designs are appropriate where separated Bicycle paths approach an intersection. The design features corner islands to provide separation between cyclists and vehicles at the intersection. This separation also improves inter-visibility between drivers and cyclists. The cyclist crossing facilities and signals are separate from the Bicycle path.

- The size of the corner island is variable depending on the size and angles of the intersection (corner splay). The corner islands size can help to slow turning vehicles which also improves safety for both cyclists and cyclists.

- The design of the intersection needs to consider left turning vehicles. To allow larger vehicles to turn safely, the stop line for the entry lane may need to be set back. If this is needed, the intersection capacity and the cyclist crossing placement can be affected.

- Green coloured surfacing can be used to improve the visibility and legibility of the Bicycle path on the approach and through the intersection.

- Ideally the design should include separate bollard-style Bicycle signals that provide a countdown to the Bicycle crossing phase. Signal phasing can also incorporate advanced Bicycle starts or exclusive vehicle turning phases to reduce the conflict.

- All Dutch intersection designs must be officially trialled and monitored in agreement with NZTA before implementation across the major Bicycleway network.

- Designing traffic signals is a specialised discipline. All designs need to engage a signal engineer for both the design and peer review.

Figure 4.14 The adapted Dutch Bicycle lane layout at an intersection (CCDG, 2013)

4.7.3 Major Bicycle-ways – intersection with protected Bicycle-ways

The design principles for protected Bicycleway at intersection are:

- Protected Bicycle lanes offer the cyclist improved protection by providing temporary separation from vehicles on the approach to the intersection. This is especially recommended at known conflict points such as left-turning traffic lanes.
- Temporary separation can be achieved by vertical edge markers (such as uprights), raised delineators (such as rumble strips or small kerbs) or painted chevrons.
- Where a protected Bicycle lane is introduced it is important not to reduce sightlines of cyclist crossings and any vertical edge markers need to be carefully maintained.
- Introducing an exclusive Bicycle signal phase or delaying the left-turning and/or the on-coming right-turning vehicles to allow cyclists a head start at intersections can provide further priority and safety for cyclists.

Figure 4.15 Example of a vertical edge marker

Figure 4.16 Intersection with protected Bicycle-ways (CCDG 2013)

- Signalised Intersections with SCOOT loop. Bicycle detectors are commonly square, rectangular or elongated

CHAPTER 5
PROBABILISTIC FUZZY LOGIC CONTROL AT DIAMOND INTERCHANGE INCORPORATING BICYCLE SIGNAL

Conflicts between cyclists and turning or merging vehicles at intersections and interchanges are major threats to bicycle safety. In a diamond interchange, the problem is less complex as cyclists move in one direction each side of the interchange. This chapter addresses design practices and probabilistic fuzzy signal control to minimise the threat for cyclists.

5.1 METHODOLOGY

The diamond interchange at the Upper Harbour Interchange in Auckland (Figure 3.1) was chosen to investigate the performance of PFLDI and PBSC. Similar to the model developed in Chapter 3. Most of the factors, assumptions and parameters remain the same.

5.1.1 Detectors placement

This system has two vehicle detectors on each approach side of the crossing. One vehicle detector is placed at the crossing while the other detector is placed 60-80 meter from the crossing stop line (Figure 5.1). These two detectors will provide the number of vehicles approaching the intersection on one particular movement. The other detectors will be placed on the bicycle lane. Since the bicycles cannot get on the motorway, the movement in this case is simple. The bicycle only needs to travel across the intersections (red arrows in Figure 5.1) and the approaching vehicles are the left or right hand turning of the intersections within the diamond interchange.

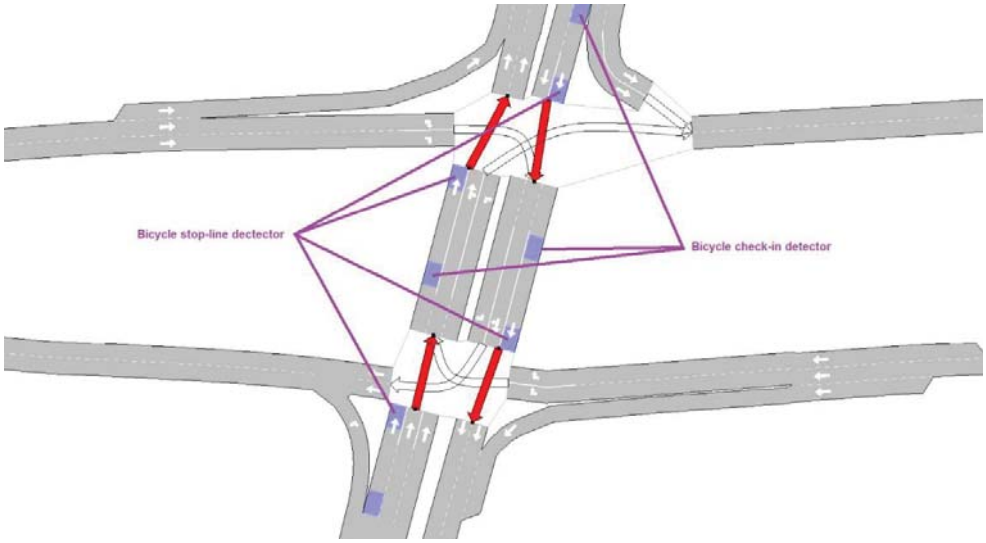


Figure 5.1 Bicycle detectors placement

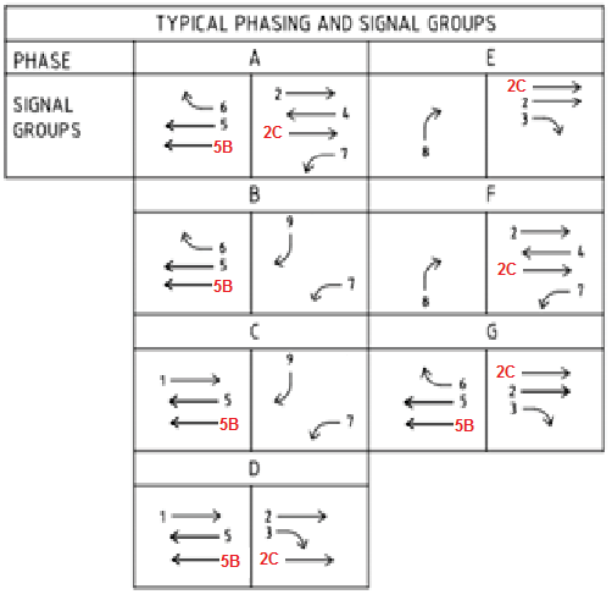
5.1.2 Bicycle crossing request button

Bicycle crossing request button (Figure 5.2) is necessary in the event that the detectors failed to detect the presence of the cyclist in the holding area.

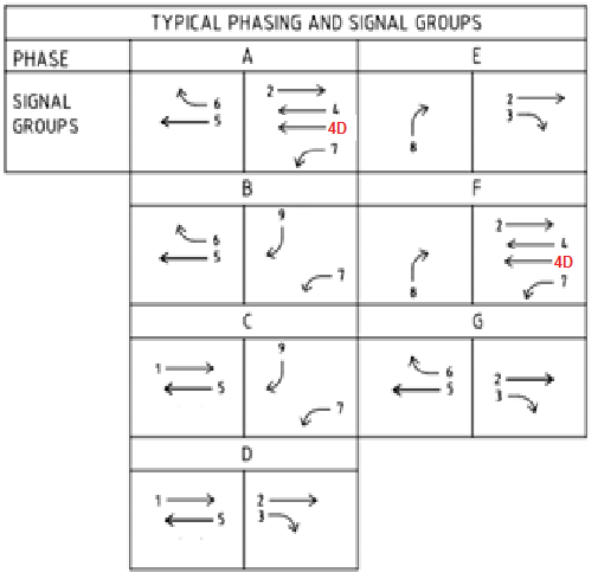
Figure 5.2 Similar cyclist crossing request button (Auckland CBD)

5.1.3 The Bicycle Signal and Phasing

The traffic signal control is almost the same in Chapter 3 with the introduction of new movements 2C, 5B and 4D for cyclist movements across the interchange. The existing traffic signal control reacts to the traffic condition by triggering different pre-defined Bicycle time.



(a)



(b)

Figure 5.3 (a) (b) Phasing and Signal Groups at Upper Harbour Interchange with Bicycle signals

The phase sequence for A, B, C, D, E, F and G are similar to Figure 3.4 in Chapter 3 with the exception of the additional bicycle signals. The signal operates as follows: when no bicycle wishing to cross the intersection, the signal phase is green for the vehicle traffic. When the detector detects the bicycle, the controller will count the total of arriving vehicles and the waiting time of the cyclists. If specified conditions are met, the current phase is terminated and a green phase is provided for the cyclists otherwise the current phase will continue as defined in the phase plan.

Table 5.1 Signal timing data for PFLDI with Bicycle Signal

		Actuated Control			
Phase	B. Min. Green*	B. Max. Green*	V. Min. Green**	V. Max. Green**	V. Yellow
B	15	40	6	28	4
D	15	40	6	48	4
F	15	40	6	17	4
		PFLDI with Bicycle Signal			
B	15	40	6	6-30	4
D	15	40	6	6-48	4
F	15	40	6	6-18	4

**B. Min. / Max. Green stands for Bicycle Minimum / Maximum Green time*

***V. Min. / Max. Green / Yellow stands for Vehicle Minimum / Maximum Green / Yelow time*

According to the site survey done on 28/08/2013, the 15 seconds minimum green time is sufficient for a cyclist to cross the intersection with average acceleration of 2.2 m/s^{-2}

5.1.4 Bicycle Probabilistic Fuzzy Signal Control (BPFSC)

Similar to Chapter 3, this part of the research on bicycle at a diamond interchange, the arrival of bicycles from one arterial street follows the Poisson distribution. For saturated traffic flow, the probability, $P(x_c)$, that x_c bicycles arrive at any interval in the two-way intersection is calculated by formula (9) below:

$$P(x_C) = \frac{m^{x_C}\ e^{-m}}{x_C!} = \frac{I}{x_C!}\left(\frac{Vt}{3600}\right)^{x_C}\ e^{-\frac{Vt}{3600}} \qquad (9)$$

where:

x_c bicycles arrival at any interval

t is the length of time interval in seconds

V is hourly traffic volume (AIMSUN output for our simulation)

m is the average number of vehicles per interval (Vt/3600)

5.1.5 Design of Probabilistic Fuzzy Logic Control for a Diamond Interchange incorporating with Bicycle Signal (PFLDIBC)

The framework of PFLDIBC and PFLDI is illustrated in Figure 5.4. The system starts off by checking the presence of the cyclists at the diamond interchange in both direction. If there is no cyclist then the system will generate the vehicle extension green time using PFLDI algorithm. In contrast, PBSC will be turned on to generate the Bicycle Green Time Extension. After the completion of this cycle, the PFLDI will take over and generate the appropriate Vehicle Green Time Extension (Figure 5.5).

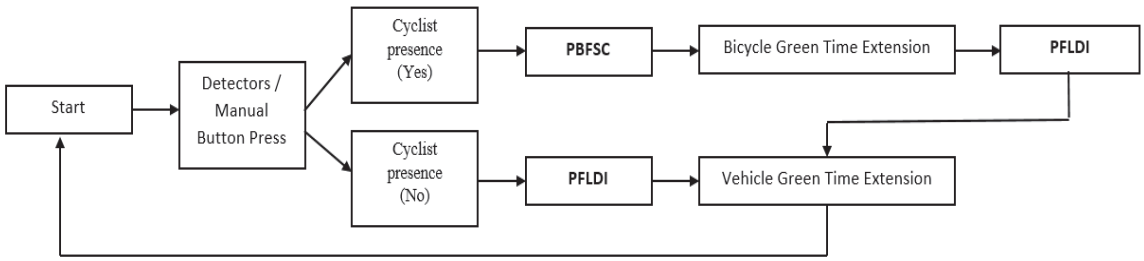


Figure 5.4 The framework of PFLDIBC and PFLDI algorithms at a diamond interchange

Figure 5.5 Green Time Extension for Bicycle and Vehicle

5.1.6 Probabilistic Fuzzy Variables and parameters

For a probabilistic fuzzy algorithm, the variables are divided into two groups: input and output. The input variables in this case are defined as follows:

- Total waiting time of one or more cyclists from the last signal change (CWT).
- The total approach vehicles are measured based on the number of vehicles detected between the two detectors (AV).
- Headway time between the two vehicles (HT). Smaller HT value equals to high flow density).
- The output variable is either green time extension (GTE) or move to the next phase (terminate the current phase) (TP).

5.1.7 Objectives of the Probabilistic Fuzzy Logic Algorithm

- Minimize cyclist waiting time – The cyclists wishing to cross the intersection must be accommodated as soon as possible.
- Minimize delay to vehicle movements – Vehicles should not be held for a long time.
- Maximum safety to the vehicles and cyclists – when a group of vehicles is approaching – When a group of vehicles approaching the intersection, the group should be given the priority to pass. This will avoid the collision with the cyclists and creating a gap for the cyclists at downstream.

5.1.8 Probabilistic Fuzzy Rules Base

As mentioned earlier in previous sections, the general format of the rules is very similar. The CWT is divided into three fuzzy sets: shot, long, very long. AV is divided into very few, moderate, and many. HT is divided into small and large.

Table 5.2 Rules base for PFLDIBC

RULE	IF			THEN
	CWT	AV	HT	EXTENSION
1	Short	Very Few	Small	TP
2	Long	Very Few	Small	TP
3	Very Long	Very Few	Small	TP
4	Long	Moderate	Small	TP
5	Very Long	Moderate	Small	TP
6	Very Long	Many	Small	TP
7	Very Long	Very Few	Small	TP
8	Very Long	Many	Large	TP
9	Short	Moderate	Large	GTE
10	Short	Many	Small	GTE
11	Long	Many	Small	GTE
12	Short	Very Few	Small	GTE
13	Long	Very Few	Large	GTE
14	Short	Moderate	Large	GTE
15	Short	Many	Large	GTE
16	Long	Many	Large	GTE

Similar to Fuzzy Phase Timing algorithm, in this case, there is 15 seconds fixed green time for cyclists and a minimum of 6 seconds for vehicle on the road. The controller will check the detectors every second and choose the appropriate rule to execute. Before terminating the current phase, the phase will run for the minimum time of 15 seconds.

The PFLDIBS module optimizes the green time extension rate considering the traffic flows, cyclist movements and other factors. The input to this module is acquired by the loop detectors installed at different locations, as stated in the preceding section. The input variables (Table 5.2) of the PFLDIBS are fuzzy and probabilistic, described by membership functions as shown in Figure 3.7 in Chapter 3, where the symbol "*P*" is the probability of a fuzzy linguistic variable. Thus, a variable in the PFLDIBS is characterized by a quintuple,

For example, the variable *CWT* is characterized by a quintuple, *("CWT", {(<"Short","0.2">, <"Long","0.2">, <"Very Long", "0.2">)}, [0, 100 seconds)], G, M).*

The membership functions of CWT, AV and HT are probabilistic variables with fuzzy rules which can be represented as (10)

$$Tv_c = \{A_{C1} / P_{C1}, A_{C2} / P_{C2,}, A_{C3} / P_{C3}\} \quad (10)$$

where:

A_{C1}, A_{C2}, A_{C3} are fuzzy variables in formula 10 (corresponds to Short / Very Few / Small, Long / Moderate and Large / Very Long / Many) with probabilities P_{C1}, P_{C2}, P_{C3} so that being

$$P_{C1} + P_{C2} + P_{C3} = 1 \quad (11)$$

The output to the PFLDIBS (11) is the *Green Time Extension* ranges from 15 – 40 seconds.

5.2 USING SIMULATION TO EVALUATE THE PFLDIBC ALGORITHM

5.2.1 Simulation Evaluation Procedure

The simulation evaluation procedure is shown in Figure 5.6. The values of calibrated parameters are the same for all comparisons cases such as driver behaviour assumptions, vehicle characteristics, traffic demands, phasing, etc. This study compares the performance of the entire diamond interchange from three signal control algorithms (PFLDI, FLDI and Actuated).

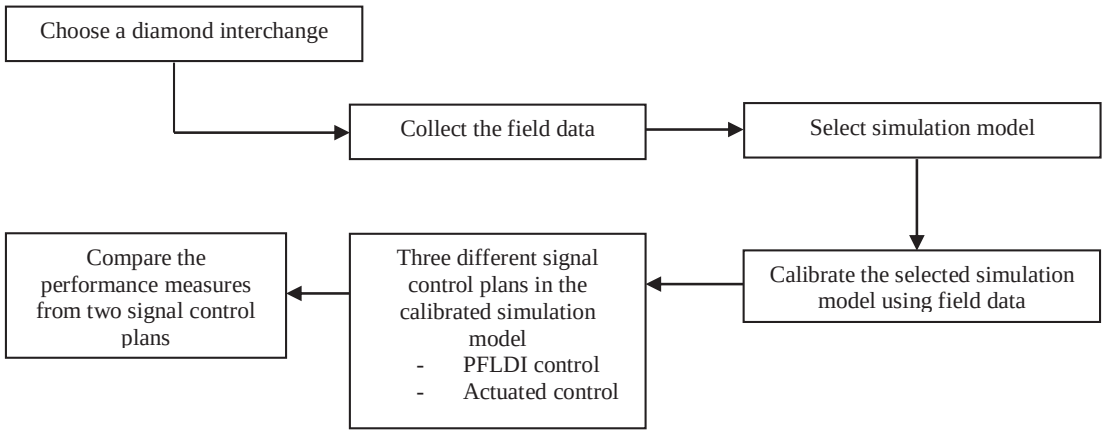


Figure 5.6 Simulation evaluation procedures

5.2.2 Calibration of Selected Simulation Model

The selected simulation model, in this case Upper Harbour Diamond Interchange, has been calibrated using field data surveyed and historical data obtained from ATOM, New Zealand Transportation Agency and North Shore City Council. According to Elefteriadou et al. (1997), calibration is the process of quantifying model parameters using real-world data in the model logic are described in Chapter 3.

The calibration of the diamond interchange for evaluating the PFLDIBS algorithm and comparison study is based on the field data of a site obtained from New Zealand Transportation Agency and North Shore City Council. The data used in this study includes

- Video of vehicle arrivals, queues, on/off ramps and motorways
- Cyclists data supplied by local council
- Geometric layout as shown in Figure 3.18
- Signal timing plan
- Traffic demand
- Other relevant information

The average flow rates, speed, occupancy and heavy occupancy vehicles data were obtained from ATOM. The turning rates were obtained from video recording of the study site. The traffic turning rates are similar to Chapter 3.

Parameters below (Figure 5.6 and Table 5.3) are modified from their default values in AIMSUN for calibration:

Table 5.3 Calibration data for AIMSUN simulation model

Parameter in AIMSUN	Maximum (calibrated / default)	Minimum (calibrated / default)
Vehicle maximum speeds (kph)	50 / 60	40 / 50
Vehicle maximum acceleration (m/s)	2.2 / 2.5	2.2 / 2.5
Arterial roads speed limit (kph)	50 / 60	40 /50
Ramp speed limit (kph)	100 / 120	80 / 100
Motorway speed limit (kph)	100 / 120	80 / 100
Arterials turning speed limit (kph)	50 / 60	40 / 50
Ramps turning speed limit (kph)	50 / 60	40 / 50

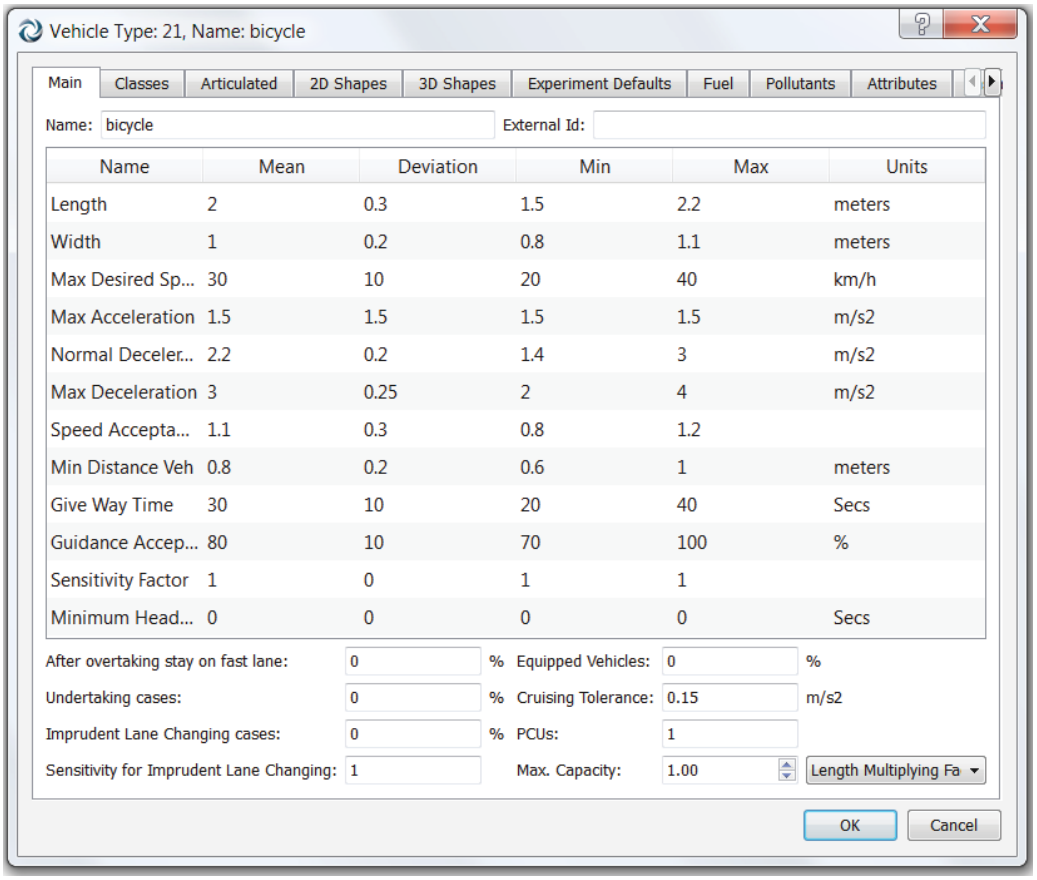


Figure 5.7 Bicycle parameters setup in AIMSUN

5.2.3 Simulation Assumption

In this study, Upper Harbour Diamond Interchange is used as a testing site for the PFLDI model implementation. The simulation model is based on the following assumptions:

- No bus is used in this simulation (starting from mid-2007, buses travelling to and from Auckland CBD operate in their own lanes which are independent from the motorway)
- The simulation will mimic the real-world peak hour period from 8:00-9:00 on Monday morning

5.2.4 Dynamic scenario setup

- Simulation type: microscopic simulator
- Detection interval: 6 seconds

- Statistics recording: every minute
- Traffic demand: 08:00 – 09:00 AM
- Simulation day: Tuesday
- Weather: dry
- Season: winter
- Control plans: actuated and fuzzy (API)
- Car following model: Deceleration estimation (Leader deceleration)
- Lane changing model:
- Percent overtake: 90%
- Percent recover: 95%
- On ramp model: cooperative mode looking gaps upstream
- Route choice model: fixed using Travel Time in Free Flow models
- Reaction time for cyclist: 0.75 s
- Reaction at stop for cyclist: 1.35 s

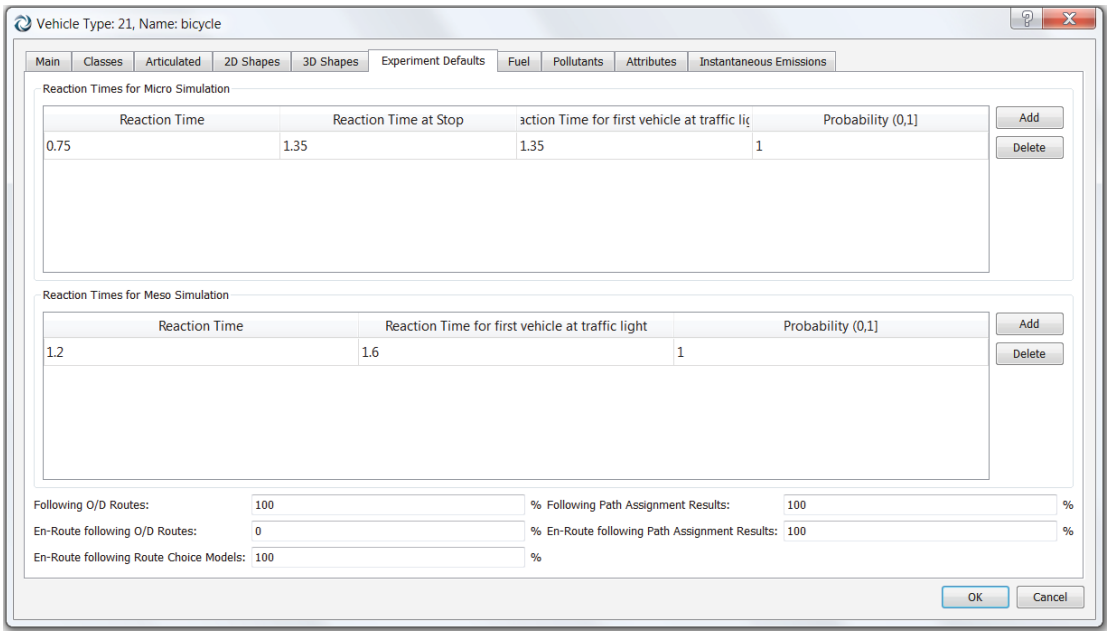


Figure 5.8 Cyclist reaction parameters setup in AIMSUN

5.2.5 Simulation of the PFLDIBC Algorithm in AIMSUN

As mentioned in Chapter 3, AIMSUN is well known for its powerful GETRAM extension module (part of the Application Programming Interface) which can be used to implement advanced traffic control applications such as FLDI, PFLDIBS and actuated control etc. The simulations of the PFLDIBS Algorithm in AIMSUN via GETRAM Extension are similar to those mentioned in previous chapter.

5.2.6 Performance Measure of Effectiveness

Similar to the PFLDI, the PFLDIBS performance is measured by using the Measures of Effectiveness **(MOE)** mentioned in Chapter 3 which consists of Motorway MOE, Ramp MOE and System MOE.

5.2.7 Traffic Demand Information

Six traffic demands are used in this simulation ranging from 2924 to 8831 vehicles per hour. These demands are used to simulate traffic congestion during free flow, medium traffic and congestion which are replications of traffic during weekdays and weekend the Upper Harbour Diamond Interchange. The data for the traffic demand is obtained from Gravita's Auckland Regional Manual Cycle Monitor 2014, Transit New Zealand State Highway Data 2002-2014 and New Zealand Transport Agency Traffic Data 2014 along with site surveys done by Massey University PhD students.

5.2.8 Demand Scenarios and Analaysis Procedure

Demand scenarios are designed to study the performance using PFLDIBC algorithm. The demand changes every 15 minutes in order to determine whether or not the PFLDIBC algorithm has the capability to discharge the increased in volume on particular movements while maintaining the operation of the entire diamond interchange system.

The PFLDIBC was evaluated by comparing it with the ADI and FLDI models. The ADI model is controlled by AIMSUN with its built in ALINEA algorithm. These values are

collected from the various detectors placed on the entire diamond interchange.The performances of the diamond interchange were measured based on the individual performance of the downstream, ramp, intersections and overall system. The performance of the model is based on the actual performance of the on ramp linked with the freeway (% change in Downstream Average Speed and % change in Downstream Average Delay) and the performance of the two intersections were based on the Delay Time. The System Total Travel Time is the overall performance of the ramp and intersections. The PFLDIBC's performance is measured by using the following Measures of Effectiveness (MOE) guidelines from (Fang, 2004). Travel delay is defined as the time difference between the desired and actual travel time of a vehicle. In this definition, desired travel time is determined (by the simulator) from both the highway speed limit and the driving characteristics of each individual vehicle. Freeway Average Delay is the time per vehicle while travelling on the freeway mainline (minutes/vehicle) within a specified period of analysis. The Freeway Average Speed is defined as the space mean speed for vehicles serviced by the freeway mainline (kilometres/hour) within a period of analysis. The System Average Delay is the total delay time accumulated by all vehicles (vehicle-kilometres). Lastly, the System Total travel time is the time accumulated by all vehicles (vehicle-hours).

5.2.9 Comparison for Low Traffic Demand

Tests under two scenarios are carried out with ten replications for each low traffic demand. Total Traffic Demand is ranging from 2925 to 3771 vehicles per hour with varied flow rate as shown in Table 5.4 and 5.5. The cyclists traffic demand ranging from 5 to over 18 cyclists per hour. These scenarios are considered as under-saturated traffic conditions and have different traffic demand distribution ratios. This traffic demand represents for weekend traffic. The MOEs are shown in Table 5.6 and 5.7 respectively.

Table 5.4 Traffic demand data for Scenario 1 (2925 vehicles per hour) – Low

Time (AM)		Constellation towards Upper Harbour			Upper Harbour towards Constellation			Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Cyclist	Car	HOV	Cyclist	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	5	379	19	0	278	14	1344	67	36	2
8:15	8:30	10	430	22	2	384	19	1582	79	96	5
8:30	8:45	17	481	24	3	490	25	1749	87	137	7
8:45	9:00	13	727	36	0	509	25	2216	111	254	13
Average Demand		11	504	25	1	415	21	1723	86	131	7
Total Demand			541			437		1809		138	

Table 5.5 Traffic demand data for Scenario 2 (3771 vehicles per hour) – Low

Time (AM)		Constellation towards Upper Harbour			Upper Harbour towards Constellation			Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Cyclist	Car	HOV	Cyclist	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	10	579	29	1	478	24	1544	77	236	12
8:15	8:30	15	630	32	3	584	29	1782	89	296	15
8:30	8:45	22	681	34	2	690	35	1949	97	337	17
8:45	9:00	18	927	46	0	709	35	2416	121	454	23
Average Demand		16	704	35	2	615	31	1923	96	331	17
Total Demand			756			648		2019		348	

Table 5.6 Measure of Effectiveness between models Scenario 1

	Downstream MOE				Ramp MOE			System MOE		
	TTT	**AD**	**AFR**	**AS**	**TTT**	**AD**	**AFR**	**TTT**	**AD**	**AFR**
	s/km	**sec/veh**	**veh/hr**	**km/hr**	**sec/km**	**sec/veh**	**veh/hr**	**sec/km**	**sec/veh**	**veh/hr**
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDIBC (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
	Percentage of change									
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

Table 5.7 Measure of Effectiveness between models Scenario 2

	Downstream MOE				Ramp MOE			System MOE		
	TTT	**AD**	**AFR**	**AS**	**TTT**	**AD**	**AFR**	**TTT**	**AD**	**AFR**
	s/km	**sec/veh**	**veh/hr**	**km/hr**	**sec/km**	**sec/veh**	**veh/hr**	**sec/km**	**sec/veh**	**veh/hr**
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDIBC (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
	Percentage of change									
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

5.2.10 Comparison for Medium Traffic Demand

Tests under two scenarios are carried out with ten replications for medium traffic demand. Total Traffic Demand is ranging from 4608 to 8301 vehicles per hour with varied flow rate as shown in Table 5.8 and 5.9 below. The cyclists traffic demand ranging from 1 to over 40 cyclists per hour. These scenarios are considered as over-saturated traffic conditions because the traffic demands are now exceeding the designed traffic capacity of the Upper Harbour Highway Interchange. This traffic demand represents for weekdaytraffic.

Table 5.8 Traffic demand data for Scenario 3 (4608 vehicles per hour) – Medium

Time (AM)		Constellation towards Upper Harbour			Upper Harbour towards Constellation			Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Cyclist	Car	HOV	Cyclist	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	15	779	39	2	678	34	1744	87	436	22
8:15	8:30	20	830	42	4	784	39	1982	99	496	25
8:30	8:45	27	881	44	3	890	45	2149	107	537	27
8:45	9:00	23	1127	56	1	909	45	2616	131	624	33
Average Demand		21	904	45	3	815	41	2123	106	523	27
Total Demand			971			859		2229		550	

Table 5.9 Traffic demand data for Scenario 4 (6301 vehicles per hour) – Medium

Time (AM)		Constellation towards Upper Harbour			Upper Harbour towards Constellation			Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Cyclist	Car	HOV	Cyclist	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	20	1179	59	3	1078	54	2144	107	836	42
8:15	8:30	25	1230	62	5	1184	59	2382	119	896	45
8:30	8:45	32	1281	64	2	1290	65	2549	127	937	47
8:45	9:00	28	1527	76	1	1309	65	3016	151	1054	53
Average Demand		26	1304	65	3	1215	61	2523	126	931	47
Total Demand			1396			1279		2649		978	

Table 5.10 Measure of Effectiveness between models Scenario 3

	Downstream MOE				Ramp MOE			System MOE		
	TTT s/km	AD sec/veh	AFR veh/hr	AS km/hr	TTT sec/km	AD sec/veh	AFR veh/hr	TTT sec/km	AD sec/veh	AFR veh/hr
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDIBC (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
	Percentage of change									
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

Table 5.11 Measure of Effectiveness between models Scenario 4

	Downstream MOE				Ramp MOE			System MOE		
	TTT s/km	AD sec/veh	AFR veh/hr	AS km/hr	TTT sec/km	AD sec/veh	AFR veh/hr	TTT sec/km	AD sec/veh	AFR veh/hr
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDIBC (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
					Percentage of change					
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

5.2.11 Comparison for Extreme Traffic Demand

Tests under two scenarios are carried out with ten replications for each demand. Total Traffic Demand is ranging from 7981 to 8831 vehicles per hour with varied flow rate. These total traffic demand exceed the designed capacity of the interchange.

