SIGNALIZED FUZZY LOGIC FOR DIAMOND INTERCHANGES INCORPORATING WITH FUZZY RAMP SYSTEM

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

Master of Engineering
in
Mechatronics

at
Massey University,
Auckland, New Zealand

Cao Van Pham

June 2009

ABSTRACT

New dynamic signal control methods such as fuzzy logic and artificial intelligence developed recently mainly focused on isolated intersection. In this study, a Fuzzy Logic Control for a Diamond Interchange incorporating with Fuzzy Ramp System (FLDI) has been developed. The signalization of two closely spaced intersections in a diamond interchange is a complicated problem that includes both increasing the diamond interchange capacity and reduce delays at the same time. The model comprises of three main modules. The Fuzzy Phase Timing module controls the current phase green time extension, the Phase Selection module select the next phase based on the pre-defined phase sequence or phase logics and the Fuzzy Ramp module determines the cycle time of the ramp meter bases on current traffic volumes and conditions of the interchanges and the motorways. The developed FLDI model has been compared with the traffic actuated simulation with respects to flow rates and the average delays of the vehicles. The model of an actual diamond interchange is described and simulated by using AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network) software. Simulation results show the FLDI model outperformed the traffic actuated models with lower system total travel time, average delay and improvements in downstream average speed and average delay.

ACKNOWLEDGEMENTS

Author wishes to express his deepest gratitude and sincerest appreciation to his advisors Prof. Peter Xu, Dr. Fakhrul Alam and Dr. Johan Potgieter for supervision, enthusiastic guidance and continuous encouragement throughout the course of study. Grateful acknowledgment is also extended to Dr. Clara Fang (University of Hartford) for her valuable comments and suggestions.

Gratitude is also expressed to the staffs of Auckland Traffic Management Unit for their help and support. Author is also thankful to his friends Ben Lin and Aeron Yu for assisting in many ways.

Finally author owes a special dept of thanks to his family members who have been behind all his achievements in life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ADIM	Actuated Diamond Interchange Control with No Ramp Metering
ADIFM	Actuated Diamond Interchange Control with Fuzzy Ramp Metering
AIMSUN	Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network
DAD	Downstream Average Delay
DAS	Downstream Average Speed
FLC	Fuzzy Logic Control
FLDI	Fuzzy Logic Diamond Interchange Control
HOV	High Occupancy Vehicle
SH1	State Highway 1
MF	Membership function
MOE	Measures of Effectiveness
MOTORWAY	Freeway (US)
NZMOT	New Zealand Ministry of Transportation
STTT	System Total Travel Time
TFL	Traffic Light
TNZ	Transit New Zealand (is now part of New Zealand Transport Agency)
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 signalized when the demand flows are high (peak hours). The signalized 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.

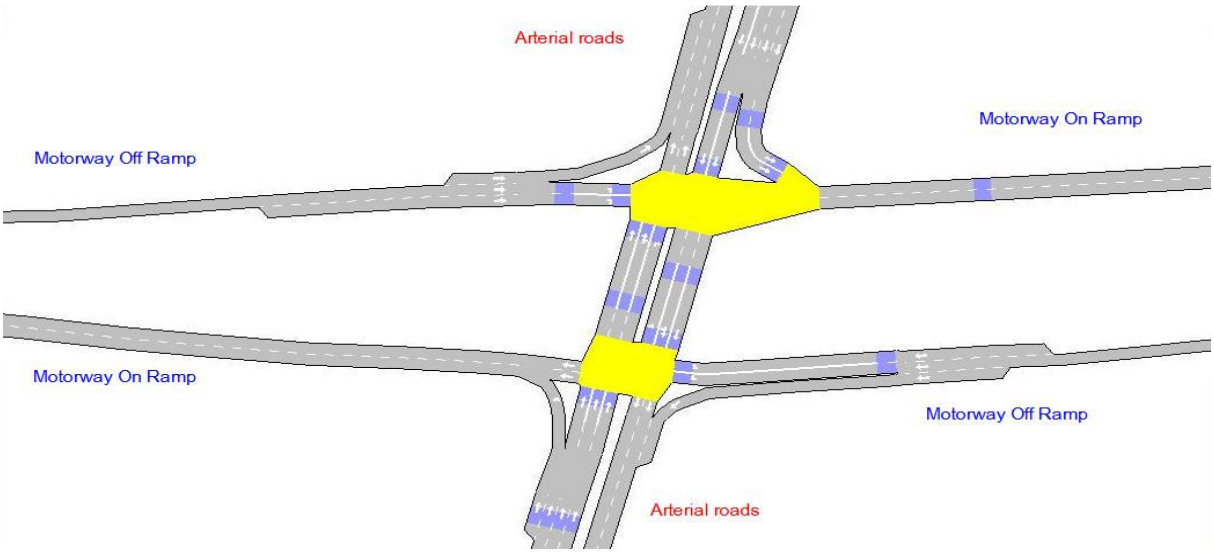


Figure 1.1 Geometric layout of a Diamond Interchange

1.2 Adaptive Signal Control

Pre-timed control operates on pre-defined, fixed intervals and phase timings with fixed cycle length and phase sequences. This signal control is widely used in rural areas and some minor intersections in urban areas around New Zealand. During the peak hour periods when the traffic conditions change substantially, the timing plan will become less effective.