Table 5.12 Traffic demand data for Scenario 5 (7981 vehicles per hour) – Extreme

Time (AM)		Constellation towards Upper Harbour			Upper Harbour towards Constellation			Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Cyclist	Car	HOV	Cyclist	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	25	1579	79	0	1478	74	2544	127	1236	62
8:15	8:30	30	1630	82	0	1584	79	2782	139	1296	65
8:30	8:45	37	1681	84	4	1690	85	2949	147	1337	67
8:45	9:00	33	1927	96	0	1709	73	3416	171	1454	73
Average Demand		31	1704	85	1	1615	78	2923	146	1331	67
Total Demand			1821			1694		3069		1398	

Table 5.13 Traffic demand data for Scenario 6 (8831 vehicles per hour) – Extreme

Time (AM)		Constellation towards Upper Harbour			Upper Harbour towards Constellation			Greville towards Tristram (South-bound)		Tristram towards Greville (North-bound)	
From	To	Cyclist	Car	HOV	Cyclist	Car	HOV	Car	HOV	Car	HOV
8:00	8:15	30	1779	89	3	1678	84	2744	137	1436	72
8:15	8:30	35	1830	92	5	1784	89	2982	149	1496	75
8:30	8:45	42	1881	94	2	1890	95	3149	157	1537	77
8:45	9:00	38	2127	106	0	1909	95	3616	181	1654	83
Average Demand		36	1904	95	3	1815	91	3123	156	1531	77
Total Demand			2036			1909		3279		1608	

Table 5.14 Measure of Effectiveness between models Scenario 5

	Downstream MOE				Ramp MOE			System MOE		
	TTT	**AD**	**AFR**	**AS**	**TTT**	**AD**	**AFR**	**TTT**	**AD**	**AFR**
	s/km	**sec/veh**	**veh/hr**	**km/hr**	**sec/km**	**sec/veh**	**veh/hr**	**sec/km**	**sec/veh**	**veh/hr**
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDIBC (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
	Percentage of change									
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

Table 5.15 Measure of Effectiveness between models Scenario 6

	Downstream MOE				Ramp MOE			System MOE		
	TTT	**AD**	**AFR**	**AS**	**TTT**	**AD**	**AFR**	**TTT**	**AD**	**AFR**
	s/km	**sec/veh**	**veh/hr**	**km/hr**	**sec/km**	**sec/veh**	**veh/hr**	**sec/km**	**sec/veh**	**veh/hr**
ADI (1)	14702	0.45	1865	86.99	9363	2.15	455	42.75	21.28	2454
FLDI (2)	13715	0.44	1745	88.23	8885	3.85	383	36.94	11.17	2388
PFLDIBC (3)	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
	Percentage of change									
(2) vs. (1)	-6.71	-2.22	-6.43	1.43	-5.11	79.07	-15.82	-13.59	-47.51	-2.69
(3) vs. (1)	-1.19	-6.67	-0.38	1.62	-0.26	43.26	-7.25	-16.12	-75.75	0.94
(3) vs. (2)	5.92	-4.55	6.48	0.19	5.12	-20.00	10.18	-2.92	-53.80	3.73

5.3 SUMMARY

According to Table 7, the PFLDIBC model gives smaller delays than the FLDI and ADI models when the traffic volumes are below 5000 veh/h. However, if traffic volumes are more than 5000 veh/h, the PFLDIBC gives higher delays than the ADI but performs better than the FLDI. In Upper Harbor diamond interchange the total traffic demand in any given day does not exceed 3800 veh/h. The PFLDIBC was designed to give higher priority to the traffic on the highways. So when traffic demand is greater than the road capacity, the PFLDIBC causes the internal congestion at the two inner intersections. This leads to the increase in average delay of the vehicles at the intersections and on-ramp which increases the overall average delay time of the whole system. Moreover, because the PFLDBC gives priority to the cyclists therefore it affects the average total delay time of the entire system.

Looking at the big picture, increasing the cyclists flow rate, their safeness (via priority control) and better infrastructure design will encourage more cyclists hit the road. This will decrease the amount of vehicles on the road as people now prefer to cycle to the bus station and get on the bus to work.

CHAPTER 6
SUMMARY AND CONCLUSIONS

6.1 SUMMARY OF THE THESIS RESEARCH

In this research, a probabilistic fuzzy logic control is developed for the control of the traffic signal system at a typical diamond interchange incorporating ramp metering. The signal phase and green time extension are decided in response to the traffic conditions in real time. The traffic signal in PFLDI operates under fuzzy logic and the stochastic uncertainties in the traffic conditions, such as the number of vehicles arriving and queuing at an intersection, are described by probabilistic fuzzy sets. To validate the developed PFLC, the traffic system is modeled using Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network version 6 (AIMSUN 6). The stochastic traffic conditions (vehicles arrival and queue) are simulated and the performance of the traffic system under PFLDI is evaluated. The PFLDI is also compared with actuated and conventional fuzzy logic controllers for intersection signals in terms of the average delay of vehicles and the average flow rate. The results show the intersection using PFLC for signal results in an improved performance.

6.2 MAJOR CONTRIBUTIONS AND CONCLUSIONS

Fuzzy Logic Diamond Interchange (PFLDI), Fuzzy Logic Diamond Interchange Metering (FLDI), Actuated Diamond Interchange Control (ADI) were built and tested in AIMSUN 6 environment with six different traffic scenarios ranging from moderate to heavy congestion. From the results, it was found that PFLDI outperforms the ADI and FLDI models in terms of system total travel time, downstream average delay and average speed even when the traffic demand exceeds the road capacity (including arterial roads, ramp and motorway). The ADI model becomes less effective once the total traffic demand is much higher than road capacity. The PFLDI helps avoid the formation of congestion and in extreme traffic conditions is also able to prevent the congestion from getting worse.

A novel bicycle signal control concept has been introduced at a diamond interchange using probabilistic fuzzy logic algorithm. The model is tested and compared with numerous controls. The model may not be effective at all-time comparing to ADI control due to the objectives of the algorithm when set up. The PFLDIBC gives higher priority to cyclists on the road than other vehicles. It is designed to improve the safety and flow rate of the cyclists. In the long term, more people will choose to cycle to work or bus station than driving, this not only saves time but also proved to be more economical than other traditional methods. Once this has been achieved, there will be a significant drop in private vehicles entering the diamond interchange which will then improve the speed, traffic flow and reduce the pollution.

6.3 FUTURE RESEARCH

Upon completion of this extensive study, the following has been identified and recommended in future research. They are:

i) The methodology developed in this thesis is designed for a typical diamond interchange with two signalised intersections.

ii) A new Fuzzy Logic Diamond Interchange Control algorithm may be developed by considering the cyclists, public transportation in future studies. The current membership functions are not chosen by using any artificial intelligence agent. It can be trained with genetic algorithm or artificial neural net-works.

iii) An expansion of Fuzzy Logic Diamond Interchange control for the entire traffic network can be planned**.**

REFERENCE

Auckland Regional Transport Authority (ARTA) (2007) Sustainable transport plan 2006–2016. Auckland: Auckland Regional Transport Authority.

Auckland Region Manual Cycle Monitor (2014). Research Report Prepared for Auckland Transport, Gravitas.

Austroads (1994) Guides to engineering practice – part 7 traffic signals. Austroads Publication no. APG11.7/3, 186.

Austroads Guides to engineering practice – part 14 . Austroads Publication no. APG11.7/3, 186.

Austroads (2008) Traffic flow models allowing for pedestrians and cyclists. Austroads Publication no. APR193-01

Austroads (2009) Guide to traffic management. Australia: Austroads.

Bogenberger, K., and Keller, H. (2000). "An Evolutionary Fuzzy System for Coordinated and Traffic Responsive Ramp-metering." *Annual Hawaiian International Conference on System Sciences,* USA, 1-3.

Brown, T., Clark, I., Evans, P. and Wee, L. (2005). "New Zealand's easy merge ramp signal (ramp-metering) trial." *IPENZ*, 8, 1-14.

Charles, P. (2001). "Time for a Change." *Traffic Technology International*, 68-71.

Chou, C., and Teng, J. (2002) "A fuzzy logic controller for traffic junction signals." *Information Science*, vol 143(1-4), 73-97.

Christchurch Cycle Design Guidelines (CCDG) (2013), Published by Christchurch City Council, New Zealand.

Colubi, A., Fernandez-Garcia, C., and Gil, M. A. (2003). "Simulation of random fuzzy variables: an empirical approach to statistical/probabilistic studies with fuzzy experimental data." *IEEE Trans. Fuzzy System*, 10(3), 384-390.

Daff, M., Cramphorn, B., Wilson, C., and Neylan, J. (1991). "Pedestrian behaviour near signalised crossings." *Proceedings 16th ARRB Conference*, 49–65.

Fang, F. C. (2004). "Optimal Adaptive Signal Control for Diamond Interchange using Dynamic Programming." Ph. D Thesis, College of Engineering, Pennsylvania State University, USA, 4-32.

Fang, F. C., and Elefteriadou, L. (2006). "Development of an Optimization Methodology for Adaptive Traffic Signal Control at Diamond interchange." *ASCE Journal of Transportation Engineering*, vol. 132, no.8, 629-637.

Favilla, J., Machion, A., Gomide, F. (1993). "Fuzzy traffic control: adaptive strategies." *Second IEEE International Conference on Fuzzy Systems II*, 506-511.

Federal Highway Administration (2009) Manual on uniform traffic control devices, 816.

Forester, J. (1994). "Bicycle Transportation: A Handbook for Cycling Transportation Engineers." 2nd ed. The MIT Press, Cambridge, Mass.

Garrett, D. L. (2003). "Coupled analysis of floating production systems." *Proc., Int. Symp. on Deep Mooring Systems*, ASCE, Reston, VA, 152-167

Gartner, N. H. (1983). "OPAC: A Demand-Responsive Strategy for Traffic Signal Control." *Transportation Research Record 906*, Transportation Research Board, Washington, DC,75-81.

Gartner, N. H., and Pooran, F. J. (2002). "Implementation and Field Testing of the OPAC Adaptive Control Strategy in RT-TRACS." *the 81st Annual TRB Meeting*, Washington DC, USA.

German HCM (1997) German highway capacity manual, version 2000. Research report. Bochum: Institute for Traffic Engineering, Ruhr-University.

Hass, P. C. (2001). “Assessing Developments Using AIMSUN.” *IPENZ Transportation Conference*, Auckland, New Zealand.

Wei, H., Lu, F., Hou, G., and Mogharabi, A. (2003). “Non-Motorized Disturbances and Control Measures at Signalized Intersections in China.” *TRB 2003 Annual Meeting*, Washington DC, 2-10.

Highway Capacity Manual (2000). Transportation Research Board, National Research Council, Washington DC, USA, 12-24.

Highway Capacity Manual (2000). Chapter 11 Pedestrian and Bicycle Concepts. Transportation Research Board, Washington DC, USA.

Hoyer, R., and Jumar, U. (1994). "An Advanced Fuzzy Controller for Traffic Lights." *Artificial Intelligence in Real Time Control Symposium*, Valencia, Spain, 3-6.

Hughes, T. J. (2000). “AIMSUN2 Simulation of a Congested Auckland Freeway.” Master thesis at The University of Auckland, New Zealand, 10-25.

Hunt, P. B., Robertson, D. I., and Bretherton, R. D., Royle, M. C. (1982). "The SCOOT on-line-traffic-signal optimization technique." *Traffic Engineering and Control*, 23(4), 190-192.

Jacobsen, P. L. (2003). “Safety in numbers: more walkers and bicyclists, safer walking and bicycling.” *Injury Prevention*, 9, 205-209.

Jamshidi, M., Kelsey, R., Bisset, K. (1993) “A simulation environment of fuzzy control of traffic systems”, *12 IFAC-World Congress*, 553-556.

Juang, J. G. (2000). “Neural fuzzy system approaches for robotic gait synthesis.” *IEEE Trans. Syst., Man., Cybern.,* 30(4), 594-601.

Kamarajugadda, A., and Park, B. (2003). “Stochastic Traffic Signal Timing Optimization.” Research Re-port, UVACTS-15-0-44, Centre for Transport Studies, University of Virginia, Virginia, 2-40.

Liang, P., and Song, F. (1996). “What does a probabilistic interpretation of fuzzy sets mean?” *IEEE Trans. Fuzzy System*, 4(2), 200-205.

Liu, Z., and Li, H. X. (2005). “A Probabilistic Fuzzy Logic Control System for Modeling and Control.” *IEEE Trans. Fuzzy System*, 13(6), 848-858.

Luk, J. Y. K. (1984). “Two Traffic Responsive Area Traffic Control Methods: SCATS and SCOOT.” *Traffic Engineering and Control*, 30, 14-20.

Lin, F. B, and Vijayayumar, S. (1988). “Adaptive signal control at isolated intersections.” *ASCE Journal of Transportation Engineering*, 114(5), 555-573.

Liu, Z., and Li, H. X. (2005). “A Probabilistic Fuzzy Logic Control System for Modeling and Control.” *IEEE Trans. Fuzzy System*, 13(6), 848-858.

Meghdadi, A.H., and Akbarzadeh-T, M.-R. (2001). “Probabilistic fuzzy logic and probabilistic fuzzy systems.” *Proc. 10th IEEE Int. Conf*, Melbourne, Australia, 1127-1130.

Messer, C. J., Fambro, D. B., and Richards, S. H. (1977). “Optimization of Pretimed Signalized Diamond interchange”, *Transportation Research Record* 644, Transportation Research Board, Washington, D.C., pp.78-84.

Miller, A. J. (1963). “A Computer Control System for Traffic Networks.” Birmingham, England.

Miller, R., Ramey, M, (1975). “Width Requirements for Bikeways: A Level of Service Approach.” *Report 75-4*, Department of Civil Engineering, University of California, Davis.

Mirchandani, P., and Head, L. (2001). “A Real-time Traffic Signal Control System: Architecture, Algorithms and Analysis.” *Transportation Research*, C(9), 414-432.

Muhurdarevic, Z., Condie, H., Ben-Shabat, E., Ernhofer, O., Hadj-SSalem, H., and Papamichail, I. (2006). “Ramp-metering in Europe – state-of-art and beyond.” *5th International Conference on Traffic and Transportation Studies* (ICTTS), Xi’an, China, 2006.

Nakatsuyama, N., Nagahashi, H., Nishizuka, N. (1984) “Fuzzy logic phase controller for traffic functions in the one-way arterial road.”, *in: Proc. IFAC 9th Triennial World Congress*, 2865-1870.

New Zealand supplement to Austroads guide to traffic engineering practice: part 14 bicycles (GTEP14) (2008). New Zealand Transportation Agency, ISBN:0-478-10559-2.

New Zealand State Highway Traffic Data Booklet 2002 – 2006. (2007), Transit New Zealand

Noland, R. B., Ishaque, M. M. (2007) “Trade-offs between vehicular and pedestrian traffic using micro-simulation methods.”, *Transport Policy*, 14(20, 124-138.

Pappis, C. P., and Mamdani, E. H. (1977). “A Fuzzy Logic Controller for a Traffic Junction.” *IEEE Transactions System, Man and Cybernetics*, SMC-7 (10), 707-717.

Pham, V. C., Alam, F., Potgieter, J., Fang, F., and Xu, W. L. (2013). “Integrated fuzzy signal and ramp-metering at a diamond interchange” *J. Adv. Transp.*, 47, 413-434.

Pham, V. C., Alam, F., Potgieter, J., Fang, F., and Xu, W. L. (2015). “Probabilistic Fuzzy Logic Signal and Ramp-Metering at a Diamond Interchange” *J. Trans. Eng.*, 141(3)

Sen, U., and Head, K. L. (1997). “Controlled Optimization of Phases at an Intersection.” *Transportation Science*, 31(1), 5-17.

Sisson, B. (2006). “Baseplus FUSE: The Link Between SCATS and S-Paramics” *Proceedings New Zealand Microsimulation Conference*, 65-89.

Stahl, D. C., Wolfe, R. W., and Begel, M. (2004). “*Improved analysis of timber rivet connections”* J. Struct. Eng., 130(8), 1272-1279.

Taylor, D. (1993). “Analysis of Traffic Signal Clearance Interval Requirements for Bicycle-Automobile Mixed Traffic.” *In Transportation Research Record 1405,* TRB, National Research Council, Washington, D.C., 13–20.

Tian, Z. Z. (2004). “Development and Evaluation of Operational Strategies for Providing an Integrated Diamond Interchange – Ramp Metering Control System”, *Ph. D Thesis*, Department of Civil & Environmental Engineering , Texas A&M University, 26-79.

Tian, Z. Z. (2007). “Modeling and implementation of an integrated ramp-metering-diamond interchange control system.” *Journal of Transportation Systems Engineering and Information Technology*, 7(1), 61-72.

Tian, Z. Z., Balke, K., Engelbrecht, B., and Rilett, L. (2002). “Integrated control strategies for surface street and freeway systems.” *Transportation Research Record 1811*, 92-99.

Trabia, M. B., Kaseko, M. S., and Ande, M. (1999). “A two-stage fuzzy logic controller for traffic signals.” *Transportation Research*, C(7), 353-36.

Turner, S. A. (2000). “Accident Prediction Models.” *Transfund New Zealand Research Report No. 192*. Transfund New Zealand.

Turner, S. A, and Roozenburg, A. P. (2004). “Accident Prediction Models at Signalised Intersections: Right-Turn-Against Accidents, Influence of Non-Flow Variables.” *Unpublished Roads Safety Trust Research Report*.

Turner, S. A., Roozenburg, A. P., & Francis, T. (2005). “Predicting Accident Rates for Cyclists and Pedestrians (Draft) Final Report.” *Land Transport NZ Research Report*.

Turner, S. A., Binder, S., and Roozenburg, A. (2009). “Cycle Safety: Reducing the Crash Risk October 2009.” *NZ Transport Agency research report 389,* 63-79.

Turner, S. A., Singh, R., and Nates, G. D. (2012). “Crash prediction models for signalised intersections: signal phasing and geometry June 2012.” *NZ Transport Agency research report 483*. 16-97.

Wachtel, A., Forester, J., and Pelz, D. (1995). “Signal Clearance Timing for Bicyclist.” *ITE Journal*, Vol. 65, No. 3, 1995, pp. 38–45.

Zadeh, L. A. (1981). “Possibility theory and soft data analysis.” *Mathematical frontiers of the social and policy sciences*, L. Cobb and R. M. Thrall, eds., Westview, Boulder, CO, 69-129.

APPENDIX A

PROBABILISTIC FUZZY LOGIC DIAMOND INTERCHANGES C++ CODE

1. AAPI.cxx (Main file)

```
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include "stdio.h"
#include "FLCB.h"
#include "FLCD.h"
#include "FLCF.h"

char astring[128];

int AAPILoad()
{

	return 0;
}

int AAPIInit()
{

	ANGConnEnableVehiclesInBatch(true);
	return 0;
}

int AAPIManage(double time, double timeSta, double timTrans, double acicle)
{
	static int det1,det2,det3,det4,det5,det6,det7,det8,det9,det10,det11,det12,det13,det14;
	static int det15,det16_1,det16_2,det16_3,det17,det18_1,det18_2,det18_3,det18_4,det19;
	static int det20_1,det20_2,det20_3,det21,det23,det24,det25;

	double light1,light2,light3,light4,light5,light6,light7,light8,light9;

	int Aphase_det=0;
	int Qphase_det=0;

	static int totalAD=0; // Total Arrival in Phase D
	static int totalQD=0; // Total Queue in Phase     D
	static int totalAB=0; // Total Arrival in Phase B
	static int totalQB=0; // Total Queue in Phase     B
	static int totalAF=0; // Total Arrival in Phase F
	static int totalQF=0; // Total Queue in Phase     F

// Read the Instant Counter measure of a detector from AIMSUN (31 detectors)
	det3    = AKIDetGetCounterAggregatedbyId (843, 0);       //light 1
	det4    = AKIDetGetCounterAggregatedbyId (1571,0);       //light 1
	det5    = AKIDetGetCounterAggregatedbyId (1572,0);       //light 1
	det18_1 = AKIDetGetCounterAggregatedbyId (1576,0);    //light 1 advanced loop
	det18_2 = AKIDetGetCounterAggregatedbyId (1575,0);    //light 1 advanced loop
	det18_3 = AKIDetGetCounterAggregatedbyId (1574,0);    //light 1 advanced loop
	det18_4 = AKIDetGetCounterAggregatedbyId (1573,0);    //light 1 advanced loop
	det1    = AKIDetGetCounterAggregatedbyId (1585,0);       //light 2
	det16_3 = AKIDetGetCounterAggregatedbyId (1590,0);    //light 2
	det16_1 = AKIDetGetCounterAggregatedbyId (741,0);     //light 2 advanced loop
	det16_2 = AKIDetGetCounterAggregatedbyId (1589,0);    //light 2 advanced loop
	det2    = AKIDetGetCounterAggregatedbyId (845,0);        //light 3
	det17   = AKIDetGetCounterAggregatedbyId (1540,0);    //light 3 advanced loop
	det12   = AKIDetGetCounterAggregatedbyId (1582,0);       //light 4
	det13   = AKIDetGetCounterAggregatedbyId (745,0);        //light 4
	det23   = AKIDetGetCounterAggregatedbyId (743,0);     //light 4 advanced loop
	det20_3 = AKIDetGetCounterAggregatedbyId (1593,0);    //light 5
	det8    = AKIDetGetCounterAggregatedbyId (1579,0);       //light 5
	det20_1 = AKIDetGetCounterAggregatedbyId  (740,0);    //light 5 advanced loop
	det20_2 = AKIDetGetCounterAggregatedbyId (1592,0);    //light 5 advanced loop
	det9    = AKIDetGetCounterAggregatedbyId (841,0);        //light 6
	det21   = AKIDetGetCounterAggregatedbyId (1541,0);    //light 6 advanced loop
	det10   = AKIDetGetCounterAggregatedbyId (1580,0);       //light 7
```

```
	det11   = AKIDetGetCounterAggregatedbyId (1581,0);	//light 7
	det25   = AKIDetGetCounterAggregatedbyId (1587,0);	//light 7 advanced loop
	det6    = AKIDetGetCounterAggregatedbyId (1578,0);	//light 8
	det7    = AKIDetGetCounterAggregatedbyId (737,0);	//light 8
	det19   = AKIDetGetCounterAggregatedbyId (744,0);	//light 8 advanced loop
	det14   = AKIDetGetCounterAggregatedbyId (844,0);	//light 9
	det15   = AKIDetGetCounterAggregatedbyId (1583,0);	//light 9
	det24   = AKIDetGetCounterAggregatedbyId (742,0);	//light 9 advanced loop


//Read the current running phase from AIMSUN
	Aphase_det = ECIGetCurrentPhase(1148);
	Qphase_det = ECIGetCurrentPhase(1143);


// Read a state of traffic light from AIMSUN (9 traffic lights)
	light1 = ECIGetStateSem(598,3,0,timeSta);//Upper Harbour Drive Traffic Light 1
	light2 = ECIGetStateSem(698,2,0,timeSta);//Constellation Drive Traffic Light 2
	light3 = ECIGetStateSem(698,2,0,timeSta);//Constellation Drive Traffic Light 3
	light4 = ECIGetStateSem(659,2,0,timeSta);//Constellation Drive Traffic Light 4
	light5 = ECIGetStateSem(703,3,0,timeSta);//Upper Harbour Drive Traffic Light 5
	light6 = ECIGetStateSem(703,3,0,timeSta);//Upper Harbour Drive Traffic Light 6
	light7 = ECIGetStateSem(660,2,0,timeSta);//South-bound on-ramp  Traffic Light 7
	light8 = ECIGetStateSem(683,1,0,timeSta);//North-bound off ramp Traffic Light 8
	light9 = ECIGetStateSem(638,2,0,timeSta);//South-bound off ramp Traffic Light 9

//-------------------------------------------------------------------------------
//PHASE D CAR COUNTS, FUZZY GREEN TIME EXTENSION CALCULATION, CHANGING PHASE DURATION
//-------------------------------------------------------------------------------
	static int Dgetphasetwo=0;
	static int c=0;
	static int d=0;
	static int e=0;
	static int f=0;
	static int Dchange=0;
	static int Dcount=0;
	static int Dphase=0;
	int 			Dextensionrate=0;


//Movement 1, 2, 3, 5 – PHASE D TOTAL ARRIVAL CALCULATION - GREEN
	if (light1==1 && light5==1 && light2==1 && light3==1)
	{
		if (det1>=0 && det16_3>=0 && det2>=0)
		{
			if (c==8)
			{
				totalAD=det1+det2+det16_3;
				c=0;
			}
		c=c+1;
		}
	}

//Movement 4, 6, 7, 8, 9 – PHASE D TOTAL QUEUE CALCULATION - RED
	if (light6 == 0 || (light4 ==0 && light7==0 && light8==0 && light9==0))
	{
		if (det25>=0 && det23>=0 && det24>=0 && det21>=0 && det19>=0)
		{
			if (d==8) {
				totalQD=det25+det23+det24+det21+det19;
				d=0;
			}
			d=d+1;
		}
	}
//Check the phase status every 1 simulation step
	if (Dgetphasetwo == Aphase_det && Aphase_det>0) {Dchange=0;}
	else {	Dchange=1;}

	if (Aphase_det>0){Dgetphasetwo = Aphase_det;}

//Check if the current phase is Phase D (Phase 1 in AIMSUN)

	if (Dchange==1)
	{
```

```
                        if(Aphase_det==1)
                        {
                                Dphase=1;
                        }
                }
//Arrival&Queue car counts for the first 6 seconds (8 in simulation steps) of PHASE D
                if (Dchange==1&&Dphase==1) {Dcount=1;}
                if (Dcount==1)
                {



if (f==8)
{
Dextensionrate= DflcExtensionRate (float(totalAD),float(totalQD));//FLC extesnion rate
sprintf_s(astring, "PHASE D started...!!");
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Arrival Vehicles in Phase D = %d\n",totalAD);
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Queue Vehicles in Phase D = %d\n",totalQD);
AKIPrintString(astring);
sprintf_s(astring,"Fuzzy Green Time Extension for Phase D=%dseconds\n, Dextensionrate);
AKIPrintString(astring);
ECIChangeTimingPhase(1148,1,Dextensionrate,timeSta);//Change the duration of Phase D
ECIChangeTimingPhase(1143,1,Dextensionrate,timeSta);//Change the duration of Phase D
totalAD=0;
totalQD=0;
f=0;
Dcount=0;
Dphase=0;
                }
f=f+1;
}

//-------------------------------------------------------------------------------
//PHASE B CAR COUNTS, PROBABILISTIC FUZZY GREEN TIME EXTENSION CALCULATION, CHANGING PHASE DURATION
//-------------------------------------------------------------------------------

                static int ABgetphasetwo=0;
                static int a=0;
                static int b=0;
                static int r=0;
                static int v=0;
                static int ABchange=0;
                static int ABcount=0;
                static int ABphase=0;
                int     Bextensionrate=0;

//Movement 5, 6, 7, 9 – TOTAL ARRIVAL CALCULATION FOR PHASE B
                if (light5==1 && light6==1 && light7==1 && light9==1)
                {
                        if (det21>=0 && det20_1>=0 && det20_2>=0 && det25>=0)
                        {
                                if (r==8)
                                {
                                        totalAB=det25+det20_1+det20_2+det21;
                                        r=0;
                                }
                        r=r+1;
                        }
                }

//Movement 1, 2, 3, 4, 8 – TOTAL QUEUE CALCULATION FOR PHASE B
                if (light1==0 && light2==0 && light3==0 && light4==0 && light8==0)
                {
                        if (det18_2>=0&&det18_3>=0&&det18_4>=0&&det23>=0&&det25>=0)
                        {
                                if (v==8) {
                                        totalQB=det18_2+det18_3+det18_4+det23+det25;v=0;
                                }
                                v=v+1;
                        }
                }
```

```
//Check the phase status every 1 simulation step
		if (ABgetphasetwo==Aphase_det&&Aphase_det>0) {ABchange=0;}
		else {ABchange=1;}
		if (Aphase_det>0) {ABgetphasetwo=Aphase_det; a=0; }


//Check if the current phase is Phase B (Phase 2 in AIMSUN)

		if (ABchange==1) {
			if (Aphase_det==2) {
				ABphase=1;

			}
		}

//Arrival&Queue car counts for the first 6 seconds (8 in simulation steps) of PHASE B
		if (ABchange==1&&ABphase==1) {ABcount=1;}
		if (ABcount==1){
			if (b==8) {
Bextensionrate=BflcExtensionRate (float(totalAB), float(totalQB));//FLC extension rate
sprintf_s(astring, "PHASE B started...!!");
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Arrival Vehicles in Phase B = %d\n",totalAB);
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Queue Vehicles in Phase B = %d\n",totalQB);
AKIPrintString(astring);
sprintf_s(astring,"Fuzzy Green Time Extension for Phase B=%dseconds\n, Bextensionrate);
AKIPrintString(astring);
ECIChangeTimingPhase(1148,2,Bextensionrate,timeSta);//Change the duration of Phase B
ECIChangeTimingPhase(1143,2,Bextensionrate,timeSta);//Change the duration of Phase B
totalAB=0;
totalQB=0;
b=0;
ABcount=0;
ABphase=0;

		}
b=b+1;}

//-------------------------------------------------------------------------------
// PHASE F CAR COUNTS, PROBABILISTIC FUZZY GREEN TIME EXTENSION CALCULATION, CHANGING PHASE DURATION
//-------------------------------------------------------------------------------

		static int Fgetphasetwo=0;
		static int m=0;
		static int n=0;
		static int o=0;
		static int p=0;
		static int Fchange=0;
		static int Fcount=0;
		static int Fphase=0;
		int	Fextensionrate=0;

//Movement 2, 4, 7, 8 – TOTAL ARRIVAL CALCULATION FOR PHASE F - GREEN
		if (light2==1 || (light4==1 && light7==1 && light8==1))
		{
			if (det19>=0 && det13>=0 && det12>=0 && det1>=0 && det16_3>=0 &&det10>=0 && det11>=0)
			{
				if (m==8)
				{
					totalAF=det19+det13+det12+det1+det16_3+det10+det11;
					m=0;
				}
			m=m+1;
			}
		}

//Movement 1, 3, 5, 6, 9 – TOTAL QUEUE CALCULATION FOR PHASE F - RED
		if (light1 == 0 && light3 ==0 && light5==0 && light6==0 && light9==0)
		{
			if (det18_2>=0&&det18_3>=0&&det18_4>=0&&det17>=0&&det20_1>=0&&det20_2>=0&&det24>=0)
			{
				if (n==8) {
					totalQF=det18_2+det18_3+det18_4+det17+det20_1+det20_2+det24;
					n=0;
				}
```

```
                                        n=n+1;
                        }
                }

//Check the phase status every 1 simulation step
if (Fgetphasetwo == Aphase_det && Qphase_det>0) {Fchange=0;}
else {          Fchange=1; }
if (Aphase_det>0){Fgetphasetwo = Aphase_det; o=0;}

//Check if the current phase is Phase F (Phase 3 in AIMSUN)

                if (Fchange==1)
                {
                        if(Aphase_det==3)
                        {
                                Fphase=1;
                        }
                }
//Arrival&Queue car counts for the first 6 seconds (8 in simulation steps) of PHASE F

                if (Fchange==1&&Fphase==1) {Fcount=1;}
                if (Fcount==1)
                {
                        if (p==16)
                        {
Fextensionrate=FflcExtensionRate (float(totalAF), float(totalQF));//FLC extension rate
sprintf_s(astring, "PHASE F started...!!");
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Arrival Vehicles in Phase F = %d\n",totalAF);
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Queue Vehicles in Phase F = %d\n",totalQF);
AKIPrintString(astring);
sprintf_s(astring,"Fuzzy Green Time Extension for Phase F=%dseconds\n, Fextensionrate);
AKIPrintString(astring);
ECIChangeTimingPhase(1148,3,Fextensionrate,timeSta);//Change the duration of Phase F
ECIChangeTimingPhase(1143,3,Fextensionrate,timeSta);//Change the duration of Phase F
totalAF=0;
totalQF=0;
p=0;
Fcount=0;
Fphase=0;
                        }
                        p=p+1;
                }

return 0;}
```

2. PFLDI.cxx (Fuzzy Logic D Phase Timing)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "FLCD.h"


//input1 definition - Phase D Arrival Membership functions

float ProbDarr_veh_few (float ProbDarr_veh_few) /*initial value 0 and 19*/
{                       if (ProbDarr_veh_few<0) {
                                return (0);}
                        if (ProbDarr_veh_few>19) {
                                return (0);}
                        if (ProbDarr_veh_few>=-19 && ProbDarr_veh_few<=0) {
                                return (ProbDarr_veh_few/19 +1);}
                        if (ProbDarr_veh_few>0 && ProbDarr_veh_few <=19) {
                                return ((-1)*ProbDarr_veh_few/19 + 1);}
                return (-1);
}


float ProbDarr_veh_small(float ProbDarr_veh_small) /*initial value 0 and 38*/
{                       if (ProbDarr_veh_small<0) {
                                return (0);}
                        if (ProbDarr_veh_small>38) {
```

```
                        return (0);}
                if (ProbDarr_veh_small>=0 && ProbDarr_veh_small <=19) {
                        return (ProbDarr_veh_small/19);}
                if (ProbDarr_veh_small>19 &&  ProbDarr_veh_small <=38) {
                        return ((-1)*ProbDarr_veh_small/19 + 2);}
        return (-1);
}


float ProbDarr_veh_medium(float ProbDarr_veh_medium) /*initial value 19 and 57*/
{               if (ProbDarr_veh_medium<19) {
                        return (0);}
                if (ProbDarr_veh_medium>57) {
                        return (0);}
                if (ProbDarr_veh_medium>=19 && ProbDarr_veh_medium<=38) {
                        return (ProbDarr_veh_medium/19 - 1);}
                if (ProbDarr_veh_medium>38 &&  ProbDarr_veh_medium<=57) {
                        return ((-1)*ProbDarr_veh_medium/19 + 3);}
        return (-1);
}


float ProbDarr_veh_many(float ProbDarr_veh_many) /*initial value 38 and 57*/
{               if (ProbDarr_veh_many<38) {
                        return (0);}
                if (ProbDarr_veh_many>76) {
                        return (0);}
                if (ProbDarr_veh_many>=38 && ProbDarr_veh_many<=57) {
                        return (ProbDarr_veh_many/19 - 2);}
                if (ProbDarr_veh_many>57 &&  ProbDarr_veh_many<=76) {
                        return ((-1)*ProbDarr_veh_many/19 + 4);}
        return (-1);
}


// Input 2 definition - Phase D Queue Membership functions

float ProbDq_veh_few (float ProbDq_veh_few) /*initial value 0 and 19*/
{               if (ProbDq_veh_few<0) {
                        return (0);}
                if (ProbDq_veh_few>19) {
                        return (0);}
                if (ProbDq_veh_few>=-19 && ProbDq_veh_few<=0) {
                        return (ProbDq_veh_few/19 +1);}
                if (ProbDq_veh_few>0 && ProbDq_veh_few <=19) {
                        return ((-1)*ProbDq_veh_few/19 + 1);}
        return (-1);
}


float ProbDq_veh_small(float ProbDq_veh_small) /*initial value 0 and 38*/
{               if (ProbDq_veh_small<0) {
                        return (0);}
                if (ProbDq_veh_small>38) {
                        return (0);}
                if (ProbDq_veh_small>=0 && ProbDq_veh_small <=19) {
                        return (ProbDq_veh_small/19);}
                if (ProbDq_veh_small>19 &&  ProbDq_veh_small <=38) {
                        return ((-1)*ProbDq_veh_small/19 + 2);}
        return (-1);
}


float ProbDq_veh_medium(float ProbDq_veh_medium) /*initial value 19 and 57*/
{               if (ProbDq_veh_medium < 19) {
                        return (0);}
                if (ProbDq_veh_medium>57) {
                        return (0);}
                if (ProbDq_veh_medium>=19 && ProbDq_veh_medium<=38) {
                        return (ProbDq_veh_medium/19 - 1);}
                if (ProbDq_veh_medium>38 &&  ProbDq_veh_medium<=57) {
                        return ((-1)*ProbDq_veh_medium/19 + 3);}
        return (-1);
}
```

```
float ProbDq_veh_many(float ProbDq_veh_many) /*initial value 38 and 57*/
{                               if (ProbDq_veh_many<38) {
                                        return (0);}
                                if (ProbDq_veh_many>76) {
                                        return (0);}
                                if (ProbDq_veh_many>=38 && ProbDq_veh_many<=57) {
                                        return (ProbDq_veh_many/19 - 2);}
                                if (ProbDq_veh_many>57 &&  ProbDq_veh_many<=76) {
                                        return ((-1)*ProbDq_veh_many/19 + 4);}
                return (-1);
}

 float DflcExtensionRate (float ProbDarr_veh, float ProbDq_veh)

{
  // Evaluate each rule
                float rule[16];
                rule[0]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_few(ProbDq_veh));
                rule[1]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_small(ProbDq_veh));
                rule[2]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_medium(ProbDq_veh));
                rule[3]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_many(ProbDq_veh));
                rule[4]=MIN(ProbDarr_veh_small(ProbDarr_veh), ProbDq_veh_few(ProbDq_veh));
                rule[5]=MIN(ProbDarr_veh_small(ProbDarr_veh), ProbDq_veh_small(ProbDq_veh));
                rule[6]=MIN(ProbDarr_veh_small(ProbDarr_veh), ProbDq_veh_medium(ProbDq_veh));
                rule[7]=MIN(ProbDarr_veh_small(ProbDarr_veh),           ProbDq_veh_many(ProbDq_veh));
                rule[8]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_few(ProbDq_veh));
                rule[9]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_small(ProbDq_veh));
                rule[10]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_medium(ProbDq_veh));
                rule[11]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_many(ProbDq_veh));
                rule[12]=MIN(ProbDarr_veh_many(ProbDarr_veh), ProbDq_veh_few(ProbDq_veh));
                rule[13]=MIN(ProbDarr_veh_many(ProbDarr_veh), ProbDq_veh_small(ProbDq_veh));
                rule[14]=MIN(ProbDarr_veh_many(ProbDarr_veh), ProbDq_veh_medium(ProbDq_veh));
                rule[15]=MIN(ProbDarr_veh_few(ProbDarr_veh), ProbDq_veh_many(ProbDq_veh));

 // The weighted sum of each class outccome
 // f_ext_zero is 0, f_ext_short is 1, f_ext_medium is 2, f_ext_long is 3
    float f_prob_ext_class[4];
                f_prob_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
    f_prob_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
    f_prob_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
    f_prob_ext_class[3]=rule[12];

 //defuzzification (discreted centroids method)
                float centroid=0;
                float area=0;
                float num=0;
                float den=0;
                float f_ext;

                for(int i=0;i<4;i++)
                {
                                if(i==0)
                                { area=8;
                                  centroid=5.33f;}
                                else if(i==1)
                                { area=16;
                                  centroid=16;}
                                else if(i==2)
                                { area=16;
                                                centroid=32;}
                                else
                                { area=8;
                                  centroid=42.67f;}

                                  num+=f_prob_ext_class[i]*area*centroid;
                                  den+=f_prob_ext_class[i]*area;
                               }

 // calculate extension rate and rescale to LL and HH range
                f_ext=num/den;
 // std::cout << "extension_rate\t"<< f_ext << std::endl;
               return(f_ext);

}
```

3. PFLCB.cxx (Probabilistic Fuzzy Logic B Phase Timing)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "FLC.h"


//input1 definition - Phase D Arrival Membership functions

float ProbBarr_veh_few (float ProbBarr_veh_few) /*initial value 0 and 19*/
{                       if (ProbBarr_veh_few<0) {
                                return (0);}
                        if (ProbBarr_veh_few>19) {
                                return (0);}
                        if (ProbBarr_veh_few>=-19 && ProbBarr_veh_few<=0) {
                                return (ProbBarr_veh_few/19 +1);}
                        if (ProbBarr_veh_few>0 && ProbBarr_veh_few <=19) {
                                return ((-1)*ProbBarr_veh_few/19 + 1);}
                return (-1);
}


float ProbBarr_veh_small(float ProbBarr_veh_small) /*initial value 0 and 38*/
{                       if (ProbBarr_veh_small<0) {
                                return (0);}
                        if (ProbBarr_veh_small>38) {
                                return (0);}
                        if (ProbBarr_veh_small>=0 && ProbBarr_veh_small <=19) {
                                return (ProbBarr_veh_small/19);}
                        if (ProbBarr_veh_small>19 &&  ProbBarr_veh_small <=38) {
                                return ((-1)*ProbBarr_veh_small/19 + 2);}
                return (-1);
}


float ProbBarr_veh_medium(float ProbBarr_veh_medium) /*initial value 19 and 57*/
{                       if (ProbBarr_veh_medium<19) {
                                return (0);}
                        if (ProbBarr_veh_medium>57) {
                                return (0);}
                        if (ProbBarr_veh_medium>=19 && ProbBarr_veh_medium<=38) {
                                return (ProbBarr_veh_medium/19 - 1);}
                        if (ProbBarr_veh_medium>38 &&  ProbBarr_veh_medium<=57) {
                                return ((-1)*ProbBarr_veh_medium/19 + 3);}
                return (-1);
}


float ProbBarr_veh_many(float ProbBarr_veh_many) /*initial value 38 and 57*/
{                       if (ProbBarr_veh_many<38) {
                                return (0);}
                        if (ProbBarr_veh_many>76) {
                                return (0);}
                        if (ProbBarr_veh_many>=38 && ProbBarr_veh_many<=57) {
                                return (ProbBarr_veh_many/19 - 2);}
                        if (ProbBarr_veh_many>57 &&  ProbBarr_veh_many<=76) {
                                return ((-1)*ProbBarr_veh_many/19 + 4);}
                return (-1);
}


// Input 2 definition - Phase D Queue Membership functions

float ProbBq_veh_few (float ProbBq_veh_few) /*initial value 0 and 19*/
{                       if (ProbBq_veh_few<0) {
                                return (0);}
                        if (ProbBq_veh_few>19) {
                                return (0);}
                        if (ProbBq_veh_few>=-19 && ProbBq_veh_few<=0) {
                                return (ProbBq_veh_few/19 +1);}
                        if (ProbBq_veh_few>0 && ProbBq_veh_few <=19) {
                                return ((-1)*ProbBq_veh_few/19 + 1);}
                return (-1);
}
```

```
float ProbBq_veh_small(float ProbBq_veh_small) /*initial value 0 and 38*/
{                               if (ProbBq_veh_small<0) {
                                        return (0);}
                                if (ProbBq_veh_small>38) {
                                        return (0);}
                                if (ProbBq_veh_small>=0 && ProbBq_veh_small <=19) {
                                        return (ProbBq_veh_small/19);}
                                if (ProbBq_veh_small>19 &&  ProbBq_veh_small <=38) {
                                        return ((-1)*ProbBq_veh_small/19 + 2);}
                return (-1);
}


float ProbBq_veh_medium(float ProbBq_veh_medium) /*initial value 19 and 57*/
{                               if (ProbBq_veh_medium < 19) {
                                        return (0);}
                                if (ProbBq_veh_medium>57) {
                                        return (0);}
                                if (ProbBq_veh_medium>=19 && ProbBq_veh_medium<=38) {
                                        return (ProbBq_veh_medium/19 - 1);}
                                if (ProbBq_veh_medium>38 &&  ProbBq_veh_medium<=57) {
                                        return ((-1)*ProbBq_veh_medium/19 + 3);}
                return (-1);
}


float ProbBq_veh_many(float ProbBq_veh_many) /*initial value 38 and 57*/
{                               if (ProbBq_veh_many<38) {
                                        return (0);}
                                if (ProbBq_veh_many>76) {
                                        return (0);}
                                if (ProbBq_veh_many>=38 && ProbBq_veh_many<=57) {
                                        return (ProbBq_veh_many/19 - 2);}
                                if (ProbBq_veh_many>57 &&  ProbBq_veh_many<=76) {
                                        return ((-1)*ProbBq_veh_many/19 + 4);}
                return (-1);
}

 float BflcExtensionRate (float ProbBarr_veh, float Bq_veh)

{
  // Evaluate each rule
                float rule[16];
                rule[0]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[1]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[2]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[3]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_many(Bq_veh));
                rule[4]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[5]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[6]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[7]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_many(Bq_veh));
                rule[8]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[9]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[10]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[11]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_many(Bq_veh));
                rule[12]=MIN(ProbBarr_veh_many(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[13]=MIN(ProbBarr_veh_many(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[14]=MIN(ProbBarr_veh_many(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[15]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_many(Bq_veh));

 // The weighted sum of each class outccome
 // f_ext_zero is 0, f_ext_short is 1, f_ext_medium is 2, f_ext_long is 3
    float f_prob_ext_class[4];
                f_prob_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
    f_prob_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
    f_prob_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
    f_prob_ext_class[3]=rule[12];

 //defuzzification (discreted centroids method)
                float centroid=0;
                float area=0;
                float num=0;
                float den=0;
                float f_ext;
```

```
        for(int i=0;i<4;i++)
        {
                if(i==0)
                { area=8;
                  centroid=5.33f;}
                else if(i==1)
                { area=16;
                  centroid=16;}
                else if(i==2)
                { area=16;
                        centroid=32;}
                else
                { area=8;
                  centroid=42.67f;}

                  num+=f_prob_ext_class[i]*area*centroid;
                  den+=f_prob_ext_class[i]*area;
                }

 // calculate extension rate and rescale to LL and HH range
                f_ext=num/den;
 // std::cout << "extension_rate\t"<< f_ext << std::endl;
        return(f_ext);

}
```

4. PFLCF.cxx (Probabilistic Fuzzy Logic F Phase Timing)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "FLCF.h"


//input1 definition - Phase D Arrival Membership functions

float ProbFarr_veh_few (float ProbFarr_veh_few) /*initial value 0 and 19*/
{                       if (ProbFarr_veh_few<0) {
                                return (0);}
                        if (ProbFarr_veh_few>19) {
                                return (0);}
                        if (ProbFarr_veh_few>=-19 && ProbFarr_veh_few<=0) {
                                return (ProbFarr_veh_few/19 +1);}
                        if (ProbFarr_veh_few>0 && ProbFarr_veh_few <=19) {
                                return ((-1)*ProbFarr_veh_few/19 + 1);}
        return (-1);
}


float ProbFarr_veh_small(float ProbFarr_veh_small) /*initial value 0 and 38*/
{                       if (ProbFarr_veh_small<0) {
                                return (0);}
                        if (ProbFarr_veh_small>38) {
                                return (0);}
                        if (ProbFarr_veh_small>=0 && ProbFarr_veh_small <=19) {
                                return (ProbFarr_veh_small/19);}
                        if (ProbFarr_veh_small>19 &&  ProbFarr_veh_small <=38) {
                                return ((-1)*ProbFarr_veh_small/19 + 2);}
        return (-1);
}


float ProbFarr_veh_medium(float ProbFarr_veh_medium) /*initial value 19 and 57*/
{                       if (ProbFarr_veh_medium<19) {
                                return (0);}
                        if (ProbFarr_veh_medium>57) {
                                return (0);}
                        if (ProbFarr_veh_medium>=19 && ProbFarr_veh_medium<=38) {
                                return (ProbFarr_veh_medium/19 - 1);}
                        if (ProbFarr_veh_medium>38 &&  ProbFarr_veh_medium<=57) {
                                return ((-1)*ProbFarr_veh_medium/19 + 3);}
        return (-1);
}
```

```
float ProbFarr_veh_many(float ProbFarr_veh_many) /*initial value 38 and 57*/
{                               if (ProbFarr_veh_many<38) {
                                        return (0);}
                                if (ProbFarr_veh_many>76) {
                                        return (0);}
                                if (ProbFarr_veh_many>=38 && ProbFarr_veh_many<=57) {
                                        return (ProbFarr_veh_many/19 - 2);}
                                if (ProbFarr_veh_many>57 &&  ProbFarr_veh_many<=76) {
                                        return ((-1)*ProbFarr_veh_many/19 + 4);}
                return (-1);
}


// Input 2 definition - Phase D Queue Membership functions

float ProFq_veh_few (float ProFq_veh_few) /*initial value 0 and 19*/
{                               if (ProFq_veh_few<0) {
                                        return (0);}
                                if (ProFq_veh_few>19) {
                                        return (0);}
                                if (ProFq_veh_few>=-19 && ProFq_veh_few<=0) {
                                        return (ProFq_veh_few/19 +1);}
                                if (ProFq_veh_few>0 && ProFq_veh_few <=19) {
                                        return ((-1)*ProFq_veh_few/19 + 1);}
                return (-1);
}


float ProFq_veh_small(float ProFq_veh_small) /*initial value 0 and 38*/
{                               if (ProFq_veh_small<0) {
                                        return (0);}
                                if (ProFq_veh_small>38) {
                                        return (0);}
                                if (ProFq_veh_small>=0 && ProFq_veh_small <=19) {
                                        return (ProFq_veh_small/19);}
                                if (ProFq_veh_small>19 &&  ProFq_veh_small <=38) {
                                        return ((-1)*ProFq_veh_small/19 + 2);}
                return (-1);
}


float ProFq_veh_medium(float ProFq_veh_medium) /*initial value 19 and 57*/
{                               if (ProFq_veh_medium < 19) {
                                        return (0);}
                                if (ProFq_veh_medium>57) {
                                        return (0);}
                                if (ProFq_veh_medium>=19 && ProFq_veh_medium<=38) {
                                        return (ProFq_veh_medium/19 - 1);}
                                if (ProFq_veh_medium>38 &&  ProFq_veh_medium<=57) {
                                        return ((-1)*ProFq_veh_medium/19 + 3);}
                return (-1);
}


float ProFq_veh_many(float ProFq_veh_many) /*initial value 38 and 57*/
{                               if (ProFq_veh_many<38) {
                                        return (0);}
                                if (ProFq_veh_many>76) {
                                        return (0);}
                                if (ProFq_veh_many>=38 && ProFq_veh_many<=57) {
                                        return (ProFq_veh_many/19 - 2);}
                                if (ProFq_veh_many>57 &&  ProFq_veh_many<=76) {
                                        return ((-1)*ProFq_veh_many/19 + 4);}
                return (-1);
}

 float FflcExtensionRate (float ProbFarr_veh, float ProFq_veh)

{
  // Evaluate each rule
                float rule[16];
                rule[0]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[1]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[2]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
```

```
                rule[3]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_many(ProFq_veh));
                rule[4]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[5]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[6]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
                rule[7]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_many(ProFq_veh));
                rule[8]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[9]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[10]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
                rule[11]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_many(ProFq_veh));
                rule[12]=MIN(ProbFarr_veh_many(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[13]=MIN(ProbFarr_veh_many(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[14]=MIN(ProbFarr_veh_many(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
                rule[15]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_many(ProFq_veh));

 // The weighted sum of each class outccome
 // f_ext_zero is 0, f_ext_short is 1, f_ext_medium is 2, f_ext_long is 3
    float f_prob_ext_class[4];
                f_prob_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
    f_prob_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
    f_prob_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
    f_prob_ext_class[3]=rule[12];

 //defuzzification (discreted centroids method)
                float centroid=0;
                float area=0;
                float num=0;
                float den=0;
                float f_ext;

                for(int i=0;i<4;i++)
                {
                        if(i==0)
                        { area=8;
                          centroid=5.33f;}
                        else if(i==1)
                        { area=16;
                          centroid=16;}
                        else if(i==2)
                        { area=16;
                                centroid=32;}
                        else
                        { area=8;
                          centroid=42.67f;}

                          num+=f_prob_ext_class[i]*area*centroid;
                          den+=f_prob_ext_class[i]*area;
                       }

 // calculate extension rate and rescale to LL and HH range
                f_ext=num/den;
 // std::cout << "extension_rate\t"<< f_ext << std::endl;
               return(f_ext);

}
```

5. FLCD.h (Probabilistic Fuzzy Logic D Phase Timing Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>

#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif

//input1: (arr_veh)arriving vehicles declaration

float ProbDarr_veh_few(float);
float ProbDarr_veh_small(float);
float ProbDarr_veh_medium(float);
float ProbDarr_veh_many(float);
```

```
//input2: (q_veh) queueing vehciles declaration
float ProbDq_veh_few(float);
float ProbDq_veh_small(float);
float ProbDq_veh_medium(float);
float ProbDq_veh_many(float);

//FLC declaration

float DflcExtensionRate (float, float);
```

6. FLCB.h (Probabilistic Fuzzy Logic B Phase Timing Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>

#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif

//input1: (arr_veh)arriving vehicles declaration

float ProbBarr_veh_few(float);
float ProbBarr_veh_small(float);
float ProbBarr_veh_medium(float);
float ProbBarr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float ProbBq_veh_few(float);
float ProbBq_veh_small(float);
float ProbBq_veh_medium(float);
float ProbBq_veh_many(float);

//FLC declaration

float BflcExtensionRate (float, float);
```

7. FLCF.h (Probabilistic Fuzzy Logic F Phase Timing Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>

#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif

//input1: (arr_veh)arriving vehicles declaration

float ProbFarr_veh_few(float);
float ProbFarr_veh_small(float);
float ProbFarr_veh_medium(float);
float ProbFarr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float ProFq_veh_few(float);
float ProFq_veh_small(float);
float ProFq_veh_medium(float);
float ProFq_veh_many(float);

//FLC declaration

float FflcExtensionRate (float, float);
```

1. AAPI.cxx (Main file)

```
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include <stdio.h>
#include "PFLC.h"
char astring[128];

int AAPILoad()
{
	return 0;
}

int AAPIInit()
{
	ANGConnEnableVehiclesInBatch(true);
	return 0;
}

int AAPIManage(double time, double timeSta, double timTrans, double acicle)
{

	//Detectors setup
	float D_v_s, D_up_flow, D_down_flow, D_up_occ, D_up_speed, D_down_speed, D_queue_occ, D_checkin_occ, D_int_queue_occ;

	//Read detectors from AIMSUN
	D_up_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(248,NULL));//D-up
	D_up_speed=(float)(AKIDetGetSpeedAggregatedbyId(248,NULL));//D-up
	D_up_flow=(float)(60*(AKIDetGetCounterAggregatedbyId(248,NULL)));//D-up
	D_down_speed=(float)(AKIDetGetSpeedAggregatedbyId(250,NULL));//D-down
	D_queue_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(249,NULL));//D-queue
	D_checkin_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(247,NULL));//D-checkin
	D_down_flow=(float)(60*(AKIDetGetCounterAggregatedbyId(250,NULL)));//Dstr. flow
	D_v_s=(float)(D_down_flow/2800); //Dstr. V/C ratio

	//Interchange Queue Occupancy from the connected intersections
	D_int_queue_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(1659,NULL);

	//Initialize the inputs
	float local_occ[7], local_speed[7], local_flow[7], downstream_vc[3], downstream_speed[3], checkin_occ[3], queue_occ[3],
	int_queue_occ[3];

	//Upstream speed (input 1)
	if(D_up_speed>=0)
	{local_speed[0]=D_up_speed;}
	else
	{local_speed[0] = 0;}
	local_speed[1]=32; local_speed[2]=0;
	local_speed[3]=32; local_speed[4]=75;
	local_speed[5]=32; local_speed[6]=150;

	//Upstream Flow (input 2)
	if(D_up_flow>=0)
	{local_flow[0]=D_up_flow;}
	else
	{local_flow[0] = 0;}
	local_flow[1]=850; local_flow[2]=0;
	local_flow[3]=850; local_flow[4]=2000;
	local_flow[5]=850; local_flow[6]=4000;

	//Upstream Occupancy (input 3)
	if(D_up_occ>=0)
	{local_occ[0]=D_up_occ;}
	else
	{local_occ[0] = 0;}
	local_occ[1]=6.4f; local_occ[2]=0;
	local_occ[3]=6.4f; local_occ[4]=15;
	local_occ[5]=6.4f; local_occ[6]=30;

	//Downstream V/C input 4
	if(D_v_s>=0)
	{downstream_vc[0]=D_v_s;}
	else
	{downstream_vc[0] = 0;}
```

```
	downstream_vc[1]=6.5; downstream_vc[2]=0.5;

	//Downstream speed input 5
	if(D_down_speed>=0)
	{downstream_speed[0]=D_down_speed;}
	else
	{downstream_speed[0] = 0;}
	downstream_speed[1]=-0.25; downstream_speed[2]=65;

	//Check-in occupancy input 6
	if(D_checkin_occ>=0)
	{checkin_occ[0]=D_checkin_occ;}
	else
	{checkin_occ[0] = 0;}
	checkin_occ[1]=0.4f; checkin_occ[2]=20;

	//Queue occupancy input 7
	if(D_queue_occ>=0)
	{queue_occ[0]=D_queue_occ;}
	else
	{queue_occ[0] = 0;}
	queue_occ[1]=0.4f; queue_occ[2]=20;

	//Interchange queue occupancy input 8
	if(D_int_queue_occ>=0)
	{int_queue_occ[0]=D_queue_occ;}
	else
	{int_queue_occ[0] = 0;}
	int_queue_occ[1]=0.12f; int_queue_occ[2]=60;

	//calculating FLC metering rate
	float flow_rate;
	flow_rate=flcMeterRate(local_occ, local_speed, local_flow, downstream_vc, downstream_speed, checkin_occ, queue_occ,
	int_queue_occ);


	//meter setup
	int MeterID_D;
	MeterID_D=ECIGetMeteringIdSection(0);

	//calculating cycle time and green time
	static int i=1;
	if(i==80)
	{		ECIChangeParametersFlowMeteringById(245,timeSta,flow_rate,flow_rate,flow_rate);
			i=0;
	}
	i=i+1;

	sprintf_s(astring,"Meter_rate is %f\n",flow_rate);
	AKIPrintString(astring);

	return 0;
}
```

2. PFLC.h (Probabilistic Fuzzy Logic Ramp Metering Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif
//input1 declaration
float Problocal_speed_med(float, float=32, float=75);
float Problocal_speed_high(float, float=32, float=150);
float Problocal_speed_low(float, float=32, float=0);
//input2 declaration
float Problocal_flow_med(float, float=850, float=2000);
float Problocal_flow_high(float, float=850, float=4000);
float Problocal_flow_low(float, float=850, float=0);
//input3 declaration
float Problocal_occ_med(float, float=6.4, float=15);
float Problocal_occ_high(float, float=6.4, float=30);
```

```
float Problocal_occ_low(float, float=6.4, float=0);
//input4 delcaration
float Probdownstream_vc_high(float, float=6.5, float=0.5);
//input5 delcaration
float Probdownstream_speed_low(float, float=-0.25, float=65);
//input6 delcaration
float Probcheckin_occ_high(float, float=0.4, float=20);
//input7 declaration
float Probqueue_occ_high(float, float=0.4, float=20);
//input8 declaration
float Probint_queue_occ_high(float, float=0.12, float=60);
//FLC declaration
float ProbflcMeterRate(float *, float *, float *, float *, float *, float *, float *, float*);
```

3. PFLC.cpp (Probabilistic Fuzzy Logic Ramp Metering C++ File)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "PFLC.h"

//input1 definition

float Problocal_speed_med(float local_speed, float q /*initial value 25.5*/, float c/*intial value 60*/)
{ double u;
  float v;
  if(Problocal_speed<0)
                {return(0);}
  if(Problocal_speed>150)
                {return(0);}
  if(Problocal_speed>=0&&local_speed<=150)
                {v=-((Problocal_speed-c)*( Problocal_speed-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_speed_high(float Problocal_speed, float q, float c)
{ double u;
  float v;
  if(Problocal_speed<0)
                {return(0);}
  if(Problocal_speed>150)
                {return(1);}
  if(Problocal_speed>=0&&local_speed<=150)
                {v=-((Problocal_speed-c)*( Problocal_speed-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_speed_low(float Problocal_speed, float q, float c)
{ double u;
  float v;
  if(Problocal_speed<0)
                {return(1);}
  if(Problocal_speed>150)
                {return(0);}
  if(Problocal_speed>=0&&Problocal_speed<=150)
                {v=-((Problocal_speed-c)*( Problocal_speed-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}

//input2 definition

float Problocal_flow_med(float Problocal_flow, float q/*initial value 850 */, float c/*intial value 2000*/)
{ double u;
  float v;
  if(Problocal_flow<0)
                {return(0);}
  if(Problocal_flow>4000)
                {return(0);}
  if(Problocal_flow>=0&&local_flow<=4000)
                {v=-((Problocal_flow-c)*( Problocal_flow-c))/(2*q*q);
                u=exp(double(v));
```

```
                return (float(u));}
 return -1;
}
float Problocal_flow_high(float Problocal_flow, float q, float c)
{ double u;
 float v;
 if(Problocal_flow<0)
                {return(0);}
 if(Problocal_flow>4000)
                {return(1);}
 if(Problocal_flow>=0&&local_flow<=4000)
                {v=-((Problocal_flow-c)*( Problocal_flow-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
 return -1;
}
float Problocal_flow_low(float Problocal_flow, float q, float c)
{ double u;
 float v;
 if(Problocal_flow<0)
                {return(1);}
 if(Problocal_flow>4000)
                {return(0);}
 if(Problocal_flow>=0&&local_flow<=4000)
                {v=-((Problocal_flow-c)*( Problocal_flow-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
 return -1;
}

//input3 definition

float Problocal_occ_med(Probfloat local_occ, float q/*initial value 6.4 */, float c/*intial value 15*/)
{ double u;
 float v;
 if(Problocal_occ<0)
                {return(0);}
 if(Problocal_occ>30)
                {return(0);}
 if(Problocal_occ>=0&&local_occ<=30)
                {v=-((Problocal_occ-c)*( Problocal_occ-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
 return -1;
}
float Problocal_occ_high(float local_occ, float q, float c)
{ double u;
 float v;
 if(Problocal_occ<0)
                {return(0);}
 if(Problocal_occ>30)
                {return(1);}
 if(Problocal_occ>=0&&local_occ<=30)
                {v=-((Problocal_occ-c)*( Problocal_occ-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
 return -1;
}
float Problocal_occ_low(float Problocal_occ, float q, float c)
{ double u;
 float v;
 if(Problocal_occ<0)
                {return(1);}
 if(Problocal_occ>30)
                {return(0);}
 if(Problocal_occ>=0&&Problocal_occ<=30)
                {v=-((Problocal_occ-c)*(Problocal_occ-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
 return -1;
}
```

```
//input4 definition

float Probdownstream_vc_high(float Probdownstream_vc, float q/*initial value 6.5*/, float c)
{double u;
 float v;
 if(Probdownstream_vc<0)
                {return(0);}
 if(Probdownstream_vc>1)
                {return(1);}
 if(Probdownstream_vc>=0&&Probdownstream_vc<=1)
                {v=-q*(Probdownstream_vc-c);
                u=1/(1+exp(double(v)));
                return (float(u));}
 return -1;
}



//input5 definition

float Probdownstream_speed_low(float Probdownstream_speed, float q/*initial value -0.25*/, float c)
{double u;
 float v;
 if(Probdownstream_speed<0)
                {return(1);}
 if(Probdownstream_speed>100)
                {return(0);}
 if(Probdownstream_speed>=0&&Probdownstream_speed<=100)
                {v=-q*(Probdownstream_speed-c);
                u=1/(1+exp(double(v)));
                return (float(u));}
 return -1;
}

//input6 definition

float Probcheckin_occ_high(float Probcheckin_occ, float q/*initial value 0.4*/, float c)
{double u;
 float v;
 if(Probcheckin_occ<0)
                {return(0);}
 if(Probcheckin_occ>50)
                {return(1);}
 if(Probcheckin_occ>=0&&Probcheckin_occ<=50)
                {v=-q*(Probcheckin_occ-c);
                 u=1/(1+exp(double(v)));
                 return (float(u));}
 return -1;
}


//input7 definition

float Probqueue_occ_high(float Probqueue_occ, float q/*initial value 0.4*/, float c)
{double u;
 float v;
 if(Probqueue_occ<0)
                {return(0);}
 if(Probqueue_occ>50)
                {return(1);}
 if(Probqueue_occ>=0&&Probqueue_occ<=50)
                {v=-q*(Probqueue_occ-c);
                 u=1/(1+exp(double(v)));
                 return (float(u));}
 return -1;
}

//input 8 definition - Interchange Queue Occupancy

 float Probint_queue_occ_high(float Probint_queue_occ, float q/*initial value 0.12*/, float c)
{double u;
 float v;
 if(Probint_queue_occ<0)
                {return(0);}
 if(Probint_queue_occ>50)
                {return(1);}
```

```
 if(Probint_queue_occ>=0&&Probint_queue_occ<=90)
                {v=-q*(Probint_queue_occ-c);
                 u=1/(1+exp(double(v)));
                 return (float(u));}
  return -1;}

 // FLC function

 float flcMeterRate(float *Problocal_occ, float *Problocal_speed, float *Problocal_flow, float *Probdownstream_vc, float
 *Probdownstream_speed, float *Probcheckin_occ, float *Probqueue_occ, float *Probint_queue_occ)
 {

 //RULES EVALUATION (10 rules)

 float rule[10];
 rule[0]= Problocal_occ_low(*local_occ,*(local_occ+1),*(local_occ+2));
 rule[1]= Problocal_occ_med(*local_occ,*(local_occ+3),*(local_occ+4));
 rule[2]= Problocal_occ_high(*local_occ,*(local_occ+5),*(local_occ+6));
 rule[3]=MIN(Problocal_speed_low(*Problocal_speed,*(Problocal_speed+1),*(Problocal_speed+2)),Problocal_flow_high(*Problocal_flow,*(Prob
 local_flow+5),*(Problocal_flow+6)));
 rule[4]=MIN(Problocal_speed_med(*Problocal_speed,*(Problocal_speed+3),*(Problocal_speed+4)),Problocal_occ_high(*Problocal_occ,*(Problo
 cal_occ+5),*(Problocal_occ+6)));
 rule[5]=MIN(Problocal_speed_med(*Problocal_speed,*(Problocal_speed+3),*(Problocal_speed+4)),Problocal_occ_low(*Problocal_occ,*(Problo
 cal_occ+1),*(Problocal_occ+2)));
 rule[6]=MIN(Problocal_speed_high(*Problocal_speed,*(Problocal_speed+5),*(Problocal_speed+6)),Problocal_flow_low(*Problocal_flow,*(Prob
 local_flow+1),*(Problocal_flow+2)));
 rule[7]=MIN(downstream_speed_low(*downstream_speed,*(downstream_speed+1),*(downstream_speed+2)),downstream_vc_high(*downstrea
 m_vc,*(downstream_vc+1),*(downstream_vc+2)));
 rule[8]=MAX(checkin_occ_high(*checkin_occ,*(checkin_occ+1),*(checkin_occ+2)),queue_occ_high(*queue_occ,*(queue_occ+1),*(queue_occ
 +2)));
 rule[9]=MAX(int_queue_occ_high(*int_queue_occ,*(int_queue_occ+1),*(int_queue_occ+2)),
 queue_occ_high(*queue_occ,*(queue_occ+1),*(queue_occ+2)));


//The weighted sum of each rule class outcome

//meter_rate_high is 0, meter_rate_low is 1, meter_rate_med is 2

 float meter_rate_class[3];
 meter_rate_class[0]=rule[0]*(3/2)+rule[5]*1+rule[6]*1+rule[8]*3+rule[9]*3;
 meter_rate_class[1]=rule[2]*2+rule[3]*2+rule[7]*3;
 meter_rate_class[2]=rule[1]*(3/2)+rule[4]*1;

//DEFUZZIFICATION (discreted centroids method)

 float base=0.5;
 float centroid=0;
 float area=0;
 float num=0;
 float den=0;
 float LL=240;
 float HL=900;
 float meter_rate;

 for(int i=0;i<3;i++)
 {
            if(i==0)
            { area=base/2;
              centroid=1-base/3;}
            else if(i==1)
            { area=base/2;
              centroid=base/3;}
            else
            { area=base;
              centroid=base;}
              num+=meter_rate_class[i]*area*centroid;
              den+=meter_rate_class[i]*area;
              }

// Calculating the metering rate and rescale to LL and HH range

meter_rate = (HL-LL)*(num/den+LL/(HL-LL));
return(meter_rate);
}
```

APPENDIX B

PROBABILISTIC FUZZY LOGIC DIAMOND INTERCHANGES INCORPORATING BICYCLE SIGNAL CONTROL C++ CODE

1. AAPI.cxx (Main file)

```
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include "stdio.h"
#include "FLCB.h"
#include "FLCD.h"
#include "FLCF.h"

char astring[128];

int AAPILoad()
{

	return 0;
}

int AAPIInit()
{

	ANGConnEnableVehiclesInBatch(true);
	return 0;
}

int AAPIManage(double time, double timeSta, double timTrans, double acicle)
{
	static int det1,det2,det3,det4,det5,det6,det7,det8,det9,det10,det11,det12,det13,det14;
	static int det15,det16_1,det16_2,det16_3,det17,det18_1,det18_2,det18_3,det18_4,det19;
	static int det20_1,det20_2,det20_3,det21,det23,det24,det25;

	double light1,light2,light3,light4,light5,light6,light7,light8,light9;

	int Aphase_det=0;
	int Qphase_det=0;

	static int totalAD=0; // Total Arrival in Phase D
	static int totalQD=0; // Total Queue in Phase     D
	static int totalAB=0; // Total Arrival in Phase B
	static int totalQB=0; // Total Queue in Phase     B
	static int totalAF=0; // Total Arrival in Phase F
	static int totalQF=0; // Total Queue in Phase     F

// Read the Instant Counter measure of a detector from AIMSUN (31 detectors)
	det3    = AKIDetGetCounterAggregatedbyId (843, 0);        //light 1
	det4    = AKIDetGetCounterAggregatedbyId (1571,0);        //light 1
	det5    = AKIDetGetCounterAggregatedbyId (1572,0);        //light 1
	det18_1 = AKIDetGetCounterAggregatedbyId (1576,0);    //light 1 advanced loop
	det18_2 = AKIDetGetCounterAggregatedbyId (1575,0);    //light 1 advanced loop
	det18_3 = AKIDetGetCounterAggregatedbyId (1574,0);    //light 1 advanced loop
	det18_4 = AKIDetGetCounterAggregatedbyId (1573,0);    //light 1 advanced loop
	det1    = AKIDetGetCounterAggregatedbyId (1585,0);        //light 2
	det16_3 = AKIDetGetCounterAggregatedbyId (1590,0);    //light 2
	det16_1 = AKIDetGetCounterAggregatedbyId (741,0);     //light 2 advanced loop
	det16_2 = AKIDetGetCounterAggregatedbyId (1589,0);    //light 2 advanced loop
	det2    = AKIDetGetCounterAggregatedbyId (845,0);         //light 3
	det17   = AKIDetGetCounterAggregatedbyId (1540,0);    //light 3 advanced loop
	det12   = AKIDetGetCounterAggregatedbyId (1582,0);    //light 4
	det13   = AKIDetGetCounterAggregatedbyId (745,0);     //light 4
	det23   = AKIDetGetCounterAggregatedbyId (743,0);     //light 4 advanced loop
	det20_3 = AKIDetGetCounterAggregatedbyId (1593,0);    //light 5
	det8    = AKIDetGetCounterAggregatedbyId (1579,0);    //light 5
	det20_1 = AKIDetGetCounterAggregatedbyId  (740,0);    //light 5 advanced loop
	det20_2 = AKIDetGetCounterAggregatedbyId (1592,0);    //light 5 advanced loop
```

```
	det9	= AKIDetGetCounterAggregatedbyId (841,0);	//light 6
	det21	= AKIDetGetCounterAggregatedbyId (1541,0);	//light 6 advanced loop
	det10	= AKIDetGetCounterAggregatedbyId (1580,0);	//light 7
	det11	= AKIDetGetCounterAggregatedbyId (1581,0);	//light 7
	det25	= AKIDetGetCounterAggregatedbyId (1587,0);	//light 7 advanced loop
	det6	= AKIDetGetCounterAggregatedbyId (1578,0);	//light 8
	det7	= AKIDetGetCounterAggregatedbyId (737,0);	//light 8
	det19	= AKIDetGetCounterAggregatedbyId (744,0);	//light 8 advanced loop
	det14	= AKIDetGetCounterAggregatedbyId (844,0);	//light 9
	det15	= AKIDetGetCounterAggregatedbyId (1583,0);	//light 9
	det24	= AKIDetGetCounterAggregatedbyId (742,0);	//light 9 advanced loop


//Read the current running phase from AIMSUN
	Aphase_det = ECIGetCurrentPhase(1148);
	Qphase_det = ECIGetCurrentPhase(1143);


// Read a state of traffic light from AIMSUN (9 traffic lights)
	light1 = ECIGetStateSem(598,3,0,timeSta);//Upper Harbour Drive Traffic Light 1
	light2 = ECIGetStateSem(698,2,0,timeSta);//Constellation Drive Traffic Light 2
	light3 = ECIGetStateSem(698,2,0,timeSta);//Constellation Drive Traffic Light 3
	light4 = ECIGetStateSem(659,2,0,timeSta);//Constellation Drive Traffic Light 4
	light5 = ECIGetStateSem(703,3,0,timeSta);//Upper Harbour Drive Traffic Light 5
	light6 = ECIGetStateSem(703,3,0,timeSta);//Upper Harbour Drive Traffic Light 6
	light7 = ECIGetStateSem(660,2,0,timeSta);//South-bound on-ramp  Traffic Light 7
	light8 = ECIGetStateSem(683,1,0,timeSta);//North-bound off ramp Traffic Light 8
	light9 = ECIGetStateSem(638,2,0,timeSta);//South-bound off ramp Traffic Light 9

//-------------------------------------------------------------------------------
//PHASE D CAR COUNTS, FUZZY GREEN TIME EXTENSION CALCULATION, CHANGING PHASE DURATION
//-------------------------------------------------------------------------------
	static int Dgetphasetwo=0;
	static int c=0;
	static int d=0;
	static int e=0;
	static int f=0;
	static int Dchange=0;
	static int Dcount=0;
	static int Dphase=0;
	int			Dextensionrate=0;


//Movement 1, 2, 3, 5 – PHASE D TOTAL ARRIVAL CALCULATION - GREEN
	if (light1==1 && light5==1 && light2==1 && light3==1)
	{
		if (det1>=0 && det16_3>=0 && det2>=0)
		{
			if (c==8)
			{
				totalAD=det1+det2+det16_3;
				c=0;
			}
		c=c+1;
		}
	}

//Movement 4, 6, 7, 8, 9 – PHASE D TOTAL QUEUE CALCULATION - RED
	if (light6 == 0 || (light4 ==0 && light7==0 && light8==0 && light9==0))
	{
		if (det25>=0 && det23>=0 && det24>=0 && det21>=0 && det19>=0)
		{
			if (d==8) {
				totalQD=det25+det23+det24+det21+det19;
				d=0;
			}
			d=d+1;
		}
	}
//Check the phase status every 1 simulation step
	if (Dgetphasetwo == Aphase_det && Aphase_det>0) {Dchange=0;}
	else {	Dchange=1;}

	if (Aphase_det>0){Dgetphasetwo = Aphase_det;}

//Check if the current phase is Phase D (Phase 1 in AIMSUN)
```

```
		if (Dchange==1)
		{
			if(Aphase_det==1)
			{
				Dphase=1;
			}
		}
//Arrival&Queue car counts for the first 6 seconds (8 in simulation steps) of PHASE D
		if (Dchange==1&&Dphase==1) {Dcount=1;}
		if (Dcount==1)
		{



if (f==8)
{
Dextensionrate= DflcExtensionRate (float(totalAD),float(totalQD));//FLC extesnion rate
sprintf_s(astring, "PHASE D started...!!");
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Arrival Vehicles in Phase D = %d\n",totalAD);
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Queue Vehicles in Phase D = %d\n",totalQD);
AKIPrintString(astring);
sprintf_s(astring,"Fuzzy Green Time Extension for Phase D=%dseconds\n, Dextensionrate);
AKIPrintString(astring);
ECIChangeTimingPhase(1148,1,Dextensionrate,timeSta);//Change the duration of Phase D
ECIChangeTimingPhase(1143,1,Dextensionrate,timeSta);//Change the duration of Phase D
totalAD=0;
totalQD=0;
f=0;
Dcount=0;
Dphase=0;
		}
f=f+1;
}

//-------------------------------------------------------------------------------
//PHASE B CAR COUNTS, PROBABILISTIC FUZZY GREEN TIME EXTENSION CALCULATION, CHANGING PHASE DURATION
//-------------------------------------------------------------------------------

		static int ABgetphasetwo=0;
		static int a=0;
		static int b=0;
		static int r=0;
		static int v=0;
		static int ABchange=0;
		static int ABcount=0;
		static int ABphase=0;
		int	Bextensionrate=0;

//Movement 5, 6, 7, 9 – TOTAL ARRIVAL CALCULATION FOR PHASE B
		if (light5==1 && light6==1 && light7==1 && light9==1)
		{
			if (det21>=0 && det20_1>=0 && det20_2>=0 && det25>=0)
			{
				if (r==8)
				{
					totalAB=det25+det20_1+det20_2+det21;
					r=0;
				}
			r=r+1;
			}
		}

//Movement 1, 2, 3, 4, 8 – TOTAL QUEUE CALCULATION FOR PHASE B
		if (light1==0 && light2==0 && light3==0 && light4==0 && light8==0)
		{
			if (det18_2>=0&&det18_3>=0&&det18_4>=0&&det23>=0&&det25>=0)
			{
				if (v==8) {
					totalQB=det18_2+det18_3+det18_4+det23+det25;v=0;
				}
				v=v+1;
			}
```

```
		}


//Check the phase status every 1 simulation step
		if (ABgetphasetwo==Aphase_det&&Aphase_det>0) {ABchange=0;}
		else {ABchange=1;}
		if (Aphase_det>0) {ABgetphasetwo=Aphase_det; a=0; }


//Check if the current phase is Phase B (Phase 2 in AIMSUN)

		if (ABchange==1) {
			if (Aphase_det==2) {
				ABphase=1;

			}
		}

//Arrival&Queue car counts for the first 6 seconds (8 in simulation steps) of PHASE B
		if (ABchange==1&&ABphase==1) {ABcount=1;}
		if (ABcount==1){
			if (b==8) {
Bextensionrate=BflcExtensionRate (float(totalAB), float(totalQB));//FLC extension rate
sprintf_s(astring, "PHASE B started...!!");
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Arrival Vehicles in Phase B = %d\n",totalAB);
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Queue Vehicles in Phase B = %d\n",totalQB);
AKIPrintString(astring);
sprintf_s(astring,"Fuzzy Green Time Extension for Phase B=%dseconds\n, Bextensionrate);
AKIPrintString(astring);
ECIChangeTimingPhase(1148,2,Bextensionrate,timeSta);//Change the duration of Phase B
ECIChangeTimingPhase(1143,2,Bextensionrate,timeSta);//Change the duration of Phase B
totalAB=0;
totalQB=0;
b=0;
ABcount=0;
ABphase=0;

		}
b=b+1;}

//-------------------------------------------------------------------------------
// PHASE F CAR COUNTS, PROBABILISTIC FUZZY GREEN TIME EXTENSION CALCULATION, CHANGING PHASE DURATION
//-------------------------------------------------------------------------------

		static int Fgetphasetwo=0;
		static int m=0;
		static int n=0;
		static int o=0;
		static int p=0;
		static int Fchange=0;
		static int Fcount=0;
		static int Fphase=0;
		int	Fextensionrate=0;

//Movement 2, 4, 7, 8 – TOTAL ARRIVAL CALCULATION FOR PHASE F - GREEN
		if (light2==1 || (light4==1 && light7==1 && light8==1))
		{
			if (det19>=0 && det13>=0 && det12>=0 && det1>=0 && det16_3>=0 &&det10>=0 && det11>=0)
			{
				if (m==8)
				{
					totalAF=det19+det13+det12+det1+det16_3+det10+det11;
					m=0;
				}
			m=m+1;
			}
		}

//Movement 1, 3, 5, 6, 9 – TOTAL QUEUE CALCULATION FOR PHASE F - RED
		if (light1 == 0 && light3 ==0 && light5==0 && light6==0 && light9==0)
		{
			if (det18_2>=0&&det18_3>=0&&det18_4>=0&&det17>=0&&det20_1>=0&&det20_2>=0&&det24>=0)
			{
				if (n==8) {
```

```
                                        totalQF=det18_2+det18_3+det18_4+det17+det20_1+det20_2+det24;
                                        n=0;
                                }
                                n=n+1;
                        }
                }

//Check the phase status every 1 simulation step
if (Fgetphasetwo == Aphase_det && Qphase_det>0) {Fchange=0;}
else {          Fchange=1; }
if (Aphase_det>0){Fgetphasetwo = Aphase_det; o=0;}

//Check if the current phase is Phase F (Phase 3 in AIMSUN)

                if (Fchange==1)
                {
                        if(Aphase_det==3)
                        {
                                Fphase=1;
                        }
                }
//Arrival&Queue car counts for the first 6 seconds (8 in simulation steps) of PHASE F

                if (Fchange==1&&Fphase==1) {Fcount=1;}
                if (Fcount==1)
                {
                        if (p==16)
                        {
Fextensionrate=FflcExtensionRate (float(totalAF), float(totalQF));//FLC extension rate
sprintf_s(astring, "PHASE F started...!!");
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Arrival Vehicles in Phase F = %d\n",totalAF);
AKIPrintString(astring);
sprintf_s(astring,"Total amount of Queue Vehicles in Phase F = %d\n",totalQF);
AKIPrintString(astring);
sprintf_s(astring,"Fuzzy Green Time Extension for Phase F=%dseconds\n, Fextensionrate);
AKIPrintString(astring);
ECIChangeTimingPhase(1148,3,Fextensionrate,timeSta);//Change the duration of Phase F
ECIChangeTimingPhase(1143,3,Fextensionrate,timeSta);//Change the duration of Phase F
totalAF=0;
totalQF=0;
p=0;
Fcount=0;
Fphase=0;
                        }
                        p=p+1;
                }

return 0;}
```

2. PFLDI.cxx (Fuzzy Logic D Phase Timing)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "FLCD.h"


//input1 definition - Phase D Arrival Membership functions

float ProbDarr_veh_few (float ProbDarr_veh_few) /*initial value 0 and 19*/
{                       if (ProbDarr_veh_few<0) {
                                return (0);}
                        if (ProbDarr_veh_few>19) {
                                return (0);}
                        if (ProbDarr_veh_few>=-19 && ProbDarr_veh_few<=0) {
                                return (ProbDarr_veh_few/19 +1);}
                        if (ProbDarr_veh_few>0 && ProbDarr_veh_few <=19) {
                                return ((-1)*ProbDarr_veh_few/19 + 1);}
                return (-1);
}


float ProbDarr_veh_small(float ProbDarr_veh_small) /*initial value 0 and 38*/
```

```
{                       if (ProbDarr_veh_small<0) {
                                return (0);}
                        if (ProbDarr_veh_small>38) {
                                return (0);}
                        if (ProbDarr_veh_small>=0 && ProbDarr_veh_small <=19) {
                                return (ProbDarr_veh_small/19);}
                        if (ProbDarr_veh_small>19 &&  ProbDarr_veh_small <=38) {
                                return ((-1)*ProbDarr_veh_small/19 + 2);}
        return (-1);
}


float ProbDarr_veh_medium(float ProbDarr_veh_medium) /*initial value 19 and 57*/
{                       if (ProbDarr_veh_medium<19) {
                                return (0);}
                        if (ProbDarr_veh_medium>57) {
                                return (0);}
                        if (ProbDarr_veh_medium>=19 && ProbDarr_veh_medium<=38) {
                                return (ProbDarr_veh_medium/19 - 1);}
                        if (ProbDarr_veh_medium>38 &&  ProbDarr_veh_medium<=57) {
                                return ((-1)*ProbDarr_veh_medium/19 + 3);}
        return (-1);
}


float ProbDarr_veh_many(float ProbDarr_veh_many) /*initial value 38 and 57*/
{                       if (ProbDarr_veh_many<38) {
                                return (0);}
                        if (ProbDarr_veh_many>76) {
                                return (0);}
                        if (ProbDarr_veh_many>=38 && ProbDarr_veh_many<=57) {
                                return (ProbDarr_veh_many/19 - 2);}
                        if (ProbDarr_veh_many>57 &&  ProbDarr_veh_many<=76) {
                                return ((-1)*ProbDarr_veh_many/19 + 4);}
        return (-1);
}


// Input 2 definition - Phase D Queue Membership functions

float ProbDq_veh_few (float ProbDq_veh_few) /*initial value 0 and 19*/
{                       if (ProbDq_veh_few<0) {
                                return (0);}
                        if (ProbDq_veh_few>19) {
                                return (0);}
                        if (ProbDq_veh_few>=-19 && ProbDq_veh_few<=0) {
                                return (ProbDq_veh_few/19 +1);}
                        if (ProbDq_veh_few>0 && ProbDq_veh_few <=19) {
                                return ((-1)*ProbDq_veh_few/19 + 1);}
        return (-1);
}


float ProbDq_veh_small(float ProbDq_veh_small) /*initial value 0 and 38*/
{                       if (ProbDq_veh_small<0) {
                                return (0);}
                        if (ProbDq_veh_small>38) {
                                return (0);}
                        if (ProbDq_veh_small>=0 && ProbDq_veh_small <=19) {
                                return (ProbDq_veh_small/19);}
                        if (ProbDq_veh_small>19 &&  ProbDq_veh_small <=38) {
                                return ((-1)*ProbDq_veh_small/19 + 2);}
        return (-1);
}


float ProbDq_veh_medium(float ProbDq_veh_medium) /*initial value 19 and 57*/
{                       if (ProbDq_veh_medium < 19) {
                                return (0);}
                        if (ProbDq_veh_medium>57) {
                                return (0);}
                        if (ProbDq_veh_medium>=19 && ProbDq_veh_medium<=38) {
                                return (ProbDq_veh_medium/19 - 1);}
                        if (ProbDq_veh_medium>38 &&  ProbDq_veh_medium<=57) {
                                return ((-1)*ProbDq_veh_medium/19 + 3);}
        return (-1);
```

```
}


float ProbDq_veh_many(float ProbDq_veh_many) /*initial value 38 and 57*/
{                               if (ProbDq_veh_many<38) {
                                        return (0);}
                                if (ProbDq_veh_many>76) {
                                        return (0);}
                                if (ProbDq_veh_many>=38 && ProbDq_veh_many<=57) {
                                        return (ProbDq_veh_many/19 - 2);}
                                if (ProbDq_veh_many>57 &&  ProbDq_veh_many<=76) {
                                        return ((-1)*ProbDq_veh_many/19 + 4);}
                return (-1);
}

 float DflcExtensionRate (float ProbDarr_veh, float ProbDq_veh)

{
  // Evaluate each rule
                float rule[16];
                rule[0]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_few(ProbDq_veh));
                rule[1]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_small(ProbDq_veh));
                rule[2]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_medium(ProbDq_veh));
                rule[3]=MIN(ProbDarr_veh_few(ProbDarr_veh),   ProbDq_veh_many(ProbDq_veh));
                rule[4]=MIN(ProbDarr_veh_small(ProbDarr_veh), ProbDq_veh_few(ProbDq_veh));
                rule[5]=MIN(ProbDarr_veh_small(ProbDarr_veh), ProbDq_veh_small(ProbDq_veh));
                rule[6]=MIN(ProbDarr_veh_small(ProbDarr_veh), ProbDq_veh_medium(ProbDq_veh));
                rule[7]=MIN(ProbDarr_veh_small(ProbDarr_veh),           ProbDq_veh_many(ProbDq_veh));
                rule[8]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_few(ProbDq_veh));
                rule[9]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_small(ProbDq_veh));
                rule[10]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_medium(ProbDq_veh));
                rule[11]=MIN(ProbDarr_veh_medium(ProbDarr_veh), ProbDq_veh_many(ProbDq_veh));
                rule[12]=MIN(ProbDarr_veh_many(ProbDarr_veh), ProbDq_veh_few(ProbDq_veh));
                rule[13]=MIN(ProbDarr_veh_many(ProbDarr_veh), ProbDq_veh_small(ProbDq_veh));
                rule[14]=MIN(ProbDarr_veh_many(ProbDarr_veh), ProbDq_veh_medium(ProbDq_veh));
                rule[15]=MIN(ProbDarr_veh_few(ProbDarr_veh), ProbDq_veh_many(ProbDq_veh));

 // The weighted sum of each class outccome
 // f_ext_zero is 0, f_ext_short is 1, f_ext_medium is 2, f_ext_long is 3
    float f_prob_ext_class[4];
                f_prob_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
    f_prob_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
    f_prob_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
    f_prob_ext_class[3]=rule[12];

 //defuzzification (discreted centroids method)
                float centroid=0;
                float area=0;
                float num=0;
                float den=0;
                float f_ext;

                for(int i=0;i<4;i++)
                {
                                if(i==0)
                                { area=8;
                                  centroid=5.33f;}
                                else if(i==1)
                                { area=16;
                                  centroid=16;}
                                else if(i==2)
                                { area=16;
                                                centroid=32;}
                                else
                                { area=8;
                                  centroid=42.67f;}

                                  num+=f_prob_ext_class[i]*area*centroid;
                                  den+=f_prob_ext_class[i]*area;
                               }

 // calculate extension rate and rescale to LL and HH range
                f_ext=num/den;
 // std::cout << "extension_rate\t"<< f_ext << std::endl;
               return(f_ext);
```

```
}
```

3. PFLCB.cxx (Probabilistic Fuzzy Logic B Phase Timing)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "FLC.h"


//input1 definition - Phase D Arrival Membership functions

float ProbBarr_veh_few (float ProbBarr_veh_few) /*initial value 0 and 19*/
{                               if (ProbBarr_veh_few<0) {
                                        return (0);}
                                if (ProbBarr_veh_few>19) {
                                        return (0);}
                                if (ProbBarr_veh_few>=-19 && ProbBarr_veh_few<=0) {
                                        return (ProbBarr_veh_few/19 +1);}
                                if (ProbBarr_veh_few>0 && ProbBarr_veh_few <=19) {
                                        return ((-1)*ProbBarr_veh_few/19 + 1);}
                return (-1);
}


float ProbBarr_veh_small(float ProbBarr_veh_small) /*initial value 0 and 38*/
{                               if (ProbBarr_veh_small<0) {
                                        return (0);}
                                if (ProbBarr_veh_small>38) {
                                        return (0);}
                                if (ProbBarr_veh_small>=0 && ProbBarr_veh_small <=19) {
                                        return (ProbBarr_veh_small/19);}
                                if (ProbBarr_veh_small>19 &&  ProbBarr_veh_small <=38) {
                                        return ((-1)*ProbBarr_veh_small/19 + 2);}
                return (-1);
}


float ProbBarr_veh_medium(float ProbBarr_veh_medium) /*initial value 19 and 57*/
{                               if (ProbBarr_veh_medium<19) {
                                        return (0);}
                                if (ProbBarr_veh_medium>57) {
                                        return (0);}
                                if (ProbBarr_veh_medium>=19 && ProbBarr_veh_medium<=38) {
                                        return (ProbBarr_veh_medium/19 - 1);}
                                if (ProbBarr_veh_medium>38 &&  ProbBarr_veh_medium<=57) {
                                        return ((-1)*ProbBarr_veh_medium/19 + 3);}
                return (-1);
}


float ProbBarr_veh_many(float ProbBarr_veh_many) /*initial value 38 and 57*/
{                               if (ProbBarr_veh_many<38) {
                                        return (0);}
                                if (ProbBarr_veh_many>76) {
                                        return (0);}
                                if (ProbBarr_veh_many>=38 && ProbBarr_veh_many<=57) {
                                        return (ProbBarr_veh_many/19 - 2);}
                                if (ProbBarr_veh_many>57 &&  ProbBarr_veh_many<=76) {
                                        return ((-1)*ProbBarr_veh_many/19 + 4);}
                return (-1);
}


// Input 2 definition - Phase D Queue Membership functions

float ProbBq_veh_few (float ProbBq_veh_few) /*initial value 0 and 19*/
{                               if (ProbBq_veh_few<0) {
                                        return (0);}
                                if (ProbBq_veh_few>19) {
                                        return (0);}
                                if (ProbBq_veh_few>=-19 && ProbBq_veh_few<=0) {
                                        return (ProbBq_veh_few/19 +1);}
                                if (ProbBq_veh_few>0 && ProbBq_veh_few <=19) {
```

```
                                        return ((-1)*ProbBq_veh_few/19 + 1);}
                return (-1);
}


float ProbBq_veh_small(float ProbBq_veh_small) /*initial value 0 and 38*/
{                               if (ProbBq_veh_small<0) {
                                        return (0);}
                                if (ProbBq_veh_small>38) {
                                        return (0);}
                                if (ProbBq_veh_small>=0 && ProbBq_veh_small <=19) {
                                        return (ProbBq_veh_small/19);}
                                if (ProbBq_veh_small>19 &&  ProbBq_veh_small <=38) {
                                        return ((-1)*ProbBq_veh_small/19 + 2);}
                return (-1);
}


float ProbBq_veh_medium(float ProbBq_veh_medium) /*initial value 19 and 57*/
{                               if (ProbBq_veh_medium < 19) {
                                        return (0);}
                                if (ProbBq_veh_medium>57) {
                                        return (0);}
                                if (ProbBq_veh_medium>=19 && ProbBq_veh_medium<=38) {
                                        return (ProbBq_veh_medium/19 - 1);}
                                if (ProbBq_veh_medium>38 &&  ProbBq_veh_medium<=57) {
                                        return ((-1)*ProbBq_veh_medium/19 + 3);}
                return (-1);
}


float ProbBq_veh_many(float ProbBq_veh_many) /*initial value 38 and 57*/
{                               if (ProbBq_veh_many<38) {
                                        return (0);}
                                if (ProbBq_veh_many>76) {
                                        return (0);}
                                if (ProbBq_veh_many>=38 && ProbBq_veh_many<=57) {
                                        return (ProbBq_veh_many/19 - 2);}
                                if (ProbBq_veh_many>57 &&  ProbBq_veh_many<=76) {
                                        return ((-1)*ProbBq_veh_many/19 + 4);}
                return (-1);
}

 float BflcExtensionRate (float ProbBarr_veh, float Bq_veh)

{
  // Evaluate each rule
                float rule[16];
                rule[0]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[1]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[2]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[3]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_many(Bq_veh));
                rule[4]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[5]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[6]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[7]=MIN(ProbBarr_veh_small(ProbBarr_veh), ProbBq_veh_many(Bq_veh));
                rule[8]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[9]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[10]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[11]=MIN(ProbBarr_veh_medium(ProbBarr_veh), ProbBq_veh_many(Bq_veh));
                rule[12]=MIN(ProbBarr_veh_many(ProbBarr_veh), ProbBq_veh_few(Bq_veh));
                rule[13]=MIN(ProbBarr_veh_many(ProbBarr_veh), ProbBq_veh_small(Bq_veh));
                rule[14]=MIN(ProbBarr_veh_many(ProbBarr_veh), ProbBq_veh_medium(Bq_veh));
                rule[15]=MIN(ProbBarr_veh_few(ProbBarr_veh), ProbBq_veh_many(Bq_veh));

 // The weighted sum of each class outccome
 // f_ext_zero is 0, f_ext_short is 1, f_ext_medium is 2, f_ext_long is 3
    float f_prob_ext_class[4];
                f_prob_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
    f_prob_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
    f_prob_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
    f_prob_ext_class[3]=rule[12];

 //defuzzification (discreted centroids method)
                float centroid=0;
                float area=0;
```

```
		float num=0;
		float den=0;
		float f_ext;

		for(int i=0;i<4;i++)
		{
			if(i==0)
			{ area=8;
			  centroid=5.33f;}
			else if(i==1)
			{ area=16;
			  centroid=16;}
			else if(i==2)
			{ area=16;
				centroid=32;}
			else
			{ area=8;
			  centroid=42.67f;}

			  num+=f_prob_ext_class[i]*area*centroid;
			  den+=f_prob_ext_class[i]*area;
			}

 // calculate extension rate and rescale to LL and HH range
			f_ext=num/den;
 // std::cout << "extension_rate\t"<< f_ext << std::endl;
		return(f_ext);

}
```

4. PFLCF.cxx (Probabilistic Fuzzy Logic F Phase Timing)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "FLCF.h"


//input1 definition - Phase D Arrival Membership functions

float ProbFarr_veh_few (float ProbFarr_veh_few) /*initial value 0 and 19*/
{			if (ProbFarr_veh_few<0) {
				return (0);}
			if (ProbFarr_veh_few>19) {
				return (0);}
			if (ProbFarr_veh_few>=-19 && ProbFarr_veh_few<=0) {
				return (ProbFarr_veh_few/19 +1);}
			if (ProbFarr_veh_few>0 && ProbFarr_veh_few <=19) {
				return ((-1)*ProbFarr_veh_few/19 + 1);}
		return (-1);
}


float ProbFarr_veh_small(float ProbFarr_veh_small) /*initial value 0 and 38*/
{			if (ProbFarr_veh_small<0) {
				return (0);}
			if (ProbFarr_veh_small>38) {
				return (0);}
			if (ProbFarr_veh_small>=0 && ProbFarr_veh_small <=19) {
				return (ProbFarr_veh_small/19);}
			if (ProbFarr_veh_small>19 &&  ProbFarr_veh_small <=38) {
				return ((-1)*ProbFarr_veh_small/19 + 2);}
		return (-1);
}


float ProbFarr_veh_medium(float ProbFarr_veh_medium) /*initial value 19 and 57*/
{			if (ProbFarr_veh_medium<19) {
				return (0);}
			if (ProbFarr_veh_medium>57) {
				return (0);}
```

```
                        if (ProbFarr_veh_medium>=19 && ProbFarr_veh_medium<=38) {
                                return (ProbFarr_veh_medium/19 - 1);}
                        if (ProbFarr_veh_medium>38 &&  ProbFarr_veh_medium<=57) {
                                return ((-1)*ProbFarr_veh_medium/19 + 3);}
        return (-1);
}


float ProbFarr_veh_many(float ProbFarr_veh_many) /*initial value 38 and 57*/
{                       if (ProbFarr_veh_many<38) {
                                return (0);}
                        if (ProbFarr_veh_many>76) {
                                return (0);}
                        if (ProbFarr_veh_many>=38 && ProbFarr_veh_many<=57) {
                                return (ProbFarr_veh_many/19 - 2);}
                        if (ProbFarr_veh_many>57 &&  ProbFarr_veh_many<=76) {
                                return ((-1)*ProbFarr_veh_many/19 + 4);}
        return (-1);
}


// Input 2 definition - Phase D Queue Membership functions

float ProFq_veh_few (float ProFq_veh_few) /*initial value 0 and 19*/
{                       if (ProFq_veh_few<0) {
                                return (0);}
                        if (ProFq_veh_few>19) {
                                return (0);}
                        if (ProFq_veh_few>=-19 && ProFq_veh_few<=0) {
                                return (ProFq_veh_few/19 +1);}
                        if (ProFq_veh_few>0 && ProFq_veh_few <=19) {
                                return ((-1)*ProFq_veh_few/19 + 1);}
        return (-1);
}


float ProFq_veh_small(float ProFq_veh_small) /*initial value 0 and 38*/
{                       if (ProFq_veh_small<0) {
                                return (0);}
                        if (ProFq_veh_small>38) {
                                return (0);}
                        if (ProFq_veh_small>=0 && ProFq_veh_small <=19) {
                                return (ProFq_veh_small/19);}
                        if (ProFq_veh_small>19 &&  ProFq_veh_small <=38) {
                                return ((-1)*ProFq_veh_small/19 + 2);}
        return (-1);
}


float ProFq_veh_medium(float ProFq_veh_medium) /*initial value 19 and 57*/
{                       if (ProFq_veh_medium < 19) {
                                return (0);}
                        if (ProFq_veh_medium>57) {
                                return (0);}
                        if (ProFq_veh_medium>=19 && ProFq_veh_medium<=38) {
                                return (ProFq_veh_medium/19 - 1);}
                        if (ProFq_veh_medium>38 &&  ProFq_veh_medium<=57) {
                                return ((-1)*ProFq_veh_medium/19 + 3);}
        return (-1);
}


float ProFq_veh_many(float ProFq_veh_many) /*initial value 38 and 57*/
{                       if (ProFq_veh_many<38) {
                                return (0);}
                        if (ProFq_veh_many>76) {
                                return (0);}
                        if (ProFq_veh_many>=38 && ProFq_veh_many<=57) {
                                return (ProFq_veh_many/19 - 2);}
                        if (ProFq_veh_many>57 &&  ProFq_veh_many<=76) {
                                return ((-1)*ProFq_veh_many/19 + 4);}
        return (-1);
}

 float FflcExtensionRate (float ProbFarr_veh, float ProFq_veh)
```

```
{
  // Evaluate each rule
                float rule[16];
                rule[0]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[1]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[2]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
                rule[3]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_many(ProFq_veh));
                rule[4]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[5]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[6]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
                rule[7]=MIN(ProbFarr_veh_small(ProbFarr_veh), ProFq_veh_many(ProFq_veh));
                rule[8]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[9]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[10]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
                rule[11]=MIN(ProbFarr_veh_medium(ProbFarr_veh), ProFq_veh_many(ProFq_veh));
                rule[12]=MIN(ProbFarr_veh_many(ProbFarr_veh), ProFq_veh_few(ProFq_veh));
                rule[13]=MIN(ProbFarr_veh_many(ProbFarr_veh), ProFq_veh_small(ProFq_veh));
                rule[14]=MIN(ProbFarr_veh_many(ProbFarr_veh), ProFq_veh_medium(ProFq_veh));
                rule[15]=MIN(ProbFarr_veh_few(ProbFarr_veh), ProFq_veh_many(ProFq_veh));

 // The weighted sum of each class outccome
 // f_ext_zero is 0, f_ext_short is 1, f_ext_medium is 2, f_ext_long is 3
    float f_prob_ext_class[4];
                f_prob_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
    f_prob_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
    f_prob_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
    f_prob_ext_class[3]=rule[12];

 //defuzzification (discreted centroids method)
                float centroid=0;
                float area=0;
                float num=0;
                float den=0;
                float f_ext;

                for(int i=0;i<4;i++)
                {
                                if(i==0)
                                { area=8;
                                  centroid=5.33f;}
                                else if(i==1)
                                { area=16;
                                  centroid=16;}
                                else if(i==2)
                                { area=16;
                                                centroid=32;}
                                else
                                { area=8;
                                  centroid=42.67f;}

                                  num+=f_prob_ext_class[i]*area*centroid;
                                  den+=f_prob_ext_class[i]*area;
                               }

 // calculate extension rate and rescale to LL and HH range
                f_ext=num/den;
 // std::cout << "extension_rate\t"<< f_ext << std::endl;
              return(f_ext);

}
```

5. PFLCD.h (Probabilistic Fuzzy Logic D Phase Timing Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>

#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif
```

```
//input1: (arr_veh)arriving vehicles declaration

float ProbDarr_veh_few(float);
float ProbDarr_veh_small(float);
float ProbDarr_veh_medium(float);
float ProbDarr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float ProbDq_veh_few(float);
float ProbDq_veh_small(float);
float ProbDq_veh_medium(float);
float ProbDq_veh_many(float);

//FLC declaration

float DflcExtensionRate (float, float);
```

6. PFLCB.h (Probabilistic Fuzzy Logic B Phase Timing Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>

#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif

//input1: (arr_veh)arriving vehicles declaration

float ProbBarr_veh_few(float);
float ProbBarr_veh_small(float);
float ProbBarr_veh_medium(float);
float ProbBarr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float ProbBq_veh_few(float);
float ProbBq_veh_small(float);
float ProbBq_veh_medium(float);
float ProbBq_veh_many(float);

//FLC declaration

float BflcExtensionRate (float, float);
```

7. PFLCF.h (Probabilistic Fuzzy Logic F Phase Timing Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>

#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif

//input1: (arr_veh)arriving vehicles declaration

float ProbFarr_veh_few(float);
float ProbFarr_veh_small(float);
float ProbFarr_veh_medium(float);
float ProbFarr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float ProFq_veh_few(float);
float ProFq_veh_small(float);
float ProFq_veh_medium(float);
float ProFq_veh_many(float);
```

```
//FLC declaration

float FflcExtensionRate (float, float);
```

4. AAPI.cxx (Main file)

```
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include <stdio.h>
#include "PFLC.h"
char astring[128];

int AAPILoad()
{
	return 0;
}

int AAPIInit()
{
	ANGConnEnableVehiclesInBatch(true);
	return 0;
}

int AAPIManage(double time, double timeSta, double timTrans, double acicle)
{

	//Detectors setup
	float D_v_s, D_up_flow, D_down_flow, D_up_occ, D_up_speed, D_down_speed, D_queue_occ, D_checkin_occ, D_int_queue_occ;

	//Read detectors from AIMSUN
	D_up_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(248,NULL));//D-up
	D_up_speed=(float)(AKIDetGetSpeedAggregatedbyId(248,NULL));//D-up
	D_up_flow=(float)(60*(AKIDetGetCounterAggregatedbyId(248,NULL)));//D-up
	D_down_speed=(float)(AKIDetGetSpeedAggregatedbyId(250,NULL));//D-down
	D_queue_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(249,NULL));//D-queue
	D_checkin_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(247,NULL));//D-checkin
	D_down_flow=(float)(60*(AKIDetGetCounterAggregatedbyId(250,NULL)));//Dstr. flow
	D_v_s=(float)(D_down_flow/2800); //Dstr. V/C ratio

	//Interchange Queue Occupancy from the connected intersections
	D_int_queue_occ=(float)(AKIDetGetTimeOccupedAggregatedbyId(1659,NULL);

	//Initialize the inputs
	float local_occ[7], local_speed[7], local_flow[7], downstream_vc[3], downstream_speed[3], checkin_occ[3], queue_occ[3],
	int_queue_occ[3];

	//Upstream speed (input 1)
	if(D_up_speed>=0)
	{local_speed[0]=D_up_speed;}
	else
	{local_speed[0] = 0;}
	local_speed[1]=32; local_speed[2]=0;
	local_speed[3]=32; local_speed[4]=75;
	local_speed[5]=32; local_speed[6]=150;

	//Upstream Flow (input 2)
	if(D_up_flow>=0)
	{local_flow[0]=D_up_flow;}
	else
	{local_flow[0] = 0;}
	local_flow[1]=850; local_flow[2]=0;
	local_flow[3]=850; local_flow[4]=2000;
	local_flow[5]=850; local_flow[6]=4000;

	//Upstream Occupancy (input 3)
	if(D_up_occ>=0)
	{local_occ[0]=D_up_occ;}
```

```
		else
		{local_occ[0] = 0;}
		local_occ[1]=6.4f; local_occ[2]=0;
		local_occ[3]=6.4f; local_occ[4]=15;
		local_occ[5]=6.4f; local_occ[6]=30;

		//Downstream V/C input 4
		if(D_v_s>=0)
		{downstream_vc[0]=D_v_s;}
		else
		{downstream_vc[0] = 0;}
		downstream_vc[1]=6.5; downstream_vc[2]=0.5;

		//Downstream speed input 5
		if(D_down_speed>=0)
		{downstream_speed[0]=D_down_speed;}
		else
		{downstream_speed[0] = 0;}
		downstream_speed[1]=-0.25; downstream_speed[2]=65;

		//Check-in occupancy input 6
		if(D_checkin_occ>=0)
		{checkin_occ[0]=D_checkin_occ;}
		else
		{checkin_occ[0] = 0;}
		checkin_occ[1]=0.4f; checkin_occ[2]=20;

		//Queue occupancy input 7
		if(D_queue_occ>=0)
		{queue_occ[0]=D_queue_occ;}
		else
		{queue_occ[0] = 0;}
		queue_occ[1]=0.4f; queue_occ[2]=20;

		//Interchange queue occupancy input 8
		if(D_int_queue_occ>=0)
		{int_queue_occ[0]=D_queue_occ;}
		else
		{int_queue_occ[0] = 0;}
		int_queue_occ[1]=0.12f; int_queue_occ[2]=60;

		//calculating FLC metering rate
		float flow_rate;
		flow_rate=flcMeterRate(local_occ, local_speed, local_flow, downstream_vc, downstream_speed, checkin_occ, queue_occ,
		int_queue_occ);


		//meter setup
		int MeterID_D;
		MeterID_D=ECIGetMeteringIdSection(0);

		//calculating cycle time and green time
		static int i=1;
		if(i==80)
		{		ECIChangeParametersFlowMeteringById(245,timeSta,flow_rate,flow_rate,flow_rate);
				i=0;
		}
		i=i+1;

		sprintf_s(astring,"Meter_rate is %f\n",flow_rate);
		AKIPrintString(astring);

		return 0;
}
```

5. PFLC.h (Probabilistic Fuzzy Logic Ramp Metering Header File)

```
#include <iostream>
#include <stdio.h>
#include <math.h>
#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
```

```
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif
//input1 declaration
float Problocal_speed_med(float, float=32, float=75);
float Problocal_speed_high(float, float=32, float=150);
float Problocal_speed_low(float, float=32, float=0);
//input2 declaration
float Problocal_flow_med(float, float=850, float=2000);
float Problocal_flow_high(float, float=850, float=4000);
float Problocal_flow_low(float, float=850, float=0);
//input3 declaration
float Problocal_occ_med(float, float=6.4, float=15);
float Problocal_occ_high(float, float=6.4, float=30);
float Problocal_occ_low(float, float=6.4, float=0);
//input4 delcaration
float Probdownstream_vc_high(float, float=6.5, float=0.5);
//input5 delcaration
float Probdownstream_speed_low(float, float=-0.25, float=65);
//input6 delcaration
float Probcheckin_occ_high(float, float=0.4, float=20);
//input7 declaration
float Probqueue_occ_high(float, float=0.4, float=20);
//input8 declaration
float Probint_queue_occ_high(float, float=0.12, float=60);
//FLC declaration
float ProbflcMeterRate(float *, float *, float *, float *, float *, float *, float *, float*);
```

6. PFLC.cpp (Probabilistic Fuzzy Logic Ramp Metering C++ File)

```
#include <stdio.h>
#include <iostream>
#include <math.h>
#include "PFLC.h"

//input1 definition

float Problocal_speed_med(float local_speed, float q /*initial value 25.5*/, float c/*intial value 60*/)
{ double u;
  float v;
  if(Problocal_speed<0)
                {return(0);}
  if(Problocal_speed>150)
                {return(0);}
  if(Problocal_speed>=0&&local_speed<=150)
                {v=-((Problocal_speed-c)*( Problocal_speed-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_speed_high(float Problocal_speed, float q, float c)
{ double u;
  float v;
  if(Problocal_speed<0)
                {return(0);}
  if(Problocal_speed>150)
                {return(1);}
  if(Problocal_speed>=0&&local_speed<=150)
                {v=-((Problocal_speed-c)*( Problocal_speed-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_speed_low(float Problocal_speed, float q, float c)
{ double u;
  float v;
  if(Problocal_speed<0)
                {return(1);}
  if(Problocal_speed>150)
                {return(0);}
  if(Problocal_speed>=0&&Problocal_speed<=150)
                {v=-((Problocal_speed-c)*( Problocal_speed-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
```

```
//input2 definition

float Problocal_flow_med(float Problocal_flow, float q/*initial value 850 */, float c/*intial value 2000*/)
{ double u;
  float v;
  if(Problocal_flow<0)
                {return(0);}
  if(Problocal_flow>4000)
                {return(0);}
  if(Problocal_flow>=0&&local_flow<=4000)
                {v=-((Problocal_flow-c)*( Problocal_flow-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_flow_high(float Problocal_flow, float q, float c)
{ double u;
  float v;
  if(Problocal_flow<0)
                {return(0);}
  if(Problocal_flow>4000)
                {return(1);}
  if(Problocal_flow>=0&&local_flow<=4000)
                {v=-((Problocal_flow-c)*( Problocal_flow-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_flow_low(float Problocal_flow, float q, float c)
{ double u;
  float v;
  if(Problocal_flow<0)
                {return(1);}
  if(Problocal_flow>4000)
                {return(0);}
  if(Problocal_flow>=0&&local_flow<=4000)
                {v=-((Problocal_flow-c)*( Problocal_flow-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}

//input3 definition

float Problocal_occ_med(Probfloat local_occ, float q/*initial value 6.4 */, float c/*intial value 15*/)
{ double u;
  float v;
  if(Problocal_occ<0)
                {return(0);}
  if(Problocal_occ>30)
                {return(0);}
  if(Problocal_occ>=0&&local_occ<=30)
                {v=-((Problocal_occ-c)*( Problocal_occ-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_occ_high(float local_occ, float q, float c)
{ double u;
  float v;
  if(Problocal_occ<0)
                {return(0);}
  if(Problocal_occ>30)
                {return(1);}
  if(Problocal_occ>=0&&local_occ<=30)
                {v=-((Problocal_occ-c)*( Problocal_occ-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
  return -1;
}
float Problocal_occ_low(float Problocal_occ, float q, float c)
{ double u;
  float v;
  if(Problocal_occ<0)
                {return(1);}
```

```
 if(Problocal_occ>30)
                {return(0);}
 if(Problocal_occ>=0&&Problocal_occ<=30)
                {v=-((Problocal_occ-c)*(Problocal_occ-c))/(2*q*q);
                u=exp(double(v));
                return (float(u));}
 return -1;
}




//input4 definition

float Probdownstream_vc_high(float Probdownstream_vc, float q/*initial value 6.5*/, float c)
{double u;
 float v;
 if(Probdownstream_vc<0)
                {return(0);}
 if(Probdownstream_vc>1)
                {return(1);}
 if(Probdownstream_vc>=0&&Probdownstream_vc<=1)
                {v=-q*(Probdownstream_vc-c);
                u=1/(1+exp(double(v)));
                return (float(u));}
 return -1;
}



//input5 definition

float Probdownstream_speed_low(float Probdownstream_speed, float q/*initial value -0.25*/, float c)
{double u;
 float v;
 if(Probdownstream_speed<0)
                {return(1);}
 if(Probdownstream_speed>100)
                {return(0);}
 if(Probdownstream_speed>=0&&Probdownstream_speed<=100)
                {v=-q*(Probdownstream_speed-c);
                u=1/(1+exp(double(v)));
                return (float(u));}
 return -1;
}

//input6 definition

float Probcheckin_occ_high(float Probcheckin_occ, float q/*initial value 0.4*/, float c)
{double u;
 float v;
 if(Probcheckin_occ<0)
                {return(0);}
 if(Probcheckin_occ>50)
                {return(1);}
 if(Probcheckin_occ>=0&&Probcheckin_occ<=50)
                {v=-q*(Probcheckin_occ-c);
                 u=1/(1+exp(double(v)));
                 return (float(u));}
 return -1;
}


//input7 definition

float Probqueue_occ_high(float Probqueue_occ, float q/*initial value 0.4*/, float c)
{double u;
 float v;
 if(Probqueue_occ<0)
                {return(0);}
 if(Probqueue_occ>50)
                {return(1);}
 if(Probqueue_occ>=0&&Probqueue_occ<=50)
                {v=-q*(Probqueue_occ-c);
                 u=1/(1+exp(double(v)));
```

```
                return (float(u));}
  return -1;
}

//input 8 definition - Interchange Queue Occupancy

 float Probint_queue_occ_high(float Probint_queue_occ, float q/*initial value 0.12*/, float c)
{double u;
  float v;
  if(Probint_queue_occ<0)
                {return(0);}
  if(Probint_queue_occ>50)
                {return(1);}
  if(Probint_queue_occ>=0&&Probint_queue_occ<=90)
                {v=-q*(Probint_queue_occ-c);
                 u=1/(1+exp(double(v)));
                 return (float(u));}
  return -1;}

// FLC function

float flcMeterRate(float *Problocal_occ, float *Problocal_speed, float *Problocal_flow, float *Probdownstream_vc, float
*Probdownstream_speed, float *Probcheckin_occ, float *Probqueue_occ, float *Probint_queue_occ)
{

//RULES EVALUATION (10 rules)

float rule[10];
rule[0]= Problocal_occ_low(*local_occ,*(local_occ+1),*(local_occ+2));
rule[1]= Problocal_occ_med(*local_occ,*(local_occ+3),*(local_occ+4));
rule[2]= Problocal_occ_high(*local_occ,*(local_occ+5),*(local_occ+6));
rule[3]=MIN(Problocal_speed_low(*Problocal_speed,*(Problocal_speed+1),*(Problocal_speed+2)),Problocal_flow_high(*Problocal_flow,*(Prob
local_flow+5),*(Problocal_flow+6)));
rule[4]=MIN(Problocal_speed_med(*Problocal_speed,*(Problocal_speed+3),*(Problocal_speed+4)),Problocal_occ_high(*Problocal_occ,*(Problo
cal_occ+5),*(Problocal_occ+6)));
rule[5]=MIN(Problocal_speed_med(*Problocal_speed,*(Problocal_speed+3),*(Problocal_speed+4)),Problocal_occ_low(*Problocal_occ,*(Problo
cal_occ+1),*(Problocal_occ+2)));
rule[6]=MIN(Problocal_speed_high(*Problocal_speed,*(Problocal_speed+5),*(Problocal_speed+6)),Problocal_flow_low(*Problocal_flow,*(Prob
local_flow+1),*(Problocal_flow+2)));
rule[7]=MIN(downstream_speed_low(*downstream_speed,*(downstream_speed+1),*(downstream_speed+2)),downstream_vc_high(*downstrea
m_vc,*(downstream_vc+1),*(downstream_vc+2)));
rule[8]=MAX(checkin_occ_high(*checkin_occ,*(checkin_occ+1),*(checkin_occ+2)),queue_occ_high(*queue_occ,*(queue_occ+1),*(queue_occ
+2)));
rule[9]=MAX(int_queue_occ_high(*int_queue_occ,*(int_queue_occ+1),*(int_queue_occ+2)),
queue_occ_high(*queue_occ,*(queue_occ+1),*(queue_occ+2)));


//The weighted sum of each rule class outcome

//meter_rate_high is 0, meter_rate_low is 1, meter_rate_med is 2

float meter_rate_class[3];
meter_rate_class[0]=rule[0]*(3/2)+rule[5]*1+rule[6]*1+rule[8]*3+rule[9]*3;
meter_rate_class[1]=rule[2]*2+rule[3]*2+rule[7]*3;
meter_rate_class[2]=rule[1]*(3/2)+rule[4]*1;

//DEFUZZIFICATION (discreted centroids method)

float base=0.5;
float centroid=0;
float area=0;
float num=0;
float den=0;
float LL=240;
float HL=900;
float meter_rate;

for(int i=0;i<3;i++)
{
                if(i==0)
                { area=base/2;
                  centroid=1-base/3;}
                else if(i==1)
                { area=base/2;
                  centroid=base/3;}
                else
```

```
                    { area=base;
                      centroid=base;}
                      num+=meter_rate_class[i]*area*centroid;
                      den+=meter_rate_class[i]*area;
                     }

// Calculating the metering rate and rescale to LL and HH range

meter_rate = (HL-LL)*(num/den+LL/(HL-LL)); return(meter_rate); }
```

JOURNAL OF TRANSPORTATION ENGINEERING
Volume 141, Issue 3 (March 2015)
Published online in American Society of Civil Engineers.
DOI: 10.1061/(ASCE)TE.1943-5436.0000735
ISSN: 1943-5436

Probabilistic Fuzzy Logic Signal and Ramp-Metering at a Diamond Interchange

V.C. Pham[1]; W. L. Xu[2]; F. Alam, Ph.D.[3]; and F. C. Fang[4]
[1]Ph.D. Student, School of Engineering and Advanced Technology, Massey University, Albany Village, Dairy Flat Highway, Auckland 0632, New Zealand
[2]Chair Professor, Department of Mechanical Engineering, The University of Auckland, 20 Symonds Street, Auckland 1142, New Zealand (corresponding author).
[3]Senior Lecturer, School of Engineering and Advanced Technology, Massey University, Albany Village, Dairy Flat Highway, Auckland 0632, New Zealand
[4]Associate Professor, Department of Civil, Environmental and Biomedical Engineering, University of Hartford, 200 Bloomfield Ave., West Hartford, CT 06117, USA

Abstract

In this paper a probabilistic fuzzy logic control is developed for the **signalised** control of a diamond interchange to improve the traffic flow on the surface streets and highways. The signal phasing, green time extension and ramp metering are decided in response to real time traffic conditions. The probabilistic fuzzy logic for diamond interchange (PFLDI) is comprised of three modules: probabilistic fuzzy phase timing (PFPT) that controls the green time extension process of the current running phase, phase selection logic (PSL) which decides the next phase based on the pre-setup phase logic by the local transport authority and probabilistic fuzzy ramp-metering (PFRM) that determines the on-ramp metering rate based on the traffic conditions of the arterial streets and highways. We used Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network (AIMSUN) software to model the diamond interchange and measure the effectiveness of the proposed PFLDI algorithm. We compared the performance of the PFLDI with that of an actuated diamond interchange (ADI) control based on the ALINEA algorithm and a conventional fuzzy logic diamond interchange algorithm (FLDI). Simulation results show that the PFLDI lowers System Total Travel Time and Average Delay, improves the Downstream Average Speed and lowers the Downstream Average Delay compared to ADI and FLDI.

Keywords: traffic signals, diamond interchange, probabilistic fuzzy logic control, adaptive control, traffic simulation

I**ntroduction**

Interchanges are designed to improve traffic flow in urban areas; they are carefully designed to enable the smooth flow of traffic from arterial streets to and from the highways. There are many types of interchanges and the diamond interchange is one of the most well-known interchanges. A diamond interchange consists of two closely spaced intersections, being less than 250 meters apart and each connecting directly to a highway direction (Fang 2004). The short distance between the two intersections limits the storage available for queued vehicles on the arterial cross street. The signal timing is very important in controlling the traffic flow for the intersection during the peak hours. If the signal timing is set improperly, the queue from the downstream intersection will spill back and block the upstream approaches (Fang 2004). One of the known effects of the diamond interchange is called demand starvation which occurs when portions of the green at the downstream intersection are not used because conditions prevent vehicles at the upstream intersection from reaching the downstream stop-line (HCM 2000). Recently, a fuzzy logic control for a diamond interchange (FLDI) was developed to improve the coordination between ramp metering and the two signals of intersections (Pham 2013).

Traffic signal control in practice includes pre-timed, actuated and adaptive control. Pre-timed control operates based on fixed intervals and phase timings with fixed number of cycle length and phase sequence. The disadvantage of this type of control is that they become less effective when traffic condition changes substantially. Actuated control which is based on the gap –seek logic to response to traffic fluctuations during the day is more sophisticated than the pre-timed control. The controller extends the green beyond a minimum time until a gap in the traffic flow on the approaches currently with green has been detected or until the pre-defined maximum green time has been reached. Actuated signal control has certain limitations including tendency to extend green inefficiently under low traffic flow conditions, and great sensitivity to incorrectly set maximum green times (Fang 2004).Adaptive control can be defined as any signal control strategy that can adjust signal operations in response to fluctuating traffic demand in real time according to certain criteria (Lin and Vijayayumar 1988). Adaptive control makes signal timing decisions based

on detected or identified current and short-term or even long-term future flow. Since Miller's pioneer work (Miller 1963), considerable research has been done to develop adaptive systems. Systems such as SCOOT (Hunt 1982), SCATS (Luk 1984, Charles 2001), OPAC (Gartner 1983, Gartner and Pooran 2002) and RHODES (Mirchandani and Head 2001) are among the best known. The dynamic programing (DP) algorithm (Fang and L. Elefteriadou 2004) was only devoted to the adaptive signal operation of a diamond interchange. Apart from surface street traffic management, ramp-metering has become a popular method (Muhurdarevic et al. 2006) to reduce congestion on the motorway and to allow the street traffic to join the motorway more safely. A field trial in New Zealand in 2005 showed that motorway traffic throughput (network wide) and travel speed increased significantly within the vicinity of the ramp-metering site (Brown et al. 2005). At an interchange, however, the intersection signals and the ramp-metering signals are mainly separately controlled. An integrated algorithm for both the signals was developed for diamond interchange and simulated in VISSIM. (Tian et al. 2002, Tian 2007).

Human decision making and reasoning in general and in traffic and transportation in particular, are characterized by a generally good performance. Even if the decision-makers have incomplete information, and key decision attributes are imprecisely or ambiguously specified, or not specified at all, and the decision-making objectives are unclear, the efficiency of human decision making is unprecedented. The most advanced central management systems rarely outperform the traffic operators. Although the operators control network traffic mainly based on their expert knowledge and experience, they seem to be able to efficiently control traffic, using qualitative and vague information about the traffic conditions. Although a driver can only very roughly estimate the gap to the preceding vehicle, he is still able to maintain a sufficient distance in most circumstances. Even though humans may be slow, sub-optimal, and inconsistent in their decision-making, their ability to reason using vague variables and their ability to reconcile different conflicting objectives is quite remarkable.. Fuzzy logics are approaches that are much closer to real human observation, reasoning and decision making than other (traditional) approaches, such as ALINEA, Green-wave algorithms. These fuzzy approaches have been applied successfully in a wide range of industrial processes (cement kilns, incineration processes, waste water treatment) and products (e.g. cameras, washing machines). Applications in the field of traffic engineering have only recently emerged in larger numbers, and in many

cases seem quite promising. Most of these applications are experimental and are often rudimentary. Real-life applications of fuzzy sets and fuzzy logic in the field of traffic engineering are rare. There have been studies on the fuzzy ramp-metering and fuzzy logic signal control for an intersection. Bogenberger et al. (2000) described a fuzzy traffic responsive ramp-metering that can automatically adjust the ramp-metering rate based on the current traffic conditions, detected by loop detectors. Pappis and Mamdani (1977) presented a fuzzy logic control (FLC) for a traffic junction of two one-way streets. The input variables are the passed time of the current interval, the number of vehicles crossing an intersection during the green phase, and the length of queuing from the red direction. The extension time inferred by the FLC is the output. Hoyer and Jumar (1994) developed a FLC that controls an intersection involving 12 main directional traffic flows using 10 inputs and 2 outputs. The inputs are traffic density of different lanes and elapsed time since the last state change. The outputs are the extension time of the current phase and the selection of the next phase.

Adaptive signal control based on fuzzy logic control (FLC) determines the duration and the sequence that traffic signal should stay in a certain state, before switching to the next state (Trabia et al. 1999, Pham 2013). The number of arriving and waiting vehicles within the interchange are quantised into fuzzy variables and fuzzy rules are used to determine if the duration of the current state should be extended. The fuzzy logic controller has been shown to be more flexible than fixed controllers and vehicle actuated controllers, allowing traffic to flow more smoothly.

The FLC does not possess the ability to handle various uncertainties especially in real world traffic control. Therefore it is not suitable for stochastic problems such as traffic signal timing optimization. Probabilistic logic is better equipped to handle the uncertainties containing both stochastic and fuzzy features (Pappis and Mamdani 1977). For a typical traffic light control system, the traffic is stochastic in nature. The variability of traffic demand is inherent to the traffic system. This is due to the random nature of arrival at an intersection which also induces stochastic variability into the degree of saturation. There have been research works (Colubi et al. 2003, Liang and Song 1996) on the relationship of randomness and fuzziness. Meghdadi and Akbarzadeh-T (1996) investigated the integration of probability theory and fuzzy logic. Recently, Liu and Li (2005) developed a systematic

framework and methodology of probabilistic fuzzy logic system for robotic system and this methodology can be applied to various fields such as signal processing, power system, networking etc.

In this paper a probabilistic fuzzy logic control is developed for the control of the traffic signal system at a typical diamond interchange incorporating ramp metering. The signal phase and green time extension are decided in response to the traffic conditions in real time. The traffic signal in PFLDI operates under fuzzy logic and the stochastic uncertainties in the traffic conditions, such as the number of vehicles arriving and queuing at an intersection, are described by probabilistic fuzzy sets. To validate the developed PFLC, the traffic system is modeled using Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network version 6 (AIMSUN 6). The stochastic traffic conditions (vehicles arrival and queue) are simulated and the performance of the traffic system under PFLDI is evaluated. The PFLDI is also compared with actuated and conventional fuzzy logic controllers for intersection signals in terms of the average delay of vehicles and the average flow rate. The results show the intersection using PFLC for signal results in an improved performance.

The Diamond Interchange

A. The Interchange and Detectors Placement

The diamond interchange at the Upper Harbor Interchange in Auckland (Figure 1) was chosen to investigate the performance of PFLDI. The distance from north to south of the diamond interchange is 3 kilometers and the distance between the two intersections is 120 meters. Its geometric layout was obtained from a high-resolution JPEG image from Google Earth and its additional information on the widths and number of lanes were supplied by Transit NZ. The interchange is equipped with various detectors for its traffic actuated signal control. The detectors for the intersection signals are placed at 1 m before the stop line and 40 to 60 m away from the stop line for each movement (Figure 2). The detectors for ramp-metering are set up as follows: Ramp Queue detector at 70 m from on-ramp entrance, Ramp Check-In detector at 413 m from on-ramp entrance; Up-stream detector at 160 m from on-ramp exit on the motorway; Down-stream detector at 200 m before on-ramp exit; and Interchange Queue Occupancy detector at 40 m away from the stop-line.

Figure 1: Aerial view of the diamond interchange (Image © 2014 Terra Link International Ltd, © 2014)

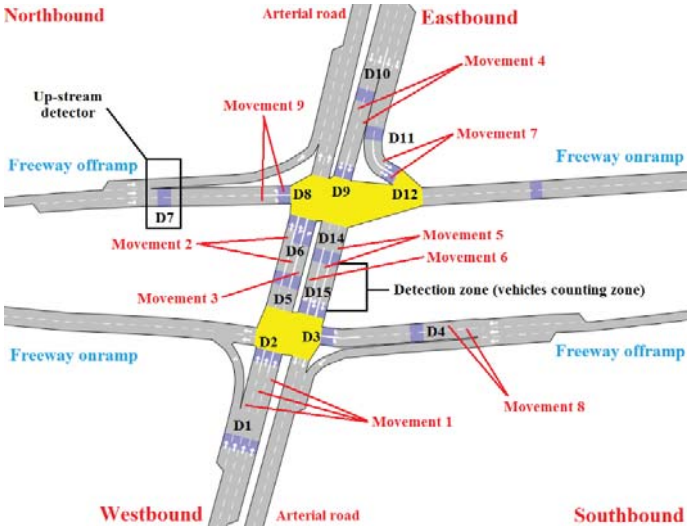


Figure 2: Diamond interchange – two intersections layout in AIMSUN with movements and detectors marked by numbered little blocks

B. The Interchange Signals and Phasing

At the study site (Upper Harbour Diamond Interchange), the existing traffic signal control reacts to the traffic condition by triggering different pre-defined cycle time. Details of this design can be found in Pham et. al. (2013).

The phase section logic (Figure 3) for all the current available phases (A, B, C, D, E, F and G) is defined in Pham et. al. (2013)

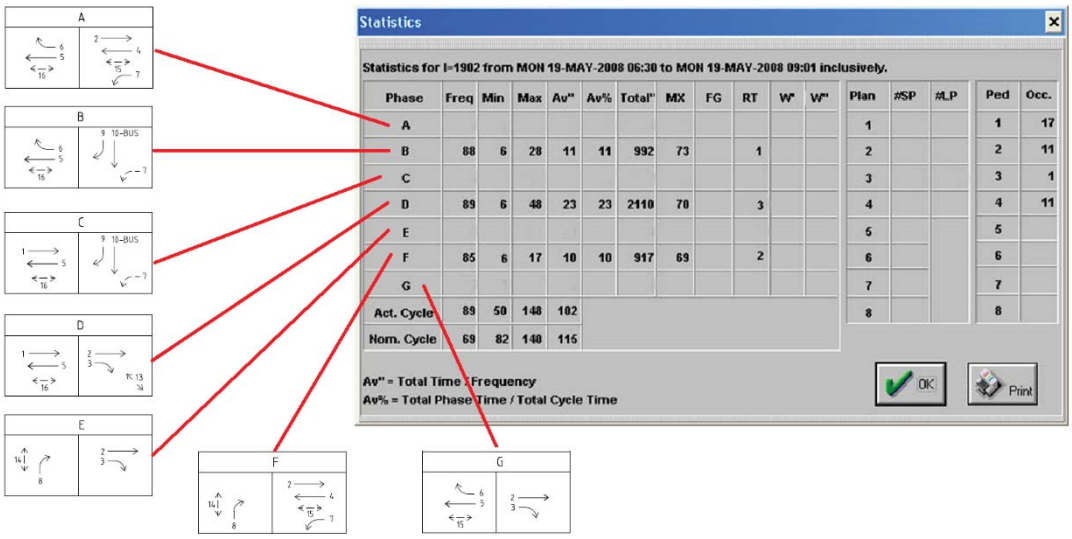


Figure 3: Phasing and signal groups for Upper Harbor diamond interchange

Running from Phase A

Phase A→Phase B (if there are cars waiting on any of movement 9's detectors) **→Phase C** (if there are cars waiting on any of movement 1's detectors) **→Phase D** (unconditional)

Phase A→Phase G (if there are cars waiting on any movement 8's detectors and no car on any of movement 9's detectors) **→Phase D** (unconditional)

Phase A→Phase G (if there are cars on movement 8's detectors and no car on any of movement 1, 2 and 9's detectors) **→Phase D** (unconditional)

Phase G logic

Phase G is a transitional phase providing mid-block clearance when phases are skipped during a signal cycle. This phase effectively provides a flexible early cut-off period for a congested scenario, and a demand for this phase will simultaneously lodge for another phase. Once in Phase G, exit must follow the predefined path. All calls for Phase G are cancelled when the demand conditions are no longer met.

Phasing operation during peak hours

During peak hours from 6:30–9:30 am on weekdays, only three phases (Phase B, F and D) are available and they operate in a fixed sequence, from Phase B to F and to D. Each phase has a green time of 6 s at minimum and, 28, 48 and 17 s at maximum for Phase B, Phase D

and Phase F, respectively. Table 1 gives the signal timing data used for ADI, FLDI and PFDI in this study

Probabilistic Fuzzy Signal

For a typical traffic signal control system, the variables that are considered as stochastic in nature are effective green time, saturation flow rate and degree of saturation (Kamarajugadda and Park 2003).

The traffic demand, conditions are expected to follow a Poisson distribution for a cycle-cycle variability. The Poisson distribution can be visualised as a limiting form of the binomial distribution, and is also used widely in queuing models (Juang 2000). The Poisson model assumes that the number of sources is finite and the traffic arrival pattern is random. Poisson processes have been used widely in traffic modelling. Probabilistic modelling is a good approximation to real world problem when random uncertainty governs the phenomenon.

Fuzzy logic applies the same tools as probability theory. But itis not able to capture the essential property of meaning (partial knowledge) like probability theory. FLC does not possess the ability to handle various uncertainties especially in real world traffic control. Therefore it is not best suited for stochastic nature problems such as traffic signal timing optimization. As a result probabilistic fuzzy logic (PFL) which combines probability theory with Fuzzy logic is the best choice to handle the uncertainties containing both stochastic and fuzzy features. The detailed concept of probabilistic fuzzy set is defined in (Liu and Li 2005).

In the diamond interchange of our study site, the arrival of vehicles from one arterial street follows the Poisson distribution. For saturated traffic flow, the probability, $P(x)$, that x vehicles arrive at any interval in the two-way intersection is calculated by

$$P(x) = \frac{m^x e^{-m}}{x!} = \frac{I}{x!}\left(\frac{Vt}{3600}\right)^x e^{-\frac{Vt}{3600}} \quad (1)$$

Here t is the length of time interval in seconds, V is hourly volume (AIMSUN output for our simulation), and m is the average number of vehicles per interval (Vt/3600).

Design of Probabilistic Fuzzy Logic for a Diamond Interchange

The developed PFLDI model consists of three main modules: Phase Selection Logic (PSL), Probabilistic Fuzzy Phase Timing (PFPT) and Probabilistic Fuzzy Ramp-metering (PFRM), as shown in Figure 4. The framework of the PFLDI algorithm is shown in Figure 5

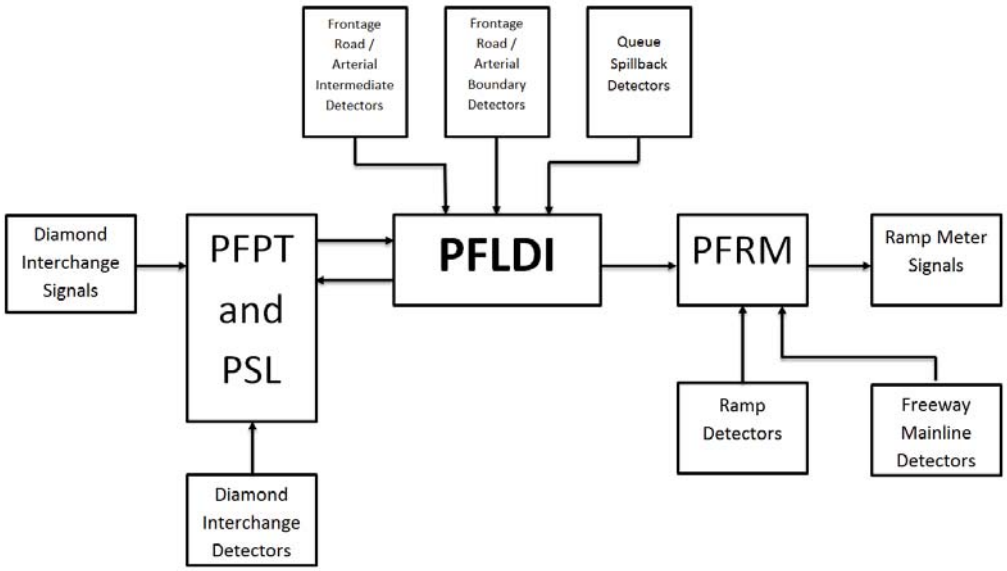


Figure 4: PFLDI model illustrations

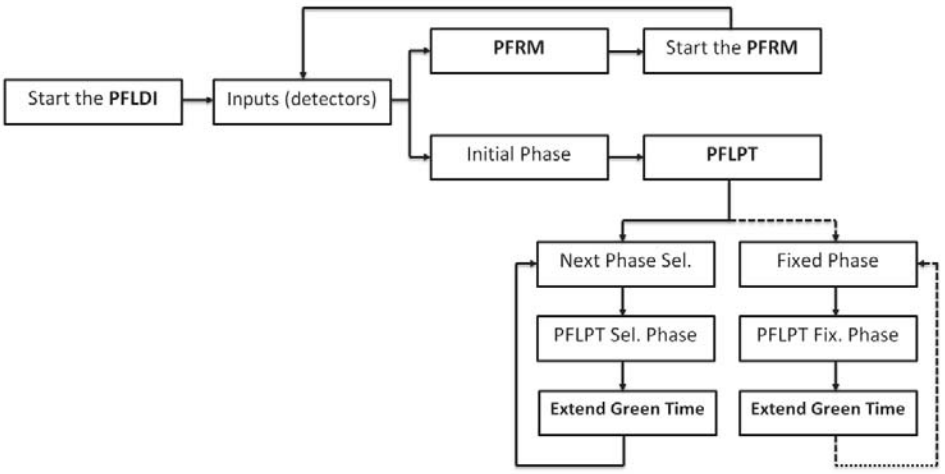


Figure 5: The framework of the PFLDI algorithm

The historical data, traffic plan, and design layout of the specified diamond interchange are based on the information provided by New Zealand Transportation Agency (NZTA) and our site surveys. The detectors can record vehicle's speed, vehicle types, occupancy,

headway, traffic flow, etc. and feed the data into the SCATS and Traffic Management System. Traffic Engineers can extract this information from SCATS at any time for any interval. Fuzzification is a process that converts each numerical (analogue) input into a set of degrees of membership by membership functions. These input data are acquired by numerous detectors on the interchange, ramp and freeway. These detectors were installed and managed by NZTA.

f. The upstream detector is designed to collect information including local speed, local traffic flow and local occupancy of the mainline. In the SH1 Upper Highway Diamond Interchange the detector is placed 140 meters (on the freeway) before the Constellation Drive off-ramp exit.
g. The downstream detector is designed to detect the downstream speed and flow rate (volume). The downstream volume/capacity-ratio is used to measure the bottleneck behaviour. The detector is placed (on the motorway) 210 metres after the South-bound on-ramp exit.
h. The detector at the end of the ramp storage, also called the queue detector, is to detect the queue occupancy. It is placed 180m from South bound on ramp entrance.
i. The check-in detector is located at the ramp metering stop bar to detect the check-in occupancy. The location of the detector is 470 meters from the South-bound on-ramp entrance.
j. Stop-line and advanced detectors are installed on every movement within the interchange.

We are able to analyze and compute the total and average traffic demand for the freeway and arterial roads from the historical data. The membership functions are built based on this analysis. For example: the two mid-block arterial roads, each with a length of 56 meters can hold a maximum of 12 cars on each lane.

A. Probabilistic Fuzzy Ramp-metering (PFRM)

The PFRM module optimizes the ramp-metering rate considering the traffic flows. The input to this module is acquired by four loop detectors installed at different locations, as stated in the preceding section. Membership functions were selected using trial and error method, however the foundation of these functions are based on the Bogenberger's model (Bogenberger and Keller 2000). The Up-stream detector supplies input variables *local*

speed, local traffic flow and local occupancy of the motorway, each of which is described by three fuzzy sets, Low, Medium and High. The downstream detector detects the *downstream speed* and flow rate (volume) for *volume/capacity ratio* (or *v/c ratio*) that is used to measure the bottleneck behavior. The queue detector at the end of the ramp detects the input variable *queue occupancy*. The check-in detector detects the input variable *check-in occupancy*. One fuzzy set is used to describe each *downstream speed, v/c ratio, queue occupancy, check-in occupancy*. The input variables (Table 2) of the PFRM are fuzzy and probabilistic, described by membership functions as shown in Figure 6, where the symbol "*P*" is the probability of a fuzzy linguistic variable. Thus, a variable in the PFRM is characterized by a quintuple,

$$(x, T(x), U, G, M) \tag{2}$$

where x is the name of a variable; $T(x)$ is the term set of x; U is the set of names of linguistic values of x defined; G is the rule for generating the names of values x and M is the semantic rule for associating its meaning with each value. For example, the variable *local speed* is characterized by a quintuple,*("local speed", {(<"low","0.2">, <"medium","0.2">, <"high", "0.2">)}, [0, 100 kilometers per hour)], G, M).*

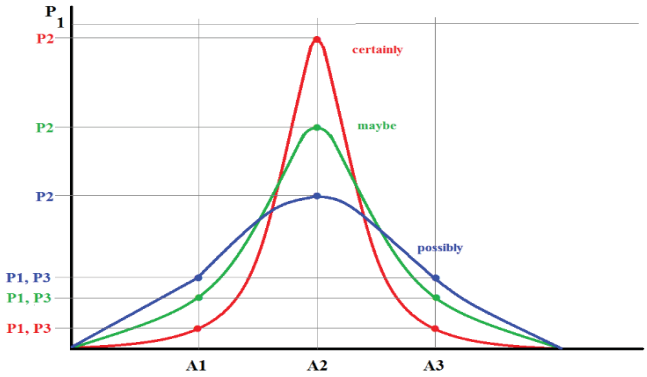


Figure 6: Probabilistic fuzzy membership functions of local speed, flow, occupancy, downstream speed, v/c ratio, check-in and queue occupancy

The membership functions of local speed, flow, occupancy, downstream speed, v/c ratio, check-in and queue occupancy are probabilistic variables with fuzzy rules which can be represented as

$$Tv = \{A_1 / P_1, A_2 / P_{2,}, A_3 / P_3\} \tag{3}$$

where A_1, A_2, A_3 are fuzzy variables (corresponds to Low, Medium and High) with probabilities P_1, P_2, P_3 so that being

$$P_1 + P_2 + P_3 = 1 \tag{4}$$

The output to the PFRM is the *metering rate* that is described by three fuzzy sets, Low, Medium and High from 240 – 900 vehicles/hour, where the metering rate was scaled down by (*metering rate*-240)/(900-240).

B. Probabilistic Fuzzy Phase Timing (PFPT)

The PFPT is designed to calculate the green time extension for each individual phase of the interchange. Vehicular detectors are installed on "Up-stream-line" and "stop-line". The number of approaching vehicles for each approach during any given time interval can be estimated using the detectors data regarding the quantity of traffic on the arrival side (Arrival) and the quantity of traffic on the queuing side (Queue).The vehicle arrivals are calculated based on all the Green signals which include many movements from arterial streets of the interchange. The total vehicle arrival is random. As the signal control is critical during peak hours, this study considers this scenario only. Suppose the PFPT runs Phase B initially with a minimum green time of 6 seconds. The PFPT needs to determine whether to extend or terminate the current green phase when the minimum green time is served. If Phase B is running green, the movements 5, 6, 7, and 9 are considered as the arrival side, whose total number of vehicles is denoted by a variable *Arrival*. The movements 1 to 4, and 8 are queued side, whose total number of vehicles is denoted by a variable Queue. If Phase D is running green then the movements 1 and 5 (since 2 and 3 are linked to 1) are on arrival side and the movements 4, 6, 7, and 8 are queue side. Phase F is running green then the movements 2, 4, 7 and 8 on the arrival side and the movements 1, 5, 6 and 9 on the queue side.

The input variable Arrival (Figure 7) is expressed in four fuzzy sets, Few (over 0~14), Small (over 0~28), Medium (over 14~42) and Many (over 42). The input variable Queue (Figure 8) is expressed in four fuzzy sets, Few (over 0~19), Small (over 0~38), Medium (over 19~38) and Many (over 38~57). The output variable green extension is fuzzified in four fuzzy sets, Very Short (over 0~10 s), Short (over 0~20 s), Medium (over 10~30 s) and Long (over 20~30 s).

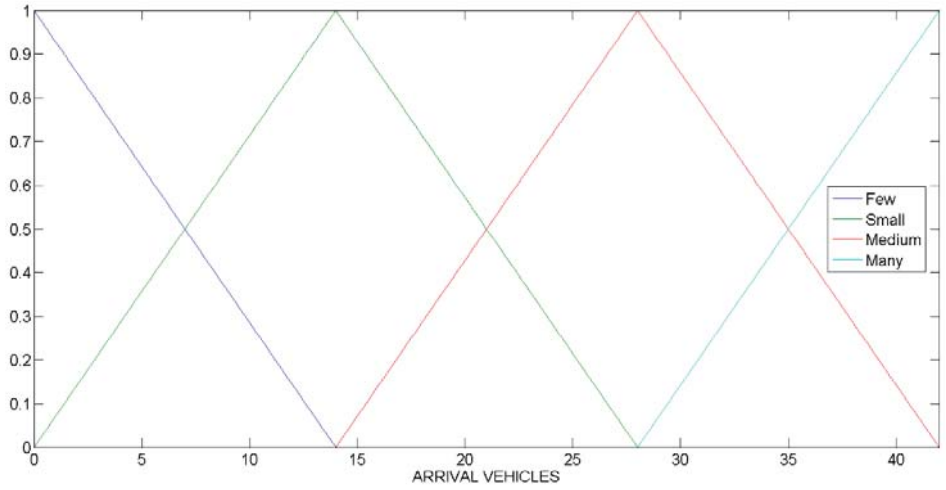


Figure 7: Total Arrival membership functions (Phase B)

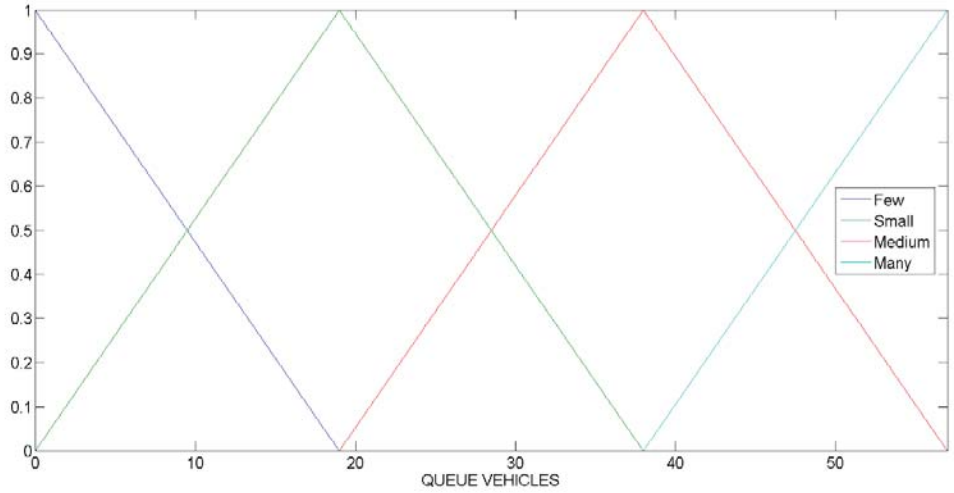


Figure 8: Total Queue membership functions

The input variable Arrival and Queue are converted to probabilistic value using the same process as mentioned in probabilistic fuzzy ramp metering. Green time extension, which is the extension needed for the green light on the arrival side and also the output of the fuzzy variable. In this model, the minimum green time is first given for arriving traffic flows. After that the extension process is stopped and the next phase is selected according to the Phase Selection Logic (PSL). For each phase there is a minimum run of 6 seconds based on Auckland Traffic Operations and Management (ATOM) timing plan (Figure 3). During this process (Figure 9), the PFLC infers the output Green Time Extension for the running phase

which has a range from 6 to 30 seconds (Table 1). Once this is complete, the PFLC moves on to the next phase.

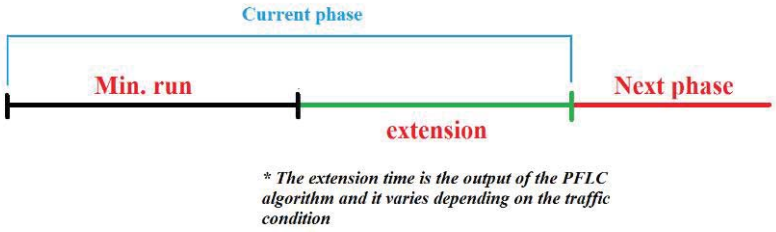


Figure 9: Fuzzy green time extensions in current running phase

C. Rule bases of the PFLDI model

There are two rule bases for the PFLDI. One is with the PFRM module and the other with the PFPT module. They are built on the combination of input and output variables. The configuration of the fuzzy rules in this study is based on the traffic data collected from the Upper Harbour Diamond Interchange site survey. By gathering the data we try our best to replicate the operation of this diamond interchange during the simulation. While the PFLDI algorithm in this study is designed based on the Upper Harbour Diamond Interchange, it can be easily applied to any typical diamond interchange.

The resulting PFRM rule base (Table 3) was formed by 10 rules. The first three rules (Rule 1, 2 and 3) are to ensure that at least one rule will be triggered as all the occupancy range has been covered. Rule 8 is designed to prevent the formation of downstream congestion. Volume/capacity ratio (v/c ratio) is calculated from the historical measured maximum flow rate of downstream and could be seen as a prediction of the downstream bottleneck behavior. Each rule has weight, ranging from 1.5 to 3.0, that sets the priority of each rule (Table 3). The choice of the weights is based on our best understanding of the traffic condition at the study site (survey and historical data). For example: if Downstream Speed is VERY LOW and Downstream v/c is VERY HIGH, then Metering Rate is LOW. The rule weight for this rule is 3.0 (highest priority). Rules 1 and 6 are designed to restrict the metering rate when the vehicles are unable to merge onto the motorway. When the motorway is highly congested, a secondary queue of metered vehicles may form. If a secondary queue persists, ramp-metering is no longer providing any benefit. Rule 9 and 10 are used for the situation when there is a long queue on the ramp.

The PFRM module has to extend the metering rate to the maximum, in order to avoid the long queue on the ramp as well as preventing the traffic spillback to the connected intersection. If this occurs, the metering rate will be set to high allowing more cars to be merged onto the motorway/freeway. The interchange queue occupancy placed in various places at the connected intersections.

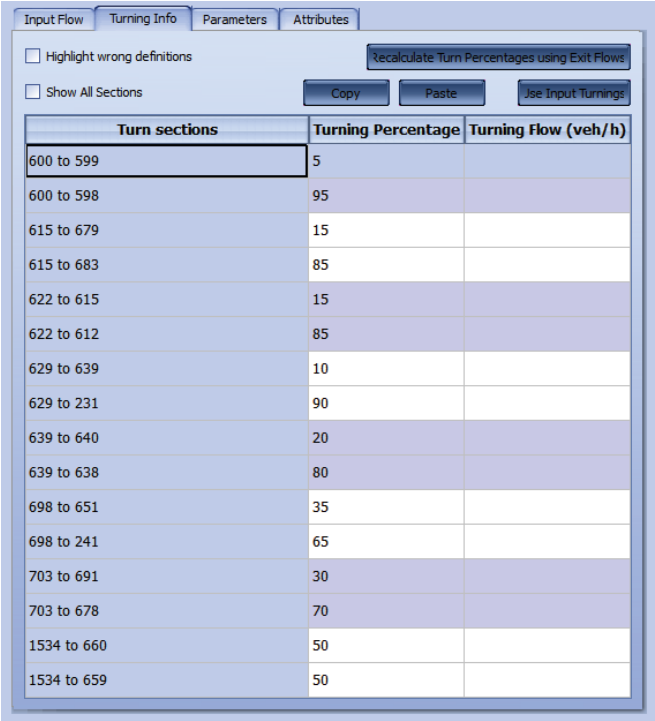
The rule base for PFPT is developed with respect to the three phases B, D and F used during the peak hours. For each phase, there are 16 rules (Table 4).Similar to a conventional fuzzy system, the final output needs to be in a non-fuzzy form (green time extension). This conversion is done by using the continuous centroid formula (Pappis and Mamdani 1977). The defuzzification operation in probabilistic fuzzy logic is associated with probabilistic fuzzy set instead of the normal fuzzy set so that the mathematical expectation of *G* (green time extension) is the crisp output of the PFLC.

$$G = E\left[X_{(G)} \right] \tag{5}$$

where *G* is the crisp output and *E* is the mathematical expectation.

Simulation and sensitivity analysis

A model of the Upper Harbor Diamond Interchange (Figure 1) was built in AIMSUN 6. It was assumed there were no bus, pedestrian crossing, bicycle and motorbike. The simulation conducted was for peak hour period from 8:00-9:00AM on a Monday, during which only Phase B, D and F were running. According to the site surveys and historic traffic data from ATOM and Transit NZ, the traffic flow remains consistent during the weekdays. Only one ramp was monitored, as this is the main traffic flow from suburban areas to the city center during morning peak hours. As the opposite direction remains under free flow for the entire day, monitoring this ramp was not necessary. The order of the phases was fixed, starting with Phase B, then moving to Phase F and ending with Phase D, which repeats until the simulation ends. Only one ramp was metered. The PFLDI was coded in C++ and interfaced to AIMSUN 6 via an API module. Cars and heavy vehicles were simulated where the percentage of the heavy vehicles is around 5-15% of the total traffic demand depending on the movements. The turning traffics within the diamond interchange are configured as Figure 10.



Turn sections	Turning Percentage	Turning Flow (veh/h)
600 to 599	5	
600 to 598	95	
615 to 679	15	
615 to 683	85	
622 to 615	15	
622 to 612	85	
629 to 639	10	
629 to 231	90	
639 to 640	20	
639 to 638	80	
698 to 651	35	
698 to 241	65	
703 to 691	30	
703 to 678	70	
1534 to 660	50	
1534 to 659	50	

Figure 10: Turning traffic configurations

The Upper Harbor diamond interchange was built based on the calibrated S-Paramics model. Traffic flow and lane configurations were provided by NZ Transport Agency. The diamond interchange was calibrated using the flow data on each section. The simulation model is calibrated based on the previous works of Hughes (2000) and Hass (2001). The Upper Harbor diamond interchange is built based on the North Shore City Council's calibrated S-Paramics model. The interface between AIMSUN and SCATS, Paramics is shown in Figure 11.

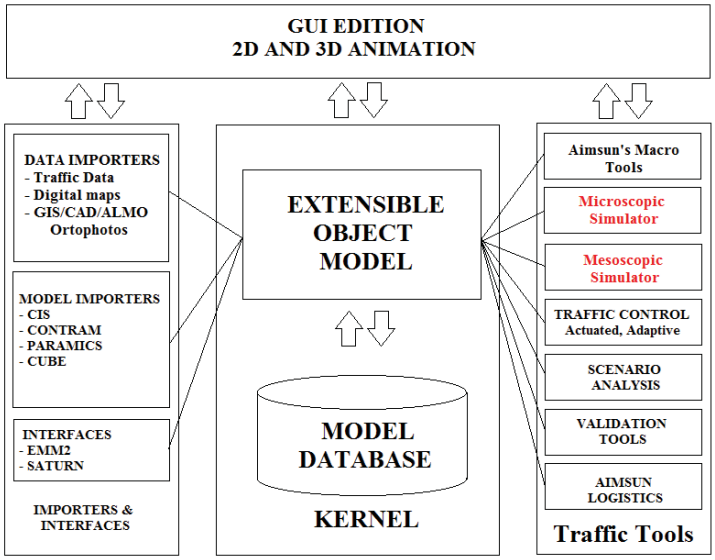


Figure 11: AIMSUN 6 model input interface

Traffic flow and lane configurations were provided by NZ Transport Agency. The model described in this paper was implemented using AIMSUN NG and simulated in AIMSUN version 6.1. The diamond interchange was calibrated using the flow data on each section. Other performance measures, e.g. travel time, travel speeds, and queue length, can be used to calibrate the model (Hughes 2000). Previous studies show that among the different performance measures used to calibrate an AIMSUN model, traffic flow have the most weight followed by travel time, speed and location of traffic jam (Hass 2001).

The operation of traffic actuated signals is highly sensitive to the correct placement of the vehicle detectors. Incorrect placement will degrade the efficiency of traffic flows as well as detrimental effects on the safety of the intersection (Table 5). Incorrect detector locations will add to the lost time per phase and consequently the total lost per cycle of the signal. An increase in lost time will add to the increased individual and total delay and cause the signal to operate at lower level of service. The positions of the detectors for on-ramp and motorways will affect the performance of the Fuzzy Ramp Metering. The current detectors setup is as follow (Figure 12):

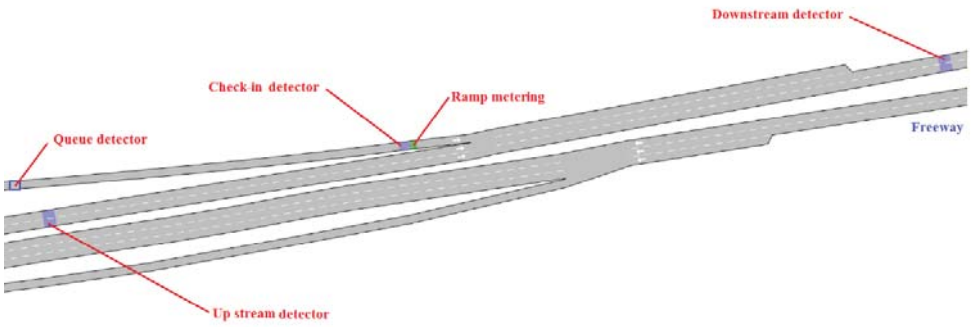


Figure 12: Detectors placement

Detector 1, 2, 5 and 6 are fixed in this analysis as they had been optimized for the Fuzzy Ramp Metering algorithm. In this analysis, the up-stream and down-stream detectors are relocated at different positions (backward and forward from the base position) to investigate the relationship between the detector positions and the performance of the proposed FLDI algorithm. To visualize the effects of the change of detector positions, the positions of detectors that are currently used in the simulation model are considered as base values. System Total Travel Time (STTT) is used as the comparison factor. According to Figure 12, the optimum position for upstream detector is 200m before from the on ramp entrance and for downstream detector is 160m away from on ramp entrance. Detector 3 (Upstream Detector): 160m from South-bound on-ramp exit on State Highway 1 (SH1). Detector 4 (Downstream Detector): 200m before South-bound on-ramp exit on SH1.

The second part of the analysis investigates the change in the system total travel time as the minimum green time changes. The current 6 seconds minimum green time is used as the base for this comparison. There are six minimum green times used in this investigation ranging from 3 to 40 seconds. Minimum green time cannot be higher than 40 seconds because the Fuzzy Phase Timing algorithm is designed to handle a maximum of 57 vehicles on Arrival side and 72 vehicles on the Queue side. From Table 6, the optimum minimum green time is around 6 seconds.

Results analysis

The PFLDI was evaluated by comparing it with the ADI and FLDI models. The ADI model is controlled by AIMSUN with its built in ALINEA algorithm. These values are collected from the various detectors placed on the entire diamond interchange as illustrated in Figure 2 and Figure 12.The performances of the diamond interchange were measured based on the

individual performance of the downstream, ramp, intersections and overall system. The performance of the model is based on the actual performance of the on ramp linked with the freeway (% change in Downstream Average Speed and % change in Downstream Average Delay) and the performance of the two intersections were based on the Delay Time. The System Total Travel Time is the overall performance of the ramp and intersections. The PFLDI's performance is measured by using the following Measures of Effectiveness (MOE) guidelines from (Fang 2004). Travel delay is defined as the time difference between the desired and actual travel time of a vehicle. In this definition, desired travel time is determined (by the simulator) from both the highway speed limit and the driving characteristics of each individual vehicle. Freeway Average Delay is the time per vehicle while travelling on the freeway mainline (minutes/vehicle) within a specified period of analysis.

The Freeway Average Speed is defined as the space mean speed for vehicles serviced by the freeway mainline (kilometres/hour) within a period of analysis. The System Average Delay is the total delay time accumulated by all vehicles (vehicle-kilometres). Lastly, the System Total travel time is the time accumulated by all vehicles (vehicle-hours).

Tests under six scenarios are carried out with Total Traffic Demand ranging from 3348 to 8783 vehicles per hour with varied flow rate. These scenarios have different traffic demand distribution ratios. The varied flow rate allows moderate to heavy traffic congestion. The total traffic demand for the interchange is approximately 3000-4000 veh/h. We purposely set up the traffic demand to be higher than the actual capacity to test the model under the extreme conditions where traffic demand is higher than the capacity. We wanted to see how the model reacted to these scenarios (scenario 4, 5 and 6). The total volume includes the through traffic in the main highway.

According to Table 7, the PFLDI model gives smaller delays than the FLDI and ADI models when the traffic volumes are below 5000 veh/h. However, if traffic volumes are more than 5000 veh/h, the PFLDI gives higher delays than the ADI but performs better than the FLDI. In Upper Harbor diamond interchange the total traffic demand in any given day does not exceed 3800 veh/h. The PFLDI was designed to give higher priority to the traffic on the highways. So when traffic demand is greater than the road capacity, the PFLDI causes the internal congestion at the two inner intersections. This leads to the increase in average delay of the vehicles at the intersections and on-ramp which increases the overall

average delay time of the whole system. Table 8 shows the comparison results of Total Travel Time (TTT) between the three models for six scenarios with Total Traffic Demand ranging from 3348 to 8783 veh/h. When the total traffic demand ranges from 4592 to 7952 veh/h, the TTT generated from the FLDI and PFLDI models are much higher than the ADI model. Similar to the average delay time results, when the traffic demand exceeds the actual road capacity, the ramp metering flushes all the on-ramp vehicles to the highway causing extra delay for the traffic condition. This has to be done in order to prevent a further queue spillback on the internal intersections below (Figure 13). From the results above, PFLDI generates lower TTT compared to the FLDI model.

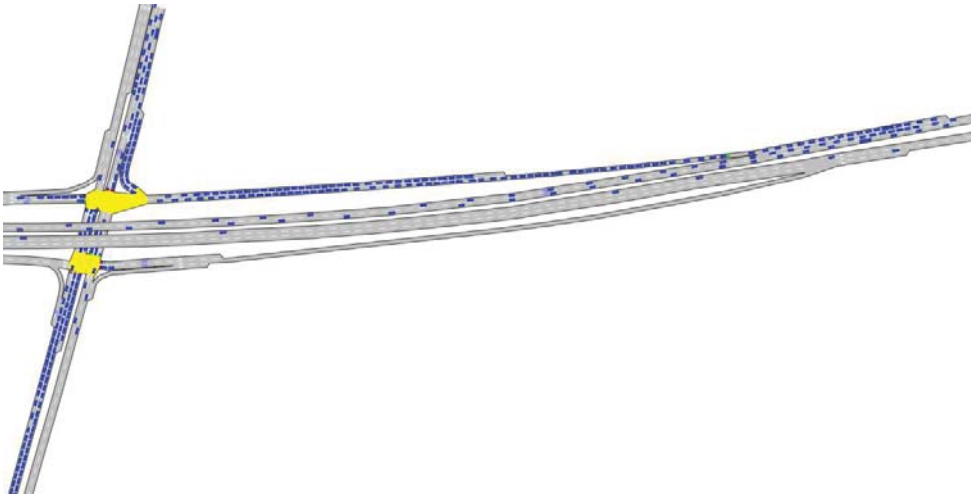


Figure 13: On-ramp queue flush in action

Table 9 and Table 10 show that PFLDI outperforms the FLDI and ADI in terms of downstream average speed and average delay time. This is because the probabilistic fuzzy generates more accurate metering rate based on the current traffic condition, thus creating a smooth flow of traffic into the mainline highway.

Conclusion

Probabilistic Fuzzy Logic Diamond Interchange (PFLDI), Fuzzy Logic Diamond Interchange Metering (FLDI), Actuated Diamond Interchange Control (ADI) were built and tested in AIMSUN 6 environment with six different traffic scenarios ranging from moderate to heavy congestion. From the results, it was found that PFLDI outperforms the ADI and FLDI models in terms of system total travel time, downstream average delay and average speed even when the traffic demand exceeds the road capacity (including arterial

roads, ramp and motorway). The ADI model becomes less effective once the total traffic demand is much higher than road capacity. The PFLDI helps avoid the formation of congestion and in extreme traffic conditions is also able to prevent the congestion from getting worse.

References

Bogenberger, K., and Keller, H. (2000). "An Evolutionary Fuzzy System for Coordinated and Traffic Responsive Ramp-metering." *Annual Hawaiian International Conference on System Sciences*, USA, 1-3.

Brown, T., Clark, I., Evans, P. and Wee, L. (2005). "New Zealand's easy merge ramp signal (ramp-metering) trial." *IPENZ*, 8, 1-14.

Charles, P. (2001). "Time for a Change." *Traffic Technology International*, 68-71, 2001.

Colubi, A., Fernandez-Garcia, C., and Gil, M. A. (2003). "Simulation of random fuzzy variables: an empirical approach to statistical/probabilistic studies with fuzzy experimental data." *IEEE Trans. Fuzzy System*, 10(3), 384-390.

Fang, F. C. (2004). "Optimal Adaptive Signal Control for Diamond Interchange using Dynamic Programming." Ph. D Thesis, College of Engineering, Pennsylvania State University, USA, 4-32.

Fang, F. C., and Elefteriadou, L. (2006). "Development of an Optimization Methodology for Adaptive Traffic Signal Control at Diamond interchange." *ASCE Journal of Transportation Engineering*, vol. 132, no.8, 629-637.

Gartner, N. H. (1983). "OPAC: A Demand-Responsive Strategy for Traffic Signal Control." *Transportation Research Record 906*, Transportation Research Board, Washington, DC,75-81.

Gartner, N. H., and Pooran, F. J. (2002). "Implementation and Field Testing of the OPAC Adaptive Control Strategy in RT-TRACS." *the 81st Annual TRB Meeting*, Washington DC, USA.

Hass, P. C. (2001). “Assessing Developments Using AIMSUN.” *IPENZ Transportation Conference*, Auckland, New Zealand.

Highway Capacity Manual (2000). Transportation Research Board, National Research Council, Washington DC, USA, 12-24.

Hughes, J. (2000). “AIMSUN2 Simulation of a Congested Auckland Freeway.” Master thesis at The University of Auckland, New Zealand, 10-25.

Hoyer, R., and Jumar, U. (1994). "An Advanced Fuzzy Controller for Traffic Lights." *Artificial Intelligence in Real Time Control Symposium*, Valencia, Spain, 3-6.

Hunt, P. B., Robertson, D. I., and Bretherton, R. D., Royle, M. C. (1982). "The SCOOT on-line-traffic-signal optimization technique." *Traffic Engineering and Control*, 23(4), 190-192.

Juang, J. G. (2000). “Neural fuzzy system approaches for robotic gait synthesis.” *IEEE Trans. Syst., Man., Cybern.*, 30(4), 594-601.

Kamarajugadda, A., and Park, B. (2003). “Stochastic Traffic Signal Timing Optimization.” Research Re-port, UVACTS-15-0-44, Centre for Transport Studies, University of Virginia, Virginia, 2-40.

Liang, P., and Song, F. (1996). “What does a probabilistic interpretation of fuzzy sets mean?” *IEEE Trans. Fuzzy System*, 4(2), 200-205.

Luk, J. Y. K. (1984). “Two Traffic Responsive Area Traffic Control Methods: SCATS and SCOOT.” *Traffic Engineering and Control*, 30, 14-20.

Lin, F. B, and Vijayayumar, S. (1988). “Adaptive signal control at isolated intersections.” *ASCE Journal of Transportation Engineering*, 114(5), 555-573.

Liu, Z., and Li, H. X. (2005). “A Probabilistic Fuzzy Logic Control System for Modeling and Control.” *IEEE Trans. Fuzzy System*, 13(6), 848-858.

Meghdadi, A.H., and Akbarzadeh-T, M.-R. (2001). “Probabilistic fuzzy logic and probabilistic fuzzy systems.” *Proc. 10th IEEE Int. Conf,* Melbourne, Australia, 1127-1130.

Miller, A. J. (1963). “A Computer Control System for Traffic Networks.” Birmingham, England.

Mirchandani, P., and Head, L. (2001). “A Real-time Traffic Signal Control System: Architecture, Algorithms and Analysis.” *Transportation Research,* C(9), 414-432.

Muhurdarevic, Z., Condie, H., Ben-Shabat, E., Ernhofer, O., Hadj-SSalem, H., and Papamichail, I. (2006). “Ramp-metering in Europe – state-of-art and beyond” *5th International Conference on Traffic and Transportation Studies* (ICTTS), Xi’an, China, 2006.

Pappis, C. P., and Mamdani, E. H. (1977). “A Fuzzy Logic Controller for a Traffic Junction.” *IEEE Transactions System, Man and Cybernetics,* SMC-7 (10), 707-717.

Pham, V. C., Alam, F., Potgieter, J., Fang, F., and Xu, W. L. (2013). “Integrated fuzzy signal and ramp-metering at a diamond interchange” *J. Adv. Transp.,* 47, 413-434.

Sen, U., and Head, K. L. (1997). “Controlled Optimization of Phases at an Intersection.” *Transportation Science*, 31(1), 5-17.

Trabia, M. B., Kaseko, M. S., and Ande, M. (1999). “A two-stage fuzzy logic controller for traffic signals.” *Transportation Research*, C(7), 353-36.

Tian, Z. Z. (2007). “Modeling and implementation of an integrated ramp-metering-diamond interchange control system.” *Journal of Transportation Systems Engineering and Information Technology,* 7(1), 61-72.

Tian, Z. Z., Balke, K., Engelbrecht, B., and Rilett, L. (2002). “Integrated control strategies for surface street and freeway systems.” *Transportation Research Record 1811,* 92-99.

List of figure captions

Table 1 Signal timing data for ADI, PFLDI and FLDI

Actuated Control			
Phase	Min. Green time (s)	Max. Green time (s)	Amber (s)
B	6	28	4
D	6	48	4
F	6	17	4
PFLDI and FLDI			
B	6	6-30	4
D	6	6-48	4
F	6	6-18	4

Table 2 Terms of the fuzzy sets for inputs and outputs for PFRM module

Fuzzy sets for inputs and outputs	**Low**	**Medium**	**High**
Local Speed (0-100 km/h)	Low	Medium	High
Local Flow Rate (0-4000 veh/h)	Low	Medium	High
Metering Rate (240-900 veh/h)	Low	Medium	High
Local Occupancy (0-30%)		High	
Downstream v/c (0-1)		High	
Downstream Speed (0-100 km/h)		High	
Check-in Occupancy (0-50%)		High	
Queue Occupancy (0-50%)		High	
Interchange Queue Occupancy (0-90%)		High	

Table 3 Rules and its weighting for PFRM module

RULE	IF	AND	THEN	WEIGHT
1	Local Occupancy = Low		Metering Rate = High	1.5
2	Local Occupancy = Medium		Metering Rate = Medium	1.5
3	Local Occupancy = High		Metering Rate = Low	2.0
4	Local Speed = Low	Local Flow Rate = High	Metering Rate = Low	2.0
5	Local Speed = Medium	Local Occupancy = High	Metering Rate = Medium	1.0
6	Local Speed = Medium	Local Occupancy = Low	Metering Rate = High	1.0
7	Local Speed = High	Local Flow Rate = High	Metering Rate = High	1.0
8	Downstream speed = Very Low	Downstream v/c = Very High	Metering Rate = Low	3.0
9	Check-in Occupancy = Very High	Queue Occupancy = Very High	Metering Rate = High	3.0
10	Interchange Queue Occupancy = High	Queue Occupancy = High	Metering Rate = High	3.0

Table 4 Rules and its weighting for PFPT module

RULE	IF ARRIVAL	QUEUE	THEN EXTENSION	WEIGHT
1	Few	Few	Very Short	1
2	Few	Small	Very Short	1
3	Few	Medium	Very Short	1
4	Few	Many	Very Short	1
5	Small	Few	Short	2
6	Small	Small	Short	2
7	Small	Medium	Very Short	1
8	Small	Many	Very Short	1
9	Medium	Few	Medium	3
10	Medium	Small	Medium	3
11	Medium	Medium	Short	2
12	Medium	Many	Short	2
13	Many	Few	Long	4
14	Many	Small	Medium	3
15	Many	Medium	Medium	3
16	Few	Many	Short	2

Table 5 Percentage of change in STTT vs. Positions of upstream/downstream detectors

Position of detector from ramp entrance (m)	System Total Travel Time (hours)	Percentage Change of System Total Travel Time	System Total Travel Time (hours)	Percentage of change in System Total Travel Time
	Upstream	*Upstream*	*Downstream*	*Downstream*
40	286.87	3.500%	276.48	-0.249%
60	284.09	2.497%	275.79	-0.498%
80	286.32	3.301%	287.43	3.702%
100	284.51	2.648%	284.71	2.720%
120	282.74	2.010%	285.48	2.998%
140	279.13	0.707%	284.93	2.800%
160	277.17	0.000%	279.39	0.801%
180	277.32	0.054%	278.55	0.498%
200	278.12	0.343%	277.17	0.000%
220	279.87	0.974%	278.67	0.541%
240	278.32	0.415%	281.23	1.465%
260	282.18	1.808%	280.21	1.097%

Table 6 Minimum Green Time vs. Percentage of change in STTT

Minimum Green Time (seconds)	Detection Interval (seconds)	STTT (hours)	Percentage change in STTT (%)
3	3	35.69	0.17
6	6	35.63	0.00
12	12	35.86	0.65
18	18	35.78	0.42
24	24	36.07	1.23
30	30	36.19	1.57
36	36	36.30	1.88

Table 7 Average delay time comparison (second per vehicle)

Scenario	Traffic demand	ADI Model	FLDI Model	PFLDI Model
1	3348	21.48	5.32	4.18
2	3752	23.92	7.71	6.50
3	4592	41.62	17.39	15.34
4	6272	71.03	135.27	121.15
5	7952	88.94	149.86	137.65
6	8783	135.62	152.79	151.49

Table 8 Total travel time comparison (hour)

Scenario	Traffic demand	ADI Model	FLDI Model	PFLDI Model
1	3348	42.78	36.74	35.82
2	3752	58.69	62.96	48.52
3	4592	82.58	122.66	68.69
4	6272	135.45	210.17	201.23
5	7952	183.23	263.88	248.66
6	8783	285.86	276.43	265.82

Table 9 Downstream average speed comparison (km/hr)

Scenario	Traffic demand	ADI Model	FLDI Model	PFLDI Model
1	3348	86.82	88.33	91.36
2	3752	87.14	84.65	89.43
3	4592	84.78	82.19	86.12
4	6272	78.03	79.57	80.72
5	7952	73.95	78.12	78.82
6	8783	41.74	76.18	77.23

Table 10 Downstream average delay (second per vehicle)

Scenario	Traffic demand	ADI Model	FLDI Model	PFLDI Model
1	3348	0.46	0.45	0.43
2	3752	0.62	0.76	0.68
3	4592	0.81	1.15	0.92
4	6272	1.88	1.69	1.36
5	7952	2.99	1.82	1.62
6	8783	21.12	2.23	1.58

JOURNAL OF ADVANCED TRANSPORTATION
J. Adv. Transp. (2011)
Published online in Wiley Online Library (wileyonlinelibrary.com).
DOI: 10.1002/atr.167

Integrated fuzzy signal and ramp-metering at a diamond interchange

C. V. Pham[1], J. Potgieter, [1] F. Alam[1], F. C. Fang[2] and W. L. Xu[3]

[1] Massey University, School of Engineering and Advanced Technology, Gate 4, Building 106, Albany Highway, Albany, Auckland, New Zealand

[2] Department of Civil Engineering, College of Engineering, Technology, and Architecture, University of Hartford, 200 Bloomfield Avenue, West Hartford, Connecticut 06117, U.S.A

[3] Department of Mechanical Engineering, The University of Auckland, Private bag 92019, Auckland, New Zealand

SUMMARY

We propose a fuzzy logic control for the integrated signal operation of a diamond interchange and its ramp meter, to improve traffic flows on surface streets and motorway. This fuzzy logic diamond interchange (FLDI) comprises of three modules: fuzzy phase timing (FPT) module that controls the green time extension of the current phase, phase logic selection (PLS) module that decides the next phase based on the pre-defined phase sequence or phase logic and, fuzzy ramp-metering (FRM) module that determines the cycle time of the ramp meter based on current traffic volumes and conditions of the surface streets and the motorways. The FLDI is implemented in Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network version 6 (AIMSUN 6), and compared with the traffic actuated signal control. Simulation results show that the FLDI outperforms the traffic actuated models with lower System Total Travel Time, Average Delay and improvements in Downstream Average Speed and Average Delay.

1. INTRODUCTION

There are many types of interchanges; the most common types is the diamond interchange which is made of two intersections, each connecting to a freeway/motorway direction. Most of these intersections are **signalised** when the demand flows are high (peak hours). One operational issue existing today is that the diamond interchange and ramp-metering are treated as independent elements, due to jurisdictional responsibilities where the surface street arterial is managed by city or county agencies while the freeway/motorway and ramp-metering system is managed by the state department of transportation. The lack of system integration or coordination between the diamond signals and ramp metering often creates major operational concerns, among which queue spillback from the metered ramp is the most obvious [1].

Fig. 1 Queue spillback at a diamond interchange with ramp metering

The **signalised** intersections connecting to the arterial cross street are often the key operational element within the interchange system. The distance between the two intersections varies from less than 120 meters in densely developed urban areas to 240 meters or more in suburban areas [2]. The short distance between the two intersections limits the storage available for queued vehicles on the arterial cross street. The signal timing is very important in controlling the traffic flow during the peak hours, and if the signal timing is not set properly, the queue from the downstream intersection would spillback and blocks the Up-stream approaches [1]. HCM [3] points out an effect known as demand starvation which occurs when portions of the green at the downstream intersection

are not used because conditions prevent vehicles at the Up-stream intersection from reaching the downstream stop-line.

There are two types of traffic controllers that are widely in use. The fixed-mode (pre-timed) traffic controller operates based on pre-defined cycle time for various periods during the day. The actuated traffic controller responds to the traffic condition based on current traffic conditions. The inputs to the actuated controller are collected from various detectors placed along the road. They react to the current traffic condition by triggering different pre-defined cycle time. The fixed cycle time is calculated based on the historic traffic data. The disadvantage with these approaches is that they become inefficient in response to the current traffic condition when there is a sudden change in the traffic flow. Moreover, the cycle time needs to be updated regularly to adapt to the changes of the traffic.

Adaptive control makes signal timing decisions based on detected or identified current and short-term or even long-term future flow. Since Miller's pioneer work [4], considerable research has been done to develop adaptive systems. Systems such as SCOOT [5], SCATS [6,7], OPAC [8,9] and RHODES [10,11] are among the best known. The DP (dynamic programming) algorithm [12] was only devoted to the adaptive signal operation of a diamond interchange.

Beside the surface street traffic management, ramp-metering has become popular [13] to reduce congestion on the motorway and allow street traffic to join the motorway more safely. A field trial in New Zealand in 2005 showed that motorway traffic throughput (network wide) and travel speed were significantly increased within the vicinity of the ramp-metering site [14]. At an interchange, however, the intersection signals and the ramp-metering signals are mainly separately controlled. An integrated algorithm for both the signals was proposed for diamond interchange and simulated in VISSIM. [15, 16]

There also have been studies on the fuzzy ramp-metering and fuzzy logic signal control for an intersection. Bogenberger et al. [17] described a fuzzy traffic responsive ramp-metering that can automatically adjust the ramp-metering rate based on the current traffic conditions, detected by loop detectors. Pappis and Mamdani [18] presented a fuzzy logic control (FLC) for a traffic junction of two one-way streets. The input variables are the passed time of the current interval, the number of vehicles crossing an intersection during the green phase, and the length of queuing from the red direction. The extension time inferred by the FLC is the output. Hoyer and Jumar [19] proposed a FLC that controls an

intersection involving 12 main directional traffic flows using 10 inputs and 2 outputs. The inputs are traffic density of different lanes and elapsed time since the last state change. The outputs are the extension time of the current phase and the selection of the next phase. However, the FLC cannot choose phase freely and, thus produces more delay time when traffic densities of each approach are highly uneven. Trabia et al. [20] discussed a FLC for an isolated 4-approach intersection with through and left turning movements. The controller, based on the estimation/ measurement of approach flows and approach queues, terminates the current signal phase or extends it. Favilla et al. [22] proposed an FLC with adaptive strategies for fuzzy urban traffic systems. The FLC adjusts the membership functions according to the traffic conditions to optimize the controller's performance. Jamshidi et al. [23] developed a simulator for fuzzy control of traffic systems. They chose three fuzzy input variables - the average density of traffic during the green signal periods, the average density of traffic during the red periods, and the length of the current cycle time. Fuzzy output decides whether to change the state of the light or remain in the same state. Nakatsuyama et. al. [24] in extending the application of fuzzy logic to two consecutive junctions and work done by Chih-Hsun et. al. [25] in applying fuzzy traffic controllers to multiple junctions. Wannige et al. [26] developed an adaptive neuro-fuzzy traffic signals control for two 4-way traffic junctions. The developed neuro-fuzzy system automatically draws membership functions and the rules by itself, thus making the designing process easier and more reliable compared to conventional fuzzy logic controllers. The traffic inflows of roads are given as inputs to the fuzzy control system which generate the corresponding green light time as the output to control the signal timing. The control systems try to minimize the delay experienced by the drivers at the two traffic junctions.

Queue spillback resulting from the lack of coordination between the ramp meter and diamond interchange creates serious operational concerns on the diamond interchange and surface arterial. Although queue override policies currently being used at ramp meters can eliminate queue spillback, frequent queue flushes can lead to freeway breakdown and diminish the main purpose of ramp-metering [1].

So far there has been no fuzzy logic control for a diamond interchange integrated with ramp-metering. Common three-phase and four-phase plans are based on off-line demand and do not respond to demand fluctuation in real time. Therefore, there is a need to develop an adaptive signal control for the entire diamond interchange including the on-ramp-metering. In this study we intend to integrate a fuzzy logic approach to control the entire traffic flow at a diamond interchange with least modifications to the current infrastructure. The developed model is designed to minimize queue spillback occurrences at the ramp meter while maintaining efficient operations for the system. The model (or FLDI) comprises of three main modules: fuzzy phase timing (FPT) module, phase logic selection (PLS) module and fuzzy ramp-metering (FRM) module. The model is implemented on an actual interchange and simulated in AIMSUN 6, and the results are compared against those of a conventional traffic actuated controller. The performance criteria used are the flow/volume/demand of the traffic and the travel time.

2. THE DIAMOND INTERCHANGE

A. The Interchange and Detectors

The diamond interchange under concern is Upper Harbor Interchange in Auckland (Fig. 2). The distance from north to south of the diamond interchange is 3 kilometers and the distance between the two intersections is 120 meters. Its geometric layout was obtained from a high-resolution JPEG image from Google Earth and its additional information on the widths and number of lanes were supplied by Transit NZ.

The interchange was instrumented with various detectors for its traffic actuated signal control. The detectors for the intersection signals were placed at 1 m before the stop line and 40 to 60 m away from the stop line for each movement (Fig. 3).

The detectors for ramp-metering were set up as follows: Ramp Queue detector at 70 m from on-ramp entrance, Ramp Check-In detector at 413 m from on-ramp entrance; Up-stream detector at 160 m from on-ramp exit on the motorway; Down-stream detector at 200 m before on-ramp exit; and Interchange Queue Occupancy detector at 40 m away from the stop-line.

B The Interchange Signal

In this study site (Upper Harbor Interchange), the current traffic signal control reacts to the traffic condition by triggering different pre-defined cycle time. This interchange signal is vehicle-actuated and has special phase selection logic (Fig. 5), which was specially designed and optimized by Transit NZ. The phasing logic is stated below. This interchange signal is based on Sydney Coordinated Adaptive Traffic System (SCATS). SCATS is a real time adaptive traffic system that uses stop-line vehicle detectors to detect the changes in traffic demand and adapts the signals accordingly. SCATS has been developed by Roads and Traffic Authority (RTA) of New South Wales, Australia. According to Sissons [21], Nineteen Local Authorities in NZ use SCATS and the majority of all the traffic signals in NZ are under SCATS control.

1) Running from Phase A

Phase A → Phase B (if there are cars waiting on any of movement 9's detectors) → **Phase C** (if there are cars waiting on any of movement 1's detectors) → **Phase D** (unconditional)

Phase A → Phase G (if there are cars waiting on any movement 8's detectors and no car on any of movement 9's detectors) → **Phase D** (unconditional)

Phase A → Phase G (if there are cars on movement 8's detectors and no car on any of movement 1, 2 and 9's detectors) → **Phase D** (unconditional)

2) Running from Phase B

Phase B → Phase A (if there are cars on movement 4's detectors and no car on movement 1, 2 and 8's detectors)

Phase B → Phase G (if there are cars on movement 2's detectors and no cars on movement 1's detectors) → **Phase D** (unconditional)

Phase B → Phase G (if there are cars on movement 8's detectors and no car on movement 1 and 2's detectors) → **Phase E** (unconditional)

3) Running from Phase C

Phase C → Phase D (unconditional)

4) Running from Phase D

Phase D → Phase G (if there are cars on movement 4, 6 and 7's detectors and no car on movement 8's detectors) → **Phase A**

Phase D → Phase E (if there are cars on movement 8's detectors)

5) Running from Phase E

Phase E → Phase F (if there are cars on movement 4 and 7's detectors) → **Phase A** (unconditional)

Phase E → Phase G (if there are cars on movement 6's detectors, and no car on movement 4 and 7's detectors) → **Phase A** (unconditional)

Phase E → Phase G (if there are cars on movement 9's detectors, and no car on movement 4, 6 and 7's detectors) → **Phase B**

Phase E → Phase D (if there are cars on movement 1's detectors and no car on movement 4, 6 and 7's detectors)

Fig. 2 Upper Harbor diamond interchange in AIMSUN 6

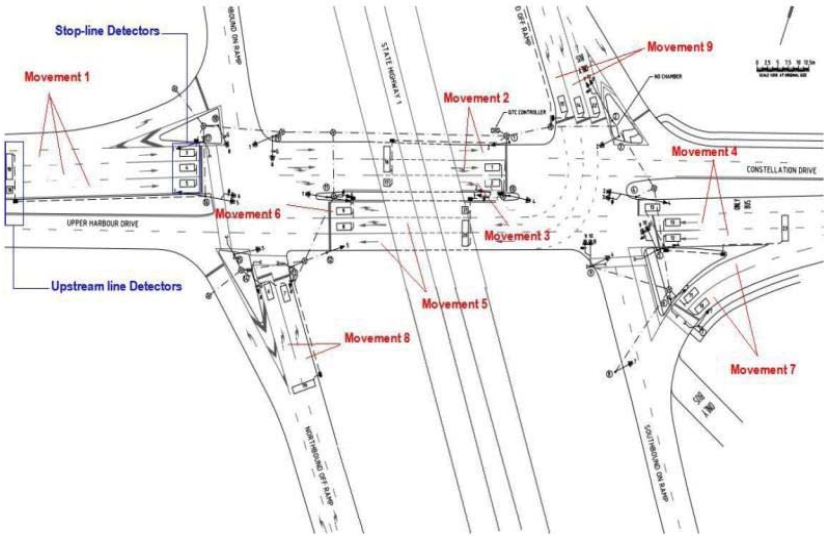


Fig. 3 Movements and detectors marked by numbered little blocks

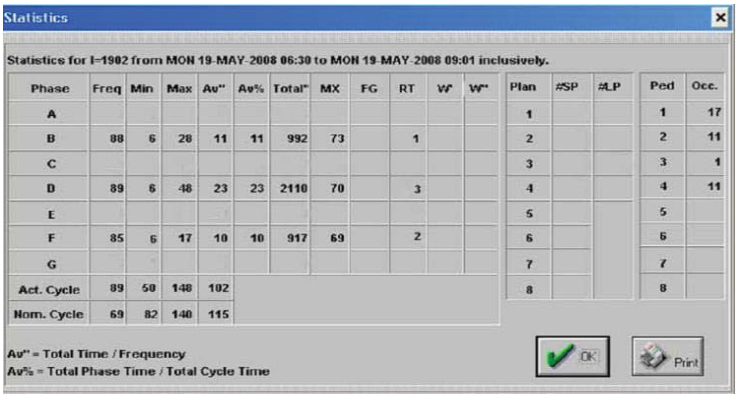


Fig. 4 IDM report from SCATS for Upper Harbor diamond interchange

6) Phase G logic

Phase G is a transitional phase providing mid block clearance when phases are skipped during a signal cycle. This phase effectively provides a flexible early cut-off period for a congested scenario, and a demand for this phase will simultaneously lodge for another phase (Fig. 6). Once in Phase G, exit must follow the predefined path. All calls for Phase G are cancelled when the demand conditions are no longer met.

7) Phasing operation during peak hours

During peak hours from 6:30–9:30 am on weekdays, only three phases (Phase B, F and D) are available that run in a fixed sequence, from Phase B to F and to D. Each phase has a green time of 6 s at minimum and, 28, 48 and 17 s at maximum for Phase B, Phase D and Phase F, respectively.

Table 1 Signal Timing Data for FLDI, ADIFM and ADINM

Signal Timing for ADIFM and ADINM (Upper Harbor diamond interchange)

PHASE	MINIMUM GREEN TIME (sec)	MAXIMUM GREEN TIME (sec)	AMBIENT (sec)
B	6	28	4
D	6	48	4
F	6	17	4

Signal Timing for FLDI (Upper Harbor diamond interchange)

PHASE	MINIMUM GREEN TIME (sec)	FUZZY GREEN TIME EXTENSION (sec)	AMBIENT (sec)
B	6	0-30	4
D	6	0-48	4
F	6	0-18	4

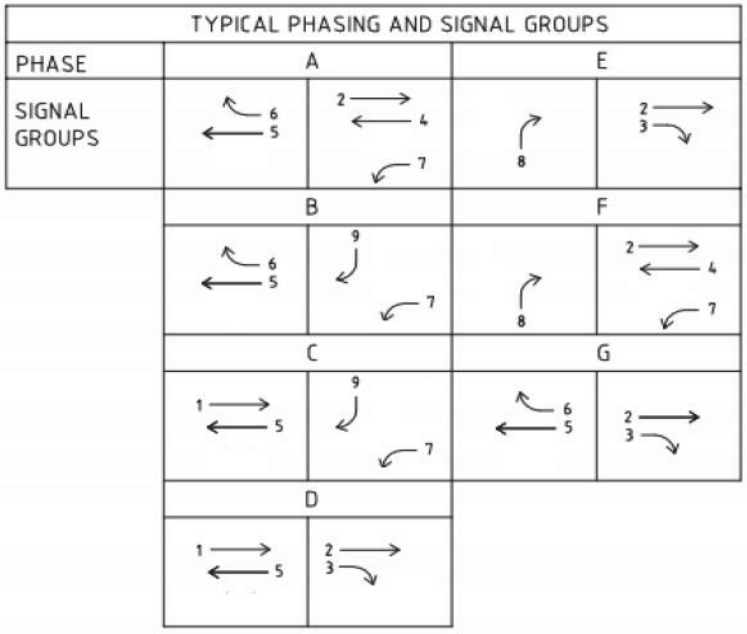


Fig. 5 Upper Harbor diamond interchange phasing and signal groups

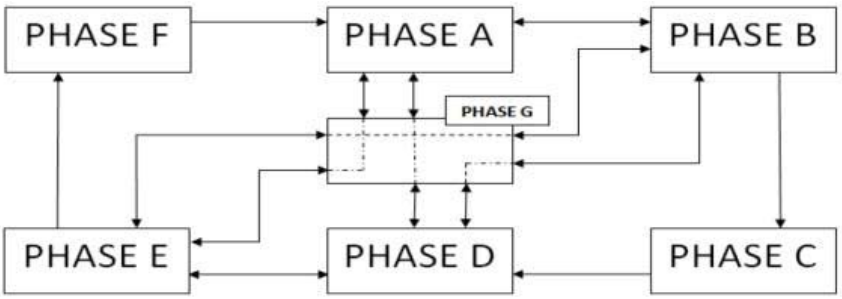


Fig. 6 Phase selection logic

3. DESIGN OF THE FUZZY SIGNALS

The proposed FLDI model is made of three main modules: Phase Selection Logic (PSL), Fuzzy Phase Timing (FPT) and Fuzzy Ramp-metering (FRM), which are tied together in the way as shown in Fig. 7.

A. Fuzzy Ramp-metering (FRM)

The FRM module optimizes the ramp-metering rate considering traffic flows. The input to this module is acquired by four loop detectors installed at different locations, as stated in preceding section. Membership functions were selected using trials and errors method, however the foundation of these functions are based on the Klaus Bogenberger's model [17]

The Up-stream detector supplies input variables *local speed, local traffic flow and local occupancy* of the motorway, each of which is described by three fuzzy sets, Low, Medium and High.

The downstream detector detects the *downstream speed* and flow rate (volume) for *volume/capacity ratio* (or *v/c ratio*) that is used to measure the bottleneck behavior. The queue detector at the end of the ramp detects the input variable *queue occupancy*. The check-in detector detects the input variable *check-in occupancy*. One fuzzy set is used to describe each *downstream speed, v/c ratio, queue occupancy, check-in occupancy*.

The output to the FRM is the *metering rate* that is described by three fuzzy sets, Low, Medium and High (Fig. 8) from 240 – 900 vehicles/hour, where the metering rate was scaled down by (*metering rate-240)/(900-240)*.

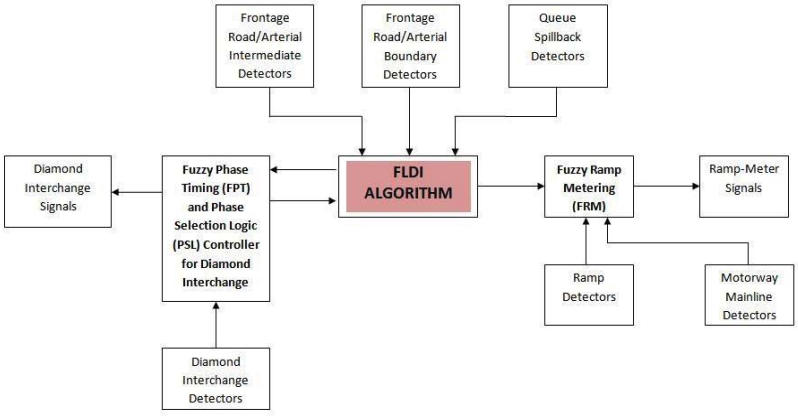


Fig. 7 Framework of FLDI Algorithm

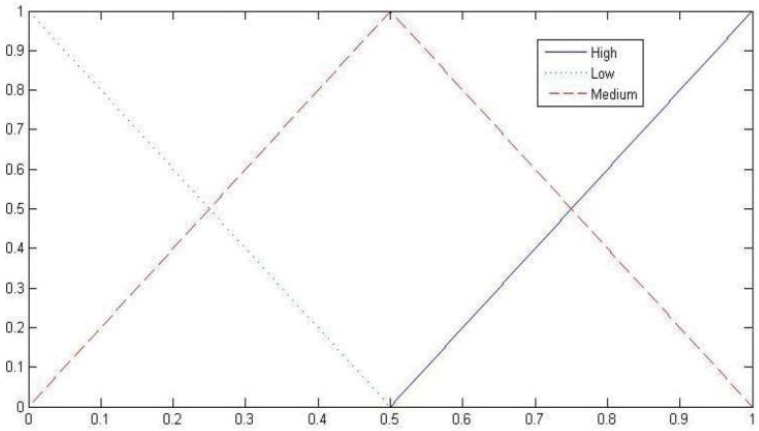


Fig. 8 Scaled Fuzzy Ramp-metering Rate

Table 2 lists all the fuzzy sets for the inputs and output of FRM. The membership functions of the input fuzzy sets are either Gaussian or sigmoid. The membership functions of the output fuzzy sets are either triangular or trapezoidal (Fig 6-13). The v/c ratio is a principal measure of congestion.

Table 2 Terms of the fuzzy sets for inputs and outputs for FRM module

Local Speed (0~100 km/h)	Low	Medium	High
Local Flow Rate (0~4000 veh/h)	Low	Medium	High
Metering Rate (240~900 veh/h)	Low	Medium	High
Local Occupancy (0~30%)	Low	Medium	High
Downstream V/C (0~1)	High		
Downstream Speed (0~100 km/h)	High		
Check-in Occupancy (0~50%)	High		
Queue Occupancy (0~50%)	High		
Interchange Queue Occupancy (0~90%)	High		

Local speed is from 0 to 100 km/h and described as three Gaussian fuzzy set, low, medium and high, with an overlap of 50%. Local occupancy is from 0 to 30% and described as three Gaussian fuzzy set, low, medium and high, with an overlap of 50%. The parameters (centre point and the sigma value) are found by Matlab plot function as shown in Figure 9.

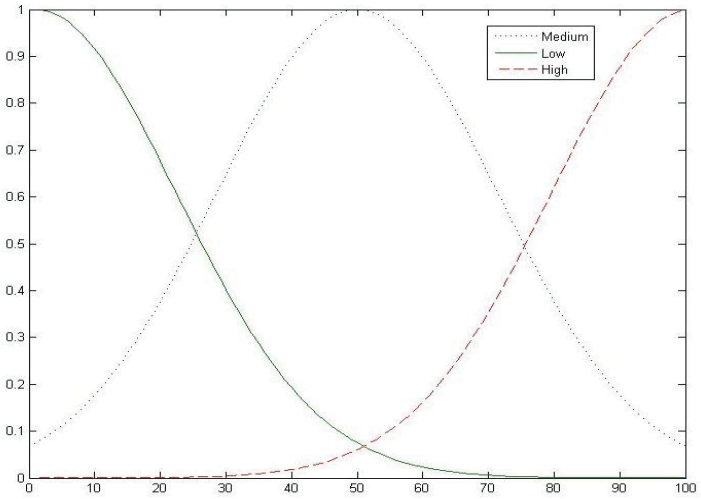


Fig. 9 Membership functions of Local speed (Low, Medium, High)

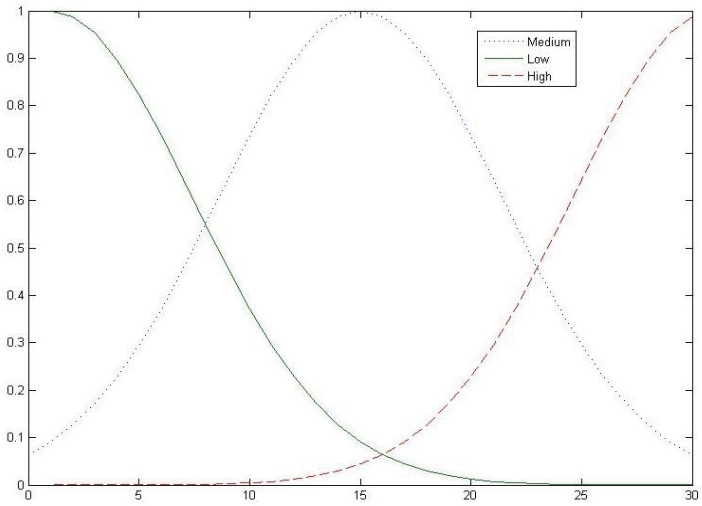


Fig. 10 Membership functions of Local occupancy (Low, Medium, High)

Local flow rate is from 0 to 4000vehs/h and are described as three Gaussian fuzzy set, low, medium and high, with an overlap of 50%. The downstream volume-capacity ratio is from 0 to 1 and fully activated at 0.9. A sigmoid function is used as the membership function. Volume/capacity-ratio (v/c-ratio) calculated with the historical measured maximum flow rate of downstream can be seen as a prediction of the downstream bottleneck behavior. When the congestion occurs, the ratio is at High, and in normal condition the ratio is always at Low.

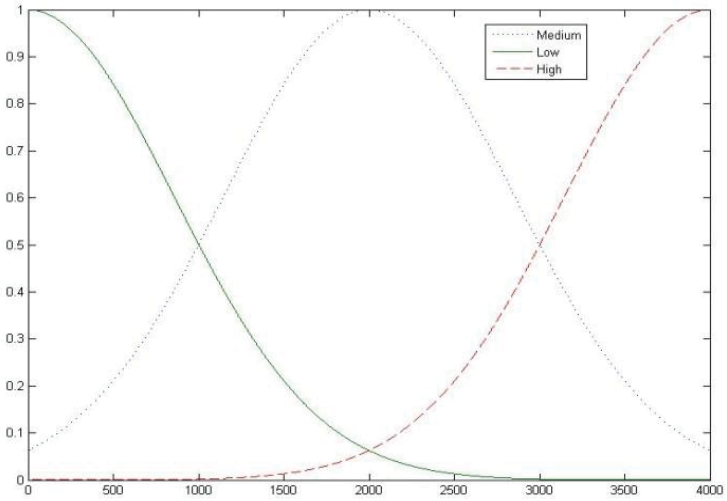


Fig. 11 Membership functions of Local flow rate (Low, Medium and High)

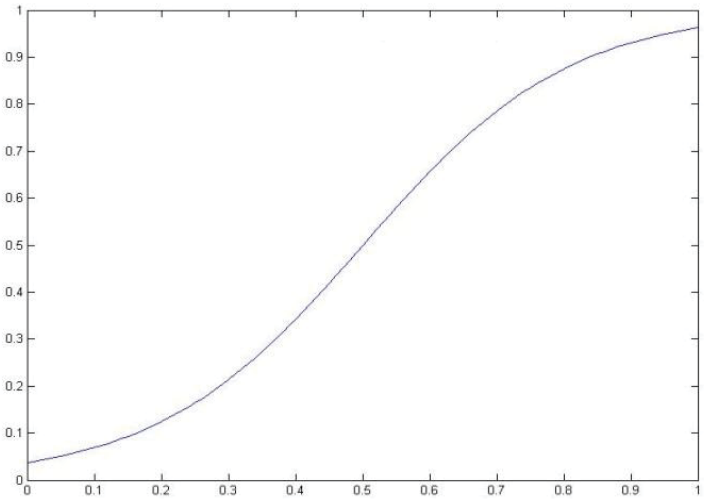


Fig. 12 Membership functions of V/C ratio (High)

Downstream speed is from 0 to 100 km/h and activated at 50km/h and ended at 80km/h. Check-in occupancy is from 0 to 50% and activated from 10% to 30%. A sigmoid function is used as the membership function for both.

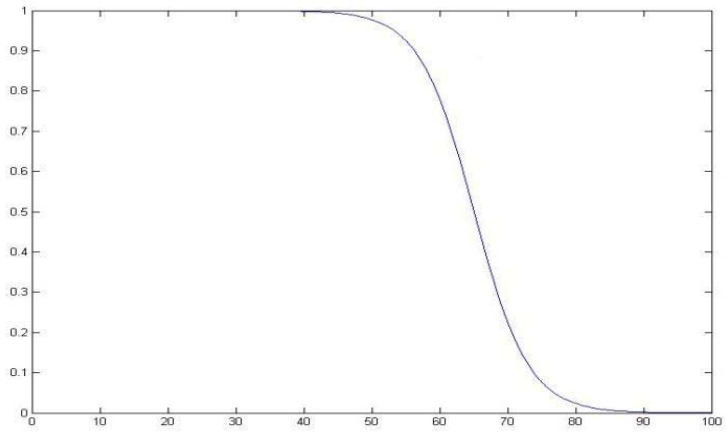


Fig. 13 Membership functions of Down-stream speed (Low)

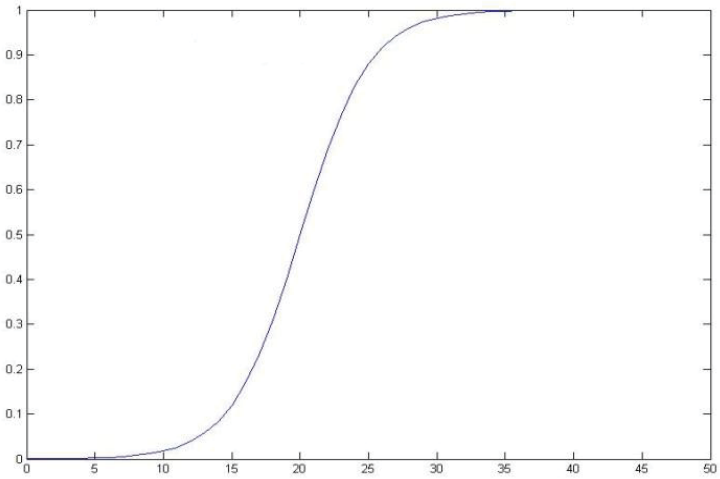


Fig. 14 Membership functions of Check-in occupancy (High)

Queue occupancy is from 0 to 50% and activated from 10% to 30%. Interchange Queue occupancy is from 0 to 90% and activated from 30% to 90%.

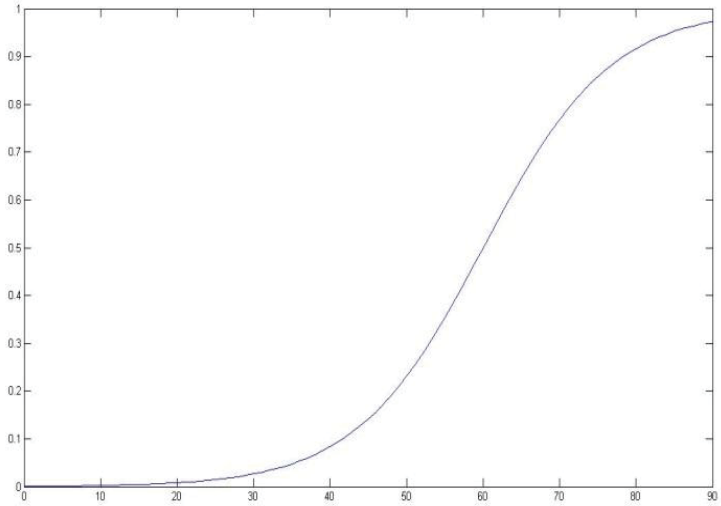


Fig. 15 Membership functions of Queue occupancy (High)

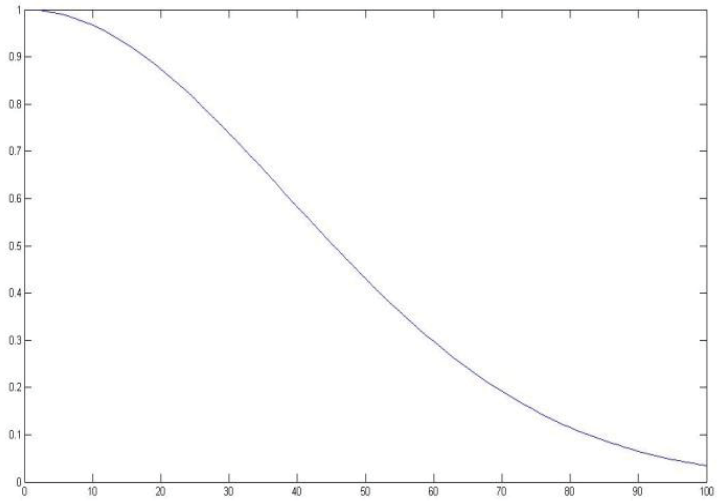


Fig. 16 Membership functions of Interchange queue occupancy (High)

B. Fuzzy Phase Timing (FPT)

The FPT is designed to calculate the green time extension for each individual phase of the interchange. Vehicular detectors are installed on “Up-stream-line” and “stop-line”. The number of approaching vehicles for each approach during given time interval can be estimated using the detectors: the quantity of traffic on the arrival side (Arrival), and the quantity of traffic on the queuing side (Queue).

As the signal control is critical during peak hours, this study considers this scenario only. Suppose the FPT runs Phase B initially with a minimum green time of 6 seconds. The

FPT needs to determine whether to extend or terminate the current green phase when the minimum green time is served.

If Phase B is running green, the movements 5, 6, 7, and 9 are considered as the arrival side, whose total number of vehicles is denoted by a variable *Arrival*, the movements 1 to 4, and 8 are queued side, whose total number of vehicles is denoted by a variable Queue.

If Phase D is running green then the movements 1 and 5 (since 2 and 3 are linked to 1) are on arrival side and the movements 4, 6, 7, and 8 are queue side.

If Phase F is running green then the movements 2, 4, 7 and 8 on the arrival side and the movements 1, 5, 6 and 9 on the queue side.

The input variable Arrival (Fig. 17) is expressed in four fuzzy sets, Few (over 0~14), Small (over 0~28), Medium (over 14~42) and Many (over 28~42), the input variable Queue (Fig. 18) is expressed in four fuzzy sets, Few (over 0~19), Small (over 0~38), Medium (over 19~38) and Many (over 38~57), and the output variable green extension is fuzzified in four fuzzy sets, Very Short (over 0~10 s), Short (over 0~20 s), Medium (over 10~30 s) and Long (over 20~30 s), which is shown in Fig. 19.

Green time extension is the output of the fuzzy variable, which is the extension needed for the green light on the arrival side. In this model, the minimum green time is first given for arriving traffic flows. After that the extension process is stopped and the next phase is selected considering according to Phase Selection Logic (PSL).

C. Rule bases of the FLDI model

There are two rule bases for the FLDI. One is with the FRM module and the other with FPT module. They are built on the combination of input and output variables. The configuration of the fuzzy rules in this study is based on the traffic data collected from the SH1 Upper Harbor Diamond Interchange site survey. The paper used SH1 Upper Harbor Diamond Interchange as a test bed model to examine the performance of the FLDI model in the simulation environment based on the traffic data collected. In other words, by gathering the data we try our best to replicates the operation of the SH1 diamond interchange in the simulation. Even though the FLDI algorithm is designed based on a SH1 Upper Harbor Diamond Interchange, for direct application to any other generic diamond interchange some parameters should be changes. However, the framework and the ideas remain the same. The resulting FRM rule base (Table 3) was formed by 10 rules. The first three rules (Rule 1, 2

and 3) are to make sure that at least one rule will be triggered as all the occupancy range has been covered. Rule 8 is designed to prevent the formation of downstream congestion. Volume/capacity ratio (v/c ratio) is calculated from the historical measured maximum flow rate of downstream and could be seen as a prediction of the downstream bottleneck behavior. Figure 19 is the green time extension only for Phase B. Phase D and F has their own green time extension

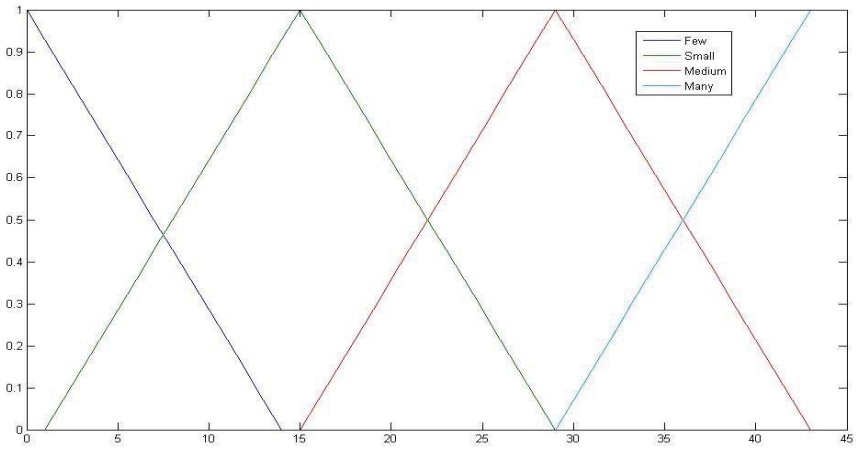


Fig. 17 Total Arrival membership functions (Phase B)

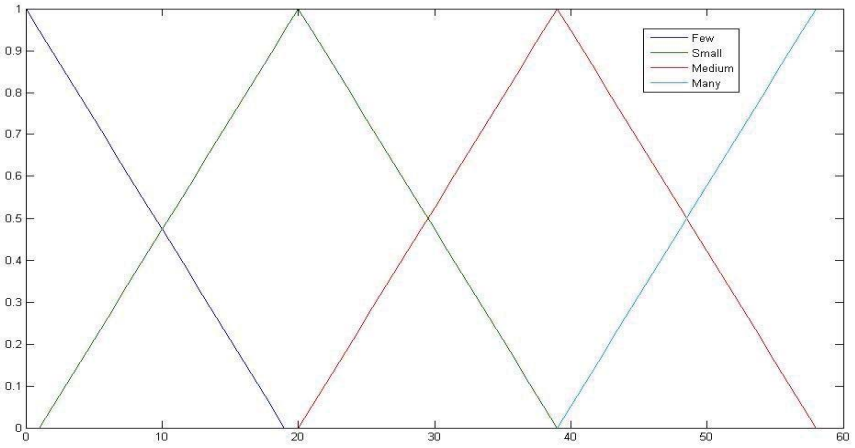


Fig. 18 Total Queue membership functions

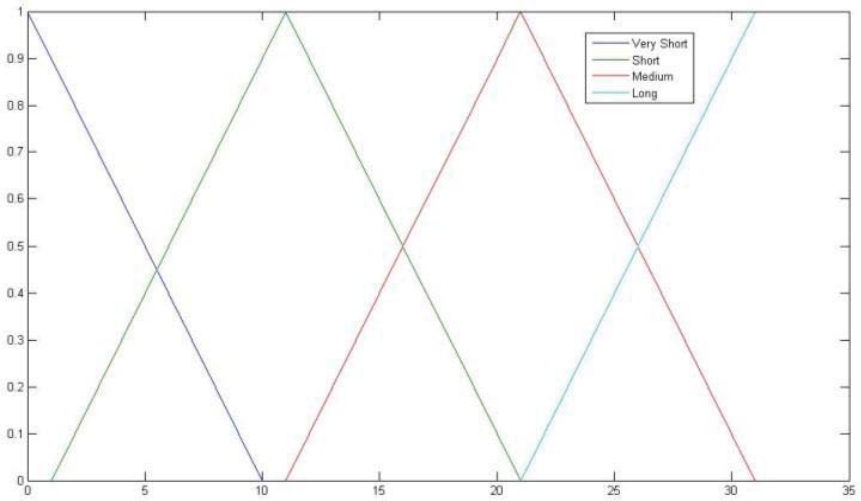


Fig. 19 Phase B Green time extension (fuzzy output)

Table 3 Rule base for fuzzy ramp-metering

Rule	IF		THEN	Weight
	Condition 1	Condition 2	Conclusion	
1	Local occupancy =Low		Metering rate =High	1.5
2	Local occupancy =Medium		Metering rate =Medium	1.5
3	Local occupancy =High		Metering rate =Low	2.0
4	Local speed = Low	Low flow rate =High	Metering rate =Low	2.0
5	Local speed =Medium	Local occupancy =High	Metering rate =Medium	1.0
6	Local speed =Medium	Local occupancy =Low	Metering rate =High	1.0
7	Local speed =High	Low flow rate =High	Metering rate =High	1.0
8	Downstream speed =Very Low	Downstream v/c =Very High	Metering rate =Low	3.0
9	Check-in occupancy =Very High	Queue occupancy =Very high	Metering rate =High	3.0
10	Interchange queue occupancy=High	Queue occupancy =High	Metering rate =High	3.0

Table 4 Fuzzy rules and its weighting for FPT module

Rule	IF		THEN	Weight
	Arrival	Queue	Extension	
1	Few	Few	Very Short	1
2	Few	Small	Very Short	1
3	Few	Medium	Very Short	1
4	Few	Many	Very Short	1
5	Small	Few	Short	2
6	Small	Small	Short	2
7	Small	Medium	Very Short	1
8	Small	Many	Very Short	1
9	Medium	Few	Medium	3
10	Medium	Small	Medium	3
11	Medium	Medium	Short	2
12	Medium	Many	Short	2
13	Many	Few	Long	4
14	Many	Small	Medium	3
15	Many	Medium	Medium	3
16	Few	Many	Short	2

In this particular model, we implemented 10 rules and each weighting from 1.5 to 3.0. The rule weight is to set the priority of each rule (Table 4). The choice of rules weights are based on our best understanding of the traffic condition at the study site (survey and historical data). For example: if Downstream Speed is VERY LOW and Downstream v/c is VERY HIGH, then Metering Rate is LOW. The rule weight for this rule is 3.0 (highest priority).

Rules 1 and 6 are designed to restrict the metering rate when the vehicles are unable to merge onto the motorway. When the motorway is highly congested, a secondary queue of metered vehicles may form. If a secondary queue persists, ramp-metering is no longer providing any benefit

Rule 9 is used for situation when there is a long queue on the ramp; the FRM module has to extend the metering rate to the maximum, to avoid the long queue on the ramp as well as preventing the traffic spillback to the connected intersection.

Rule 10 is designed to link the Fuzzy Ramp-metering with the Interchange's operation solving problem when there are vehicles on the connected intersections waiting to get onto the ramp. If this occurs, the metering rate will be set to high allowing more cars to be

merged onto the motorway/freeway. The interchange queue occupancy placed in various places at the connected intersections.

The rule base for FPT is developed with respect to the three phases B, D and F used during the peak hours. For each phase, there are 16 rules (Table 4).

The defuzzification process converts a fuzzy output variable into a crisp value (metering rate and fuzzy green time extension). In this model, the centroid method is used for the defuzzification process. The equation of the central gravity method shown blow:

$$\frac{\int xf(x)dx}{\int f(x)dx} \quad (2.1)$$

In practice, a discrete fuzzy centroid equation is used to replace the continuous centroid equation since it is easier to calculate.

$$\frac{\sum_{i=1}^{N} w_i c_i I_i}{\sum_{i=1}^{N} w_i c_i} \quad (2.2)$$

where:

N is the numbers of the output classes.

$\mathbf{c_i}$ is the centroid of the ith output class.

$\mathbf{w_i}$ is the results of the aggregation of rules at the ith output class.

$\mathbf{I_i}$ is the area of the ith output class.

4. SIMULATION STUDIES

In this study, the Upper Harbor Diamond Interchange was used as the test bed for the proposed FLDI model. It was assumed there were no bus, pedestrian crossing, bicycle and motorbike. The simulation conducted was for peak hour period from 8:00-9:00AM on a Monday, during which only Phase B, D and F were running. The order of the phases was fixed, starting with Phase B, then moving to Phase F and ending with Phase D, which repeats until the simulation ends. Only one ramp was metered. The interchange (Fig. 2) was simulated in AIMSUN 6. The FLCs were coded in C++ and interfaced to AIMSUN 6 via

an API module. Cars and heavy vehicles are simulated; the percentage of the heavy vehicles is around 5-15% of the total traffic demand depending on the movements.

The simulation model is calibrated based on the previous works of John T. Hughes [27], Christine P. Hass [28], Susan McMillan [29]. The Upper Harbor diamond interchange is built based on the North Shore City Council's calibrated S-Paramics model. Traffic flow and lane configurations were provided by NZ Transport Agency. The model described in this paper was implemented using AIMSUN NG and simulated in AIMSUN version 6.1. The diamond interchange was calibrated using the flow data on each section. Other performance measures, e.g. travel time, travel speeds, and queue length, can be used to calibrate the model [28]. Hughes presented weightings for the different performance measures used to calibrate an AIMSUN model with traffic flow having the most weight followed by travel time, speed and location of traffic jams [27].

The FLDI was evaluated by comparing it with the Actuated Diamond Interchange Control with No Metering (ADINM) and Actuated Diamond Interchange Control with Fuzzy Ramp-metering (ADIFM). The performances of the diamond interchange were measured based on the individual performance of the downstream, ramp, intersections and overall system.

Figures 20 and 21 show the flow percentage of the traffic within the interchange. This interchange is a key interchange for the North Shore Motorway Network as it links the State Highway 1 with the State Highway 18 (Upper Harbour Highway). State Highway 18 connects the western side of Auckland region to the northern part. The SH18 then links with SH1 via this interchange to the northern part of New Zealand. In other words, this interchange is the only gateway to the northern part of New Zealand. Only one ramp was metered by the fuzzy logic because this is the city inbound direction. In the morning, most of the vehicles will be heading south and city center for work.

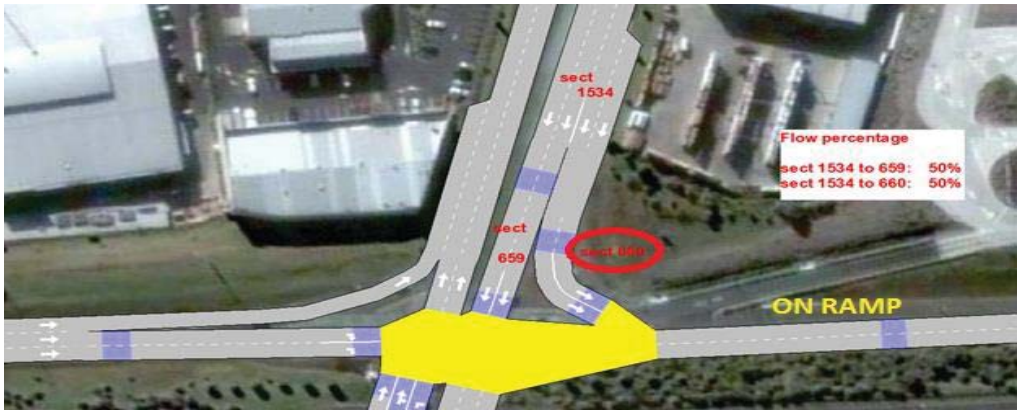


Fig. 20 Turning movements feed into metered-ramp (on ramp city bound)

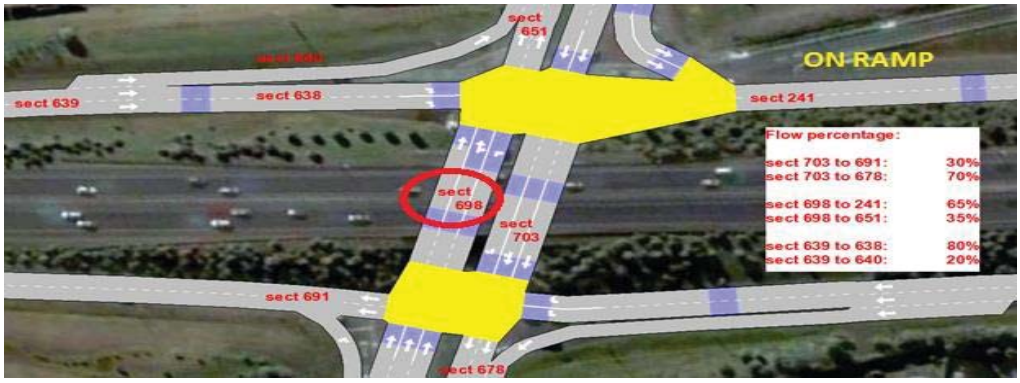


Fig. 21 Other turning movements within the interchange

The performance of the models based on performance of the on ramp linked with the freeway (% change in Downstream Average Speed and % change in Downstream Average Delay) and the performance of the two intersections are based on the Delay Time. The System Total Travel Time is the overall performance of both the ramp and intersections

Table 5 Comparison of the FLDI model with ADINM and ADIFM models for three-phase control and equal traffic volumes

		Average Delay Time (second per vehicle)		
Scenario	Traffic volumes (vehicles per hour)	ADINM model	ADIFM model	FLDI model
1	3348	21.28	11.17	5.16
2	3752	24.00	32.71	7.17
3	4592	40.04	103.87	17.09
4	6272	70.18	138.50	133.75
5	7952	88.37	151.97	149.24
6	8783	130.28	152.94	152.59

Table 6 Traffic demand data for Scenario 1 to 6

Table 5.1 Traffic demand data for Scenario 1 (3348 vehicles per hour)

Time (AM)		Constellation towards Upper Harbour Highway		Upper Harbour towards Constellation Drive		SH1 between Greville Road to Tristam Avenue (South-bound)		SH1 between Tristam Avenue to Massey Exit (North-bound)	
From	*To*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*
8:00	8:15	379	19	278	14	1344	67	36	2
8:15	8:30	430	22	384	19	1582	79	96	5
8:30	8:45	481	24	490	25	1749	87	137	7
8:45	9:00	727	36	509	25	2216	111	254	13
Average Demand		504	25	415	21	1723	86	131	7
Total Traffic Demand		529		436		1809		138	

Table 5.2 Traffic demand data for Scenario 2 (3752 vehicles per hour)

Time (AM)		Constellation towards Upper Harbour Highway		Upper Harbour towards Constellation Drive		SH1 between Greville Road to Tristam Avenue (South-bound)		SH1 between Tristam Avenue to Massey Exit (North-bound)	
From	*To*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*
8:00	8:15	579	29	478	24	1544	77	256	12
8:15	8:30	630	32	584	29	1782	89	296	15
8:30	8:45	681	34	690	35	1949	97	337	17
8:45	9:00	927	46	709	35	2416	121	454	23
Average Demand		704	35	615	31	1923	96	331	17
Total Traffic Demand		739		646		2019		348	

Table 5.3 Traffic demand data for Scenario 3 (4592 vehicles per hour)

Time (AM)		Constellation towards Upper Harbour Highway		Upper Harbour towards Constellation Drive		SH1 between Greville Road to Tristam Avenue (South-bound)		SH1 between Tristam Avenue to Massey Exit (North-bound)	
From	*To*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*
8:00	8:15	779	39	678	34	1744	87	436	22
8:15	8:30	830	42	784	39	1982	99	496	25
8:30	8:45	881	44	890	45	2149	107	537	27
8:45	9:00	1127	56	909	45	2616	131	654	33
Average Demand		904	45	815	41	2123	106	531	27
Total Traffic Demand		949		856		2229		558	

Table 5.4 Traffic demand data for Scenario 4 (6272 vehicles per hour)

Time (AM)		Constellation towards Upper Harbour Highway		Upper Harbour towards Constellation Drive		SH1 between Greville Road to Tristam Avenue (South-bound)		SH1 between Tristam Avenue to Massey Exit (North-bound)	
From	*To*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*
8:00	8:15	1179	59	1078	54	2144	107	836	42
8:15	8:30	1230	62	1184	59	2382	119	896	45
8:30	8:45	1281	64	1290	65	2549	127	937	47
8:45	9:00	1527	76	1309	65	3016	151	1054	53
Average Demand		1304	65	1215	61	2523	126	931	47
Total Traffic Demand		1369		1276		2649		978	

Table 5.5 Traffic demand data for Scenario 5 (7952 vehicles per hour)

Time (AM)		Constellation towards Upper Harbour Highway		Upper Harbour towards Constellation Drive		SH1 between Greville Road to Tristam Avenue (South-bound)		SH1 between Tristam Avenue to Massey Exit (North-bound)	
From	*To*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*
8:00	8:15	1579	79	1478	74	2544	127	1236	62
8:15	8:30	1630	82	1584	79	2782	139	1296	65
8:30	8:45	1681	84	1690	85	2949	147	1337	67
8:45	9:00	1927	96	1709	85	3416	171	1454	73
Average Demand		1704	85	1615	81	2923	146	1331	67
Total Traffic Demand		1789		1696		3069		1398	

Table 5.6 Traffic demand data for Scenario 6 (8783 vehicles per hour)

Time (AM)		Constellation towards Upper Harbour Highway		Upper Harbour towards Constellation Drive		SH1 between Greville Road to Tristam Avenue (South-bound)		SH1 between Tristam Avenue to Massey Exit (North-bound)	
From	*To*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*	*Car*	*HOV*
8:00	8:15	1779	89	1678	84	2744	137	1436	72
8:15	8:30	1830	92	1784	89	2982	149	1496	75
8:30	8:45	1881	94	1890	95	3149	157	1537	77
8:45	9:00	2127	106	1909	95	3616	181	1654	83
Average Demand		1904	95	1815	91	3123	156	1531	77
Total Traffic Demand		2000		1906		3279		1598	

There were six testing scenarios (Table 5 and 6), with the total traffic demanding ranging from 3348 to 8783 veh/h of varying flow rate. The varying flow rate allows complex traffic situations and the scenarios cover most of the real life traffic conditions from free flow to congestion. In each scenario, there were three simulation runs, one for each model. Partial results are given in Table 5 and depicted in Fig. 22. . The total traffic demand for the interchange is approximately 3000-400 vehicles / hour. We purposely set up the traffic demand to be higher than the actual capacity to test the model under the extreme conditions where traffic demand is higher than the capacity. We want to see how the model reacted to these scenarios (scenario 4, 5 and 6). The total volume includes the through traffic in the main highway.

According to Table 5 and Fig. 22, the FLDI model gives smaller delays than the ADINM and ADIFM when the traffic volumes are below 5000 veh/h. However, if traffic volumes are more than 5000 veh/h, the FLDI gives higher delays than the ADINM but performs better than the ADIFM. The FLDI controller decreases the delays of the vehicles about 36.81% comparing to the ADIFM model and 4.4% comparing to the ADINM model.

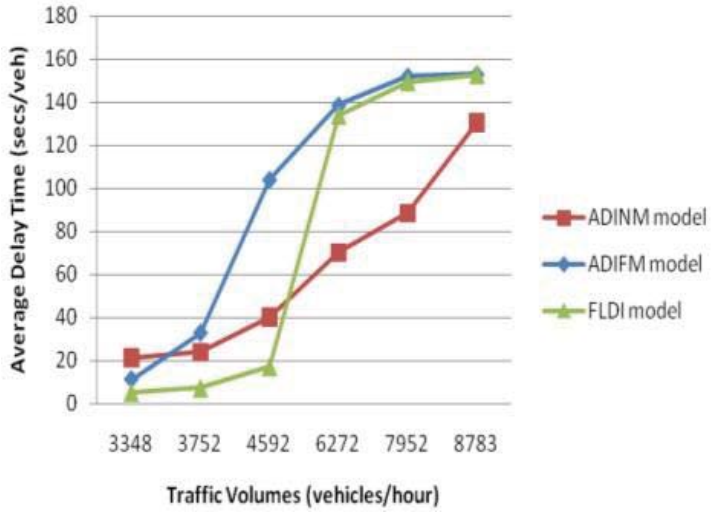


Fig. 22 Delay time comparison of the FLDI, ADINM and ADIFM

Fig. 23 shows the percentage change in Total Travel Time (TTT) of FLDI and ADIFM for six scenarios with Total Traffic Demand ranging from 3348 to 8783 veh/h. When the total traffic demand ranges from 4592 to 7952 veh/h, the FLDI and ADIFM fail to reduce

the system's TTT which means the ramp-metering for both models are not working properly and even cause the extra delay for the traffic condition. When the total traffic demand reaches 7952 veh/h the performance of FLDI and ADIFM are very close.

Fig. 23 Percentage change in System Total Travel Time of ADIFM and FLDI

When the total traffic demand reaches 8784 veh/h, both FLDI and ADIFM reduce the TTT by 3%. However, during the non congestion periods (3348 to 4592 veh/h), FLDI performs better than ADIFM model. The FLDI reduces the TTT from -16.12% to -17.29% while the ADIFM increases the TTT from -13.59% to 7.46%. The main reason for this difference is that the FLDI takes the connected intersection's operation into account. In some cases, when there are many cars waiting to get onto the ramp from the intersections and also when there is a long queue on the ramp, the ramp meter will increase its metering flow rate to the maximum to flush as many vehicles as possible into the motorway to avoid the vehicles on ramp spillback into the intersections.

Fig. 24 shows the percentage change in Downstream Average Speed of FLDI and ADIFM for six scenarios with Total Traffic Demand ranging from 3348 to 8783 veh/h. The FLDI and ADIFM do no increase the downstream speed under free flow condition where traffic demand ranges from 3348 to 4592 vehicles per hour. The reason may be because the ramp demand is relatively low and the motorway is under free flow condition.

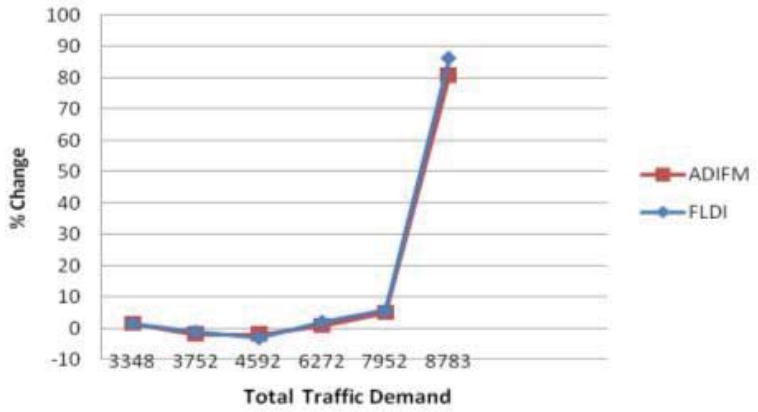


Fig. 24 Percentage change in Downstream Average Speed (ADIFM and FLDI)

In the extreme conditions when total traffic demand is higher than the road capacity, in this case the traffic is congested with the ADINM model the vehicles are not moving, however in the ADIFM and FLDI because there is a ramp-metering control cars are still able to move and merge into the motorway (slow speed) so therefore it is not as congested as in the ADINM model. Therefore, we can see that the ADIFM and FLDI reduce the congestion (not significantly). The reason why the ADIFM and FLDI have higher Delay Time is because of the extension of the green time for certain movements. Because of this extension, the Delay Time goes up. Most of the control methods which control the green time extension experienced this situation. In scenario 6 (8783 vehicles per hour), with extremely high traffic demand where the traffic demand exceed the freeway's capacity, the delay time of the on ramp could be higher than the uncontrolled ramp but still improves the overall freeway's speed. It is shown in Fig. 25 and 26 below:

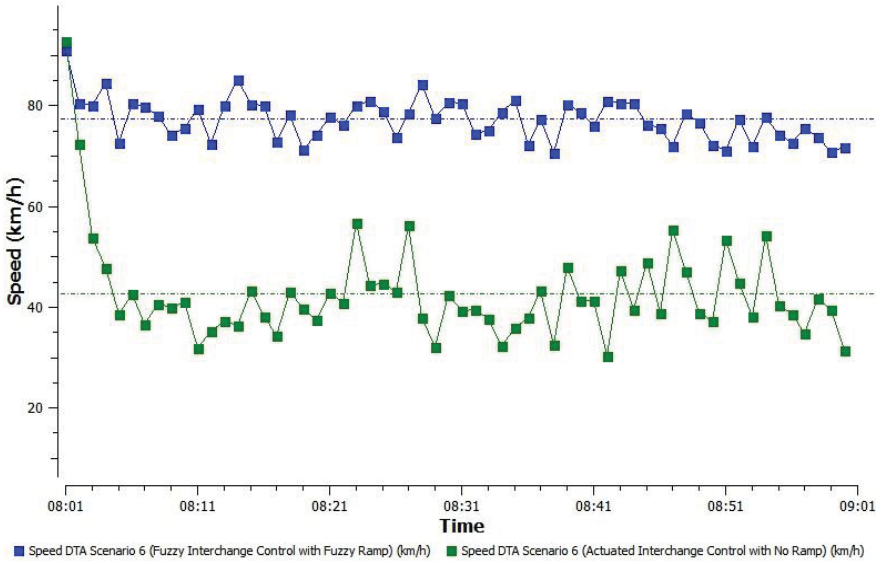


Fig. 25 Scenario 6 Downstream average speed (FLDI and ADINM)

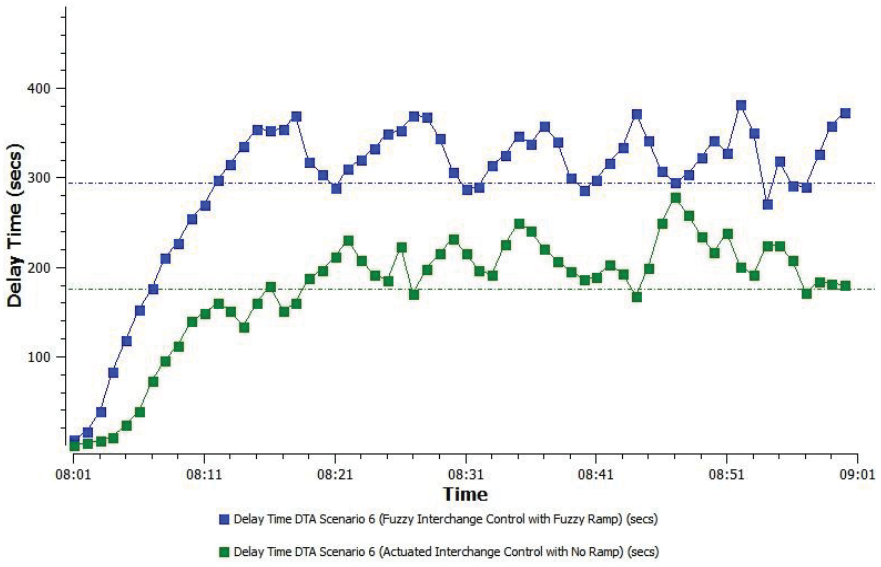


Fig. 26 Scenario 6 On ramp delay time (FLDI and ADINM)

When total traffic demand reaches 8783 veh/h (exceeds the road capacity) the FLDI perform better than ADIFM by 1.47% in term of speed. Moreover, the FLDI's average delay of downstream vehicles is better than ADIFM. And these are the advantage of the FLDI model over ADIFM, it performs better in extreme conditions (for example: if there is a big soccer match between NZ and Brazil in one side of the city, this will cause an increase in the traffic demand that may exceed the road capacity).

Fig. 27 Percentage change in Downstream Average Delay (ADIFM and FLDI)

Fig. 27 shows the percentage change in Downstream Average Delay (DAD) of FLDI and ADIFM for six scenarios with Total Traffic Demand ranging from 3348 to 8783 veh/h. Similar to the Downstream Average Speed, the FLDI and ADIFM don't reduce the delay significantly when the total traffic demand is less than 6727 veh/h during the non congestion periods. Situation starts to change once the total traffic demand exceed 6272 veh/h as FLDI perform better than the ADIFM by further reducing the average delay of downstream vehicles.

Overall, the FLDI algorithm performs better than the ADIFM when the traffic is congested. This makes sense as the fuzzy membership functions were designed to cope with different traffic conditions and emphasised on the heavily congested traffic conditions. The FLDI, FPT and FRM form the closed loop feedback system, the queue at the on-ramp has an effect on the signal coordination and bottleneck on the main highway (at downstream of the merge point). However, the total travel time (TTT) in this simulation

does not change much as it was averaged out by the traffic in other movements that is not affected by the fuzzy logic and ramp metering.

5. CONCLUSION

Fuzzy Logic Diamond Interchange Control with Fuzzy Ramp Metering (FLDI), Actuated Diamond Interchange Control with No Ramp Metering (ADIFM) and Actuated Diamond Interchange Control with Fuzzy Ramp (ADINM) were built and tested in AIMSUN 6 environment with six different traffic scenarios ranges from free flow to congestion. From the comparison results, FLDI outperforms the ADIFM and ADINM models with significant improvements in system total travel time, downstream average delay and average speed even when the traffic demand exceed the road capacity (including arterial roads, ramp and motorway). The ADIFM becomes less effective once the total traffic demand is much higher than road capacity. The percentage of change in the system total travel time, downstream average delay and average speed (six scenarios) shows the reason why FLDI model has better performance the other models, the FLDI model is designed to avoid the formation of congestion and in extreme traffic conditions the algorithm also prevents the congestion getting worse. We have proposed a unique integration of fuzzy intersection signals and ramp-metering signal at a diamond interchange that reduces the total travel time and delay of the integrated diamond interchange and metered ramp simultaneously.

ACKNOWLEDGEMENT

Thanks are given to Aaron Yu and the staff member at Auckland Traffic Management Unit, North Shore City Council and TSS Spain for their support in this study.

REFERENCES

[1]. Z.Z. Tian, “Development and Evaluation of Operational Strategies for Providing an Integrated Diamond Interchange – Ramp Metering Control System”, *Ph. D Thesis*, Department of Civil & Environmental Engineering , Texas A&M University, USA, 2004

[2]. C.J. Messer, D.B. Fambro and S.H. Richards, “Optimization of Pretimed Signalized Diamond interchange”, *Transportation Research Record 644*, Transportation Research Board, Washington, D.C., pp.78-84, 1977.

[3]. Highway Capacity Manual 2000, *Transportation Research Board*, National Research Council, Washington DC, 2000.

[4]. A.J. Miller, "A Computer Control System for Traffic Networks", Birmingham, England, 1963.

[5]. P.B. Hunt, et al., "The SCOOT on-line-traffic-signal optimization technique" *Traffic Engineering and Control*, vol.23, no.4, pp.190-192, 1982.

[6]. J.Y.K. Luk, "Two Traffic Responsive Area Traffic Control Methods: SCATS and SCOOT", *Traffic Engineering and Control*, vol. 30, pp.14-20, 1984.

[7]. P. Charles, "Time for a Change", *Traffic Technology International*, pp. 68-71, June/July, 2001.

[8]. N.H. Gartner, "OPAC: A Demand-Responsive Strategy for Traffic Signal Control", *Transportation Research Record 906*, Transportation Research Board, Washington, DC, pp. 75-81, 1983.

[9]. N.H. Gartner. and F.J. Pooran, "Implementation and Field Testing of the OPAC Adaptive Control Strategy in RT-TRACS", *the 81st Annual TRB Meeting*, Washington DC, January 2002

[10]. U. Sen and K.L. Head, "Controlled Optimization of Phases at an Intersection", *Transportation Science*, vol. 31, no.1, pp.5-17, February, 1997.

[11]. P. Mirchandani and L. Head, "A Real-time Traffic Signal Control System: Architecture, Algorithms and Analysis", *Transportation Research,* Part C, 9, pp.414-432, 2001.

[12]. F. C. Fang and L. Elefteriadou, "Development of an Optimization Methodology for Adaptive Traffic Signal Control at Diamond interchange", *ASCE Journal of Transportation Engineering*, vol. 132, no.8, pp.629-637, 2006.

[13]. Z. Muhurdarevic, H. Condie, E. Ben-Shabat, O. Ernhofer, H. Hadj-SSalem and I. Papamichail, "Ramp-metering in Europe – state-of-art and beyond", *5th International Conference on Traffic and Transportation Studies* (ICTTS), Xi'an, China, August 2-4, 2006.

[14]. T. Brown, I. Clark, P. Evans and L. Wee, "New Zealand's easy merge ramp signal (ramp-metering) trial", *IPENZ*, no.8, pp.1-14, 2005.

[15]. Z.Z. Tian, K. Balke, B. Engelbrecht and L. Rilett, “Integrated control strategies for surface street and freeway systems“, *Transportation Research Record 1811*, pp.92-99, 2002.

[16]. Z.Z. Tian, “Modelling and implementation of an integrated ramp-metering-diamond interchange control system”, *Journal of Transportation Systems Engineering and Information Technology*, vol.7, no.1, pp.61-72, 2007.

[17]. K. Bogenberger, and H. Keller, “An Evolutionary Fuzzy System for Coordinated and Traffic Responsive Ramp-metering,” *Annual Hawaiian International Conference on System Sciences*, USA, pp. 1-3, 2000

[18]. C.P. Pappis and E.H. Mamdani, “A Fuzzy Logic Controller for a Traffic Junction”, *IEEE Transactions System, Man and Cybernetics*, vol. SMC-7, no.10, pp. 707-717, 1977.

[19]. R. Hoyer, and U. Jumar, "An Advanced Fuzzy Controller for Traffic Lights", *Artificial Intelligence in Real Time Control Symposium*, Valencia, Spain, pp. 3-6, 1994.

[20]. M.B. Trabia, M.S. Kaseko and M. Ande, “A two-stage fuzzy logic controller for traffic signals”, *Transportation Research*, Part C 7, pp. 353-367, 1999.

[21]. Sisson B. 2006. Baseplus FUSE: The Link Between SCATS and S-Paramics, *Proceedings New Zealand Microsimulation Conference*, Queenstown, New Zealand.

[22]. Favilla J., Machion A., Gomide F., “Fuzzy traffic control: adaptive strategies”, Second IEEE International Conference on Fuzzy Systems II, 1993, pp. 506-511.

[23]. Jamshidi M., Kelsey R., Bisset K., “A simulation environment of fuzzy control of traffic systems”, 12 IFAC-World Congress, Sydney, Australia, 1993, pp. 553-556.

[24]. M. Nakatsuyama, H. Nagahashi, N. Nishizuka, “Fuzzy logic phase controller for traffic functions in the one-way arterial road”, in: Proc. IFAC 9th Triennial World Congress, Pergamon Press, Oxford, 1984, pp. 2865-1870.

[25]. C. Chou, and J. Teng, “A fuzzy logic controller for traffic junction signals,” Information Science, vol 143(1-4), 2002, pp. 73-97.

[26]. C.T. Wannige and D.U.J Sonnadara, “Adaptive neuro-fuzzy traffic signals control for multiple junctions,” Fourth International Conference on Industrial and Information Systems, ICIIS 2009, Srilanka, December 2009, pp. 262-267.

[27]. John T. Hughes, “Micro-Simulation of Auckland Central Motorways for Traffic Management”, Master’s Thesis, University of Auckland, 1999.

[28]. Christine P. Hass, “Assessing Developments Using AIMSUN”, Institution of Professional Engineers New Zealand. Annual conference 2001, Auckland.

[29]. Susan McMillan, “Incident Management Modelling Using Microsimulation with Adaptive Signal Control”, ITS Asia-Pacific, Bangkok, July 2009,