Actuated control uses gap-seek logic to respond to traffic fluctuations. 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.

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 respond 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 has been calculated based on the historic traffic data. The disadvantage with these approaches is they become inefficient in responding 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 (Bell, 1990).

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. Miller (1963) came up with this concept; he created an algorithm for adjusting signal timings in small time intervals of 1-2 seconds. A decision to be made is whether to extend the current green duration or terminate it immediately. The algorithm calculates the difference in vehicle-seconds of delay between the gain made during an extension and the loss in the cross-street resulting from that extension (Fang, 2004).

1.3 Fuzzy Logic Control

The fuzzy logic control determines the duration and sequence that the traffic light should stay in a certain state, before switching to the next state. The amount of arriving and waiting vehicles area quantised into fuzzy variables, like many, medium and none. The activation of the variables in a certain situation is given by a membership function, so where there are 5 cars in the queue, this might result in an activation of 25% of 'many' and 75% of 'medium'. 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, and reducing waiting time. A disadvantage of the controller seems to be its dependence on the preset quantification values for the fuzzy variables.

1.4 Objectives of the Thesis

This thesis proposes a new fuzzy logic algorithm for controlling the traffic flow at a diamond interchange with least modifications to the current infrastructure. The model of an actual interchange is described and simulated by using AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network) software. The controllers based on fuzzy logic are implemented in C++ and imported into AIMSUN. The proposed method adaptively controls the cycle of traffic signals of the current running phase to accommodate different traffic volumes. There are two main simulations in this project; the first one is the simulation of the fuzzy ramp by itself and the second part is the simulation of the diamond interchange incorporating with the fuzzy ramp. The results obtained from the implementation of fuzzy logic strategy based on the simulations are compared against those of a conventional controller. The performance criteria are the flow/volume/demand of the traffic and minimization of the travel time.

1.5 Scope of the Study

The final objective of this project is to develop a universal fuzzy logic algorithm for controlling the traffic flow at any local diamond interchanges.

CHAPTER 2

LITERATURE REVIEW

2.1 Traffic Signal Control for Intersection

There are three common traffic signal controls at intersections that are widely used in New Zealand: pre-timed, actuated and adaptive control.

Highway Capacity Manual (2000) definitions for pre-timed and actuated controls:

a. Pre-timed control: Pre-timed controllers have a pre-set sequence of phase displayed in repetitive order. Each phase has fixed green time and change intervals that are repeated in each cycle to produce a constant cycle length.

b. Actuated control is divided three types: semi-actuated, fully actuated and adaptive:

- Semi-actuated: Semi-actuated controllers operate with some approaches (typically on the minor street) having detectors. The earliest form of semi-actuated control was designed to maintain the green on the major street in the absence of a minor-street actuation. Once actuated, the minor-street green is displayed for a period just long enough to accommodate the traffic demand.

- Fully actuated: Fully actuated controllers operate with timing on all approaches being influenced by vehicle detectors. Each phase is subject to a minimum and maximum green time, and some phases may be skipped if no demand is detected. The cycle length for fully actuated control will vary from cycle to cycle.

- Adaptive signal control: Allows for on-line optimal signal timings based on prevailing demand. The system collects data on traffic flow conditions, determines a desired timing plan and then implements the timing plan in a time period such as each cycle, or every 5 minutes.

There are two notable real-time systems deployed based on the optimization of parameters (e.g., cycle time, splits and offsets). They are SCOOT - Split, Cycle 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 predictions using 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 cycle. The demand profiles obtained upstream and downstream of each intersection are used to assess whether a change in offset would minimize the overall disutility of delay for that cycle. The cycle length optimizer tries to maintain the degree of saturation for the heaviest intersection at 90 percent. A change in cycle 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 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).

2.2 Traffic-responsive Ramp Metering System

Traffic-responsive ramp metering system can automatically adjust the ramp metering rate based on current traffic conditions in the local ramp. Local traffic condition will be detected by loop detectors. Controller electronics and software algorithms can select an appropriate metering rate based on the occupancy or flow data from the ramp and mainline detectors; therefore, traffic-responsive ramp metering systems can generally deliver better results than fixed time metering (Bogenberger et al., 2001). For a fixed time ramp metering, when downstream density exceeds the critical point, congestion may form and degrade the downstream capacity therefore the traffic-responsive ramp metering can reduce motorway congestion in space and time as well as increase motorway throughput.

Local responsive ramp metering consists of physical components that are described in the following sections (Figure 2.1):

2.2.1 Ramp Metering Signal

The signal is typically located to the drivers left, or on both sides of the ramp. Each ramp meter typically has one nearby weatherproof control cabinet that houses the controller, modems, and

inputs of each loop. The controller is set to a specific algorithm, which generates the ramp-metering rate.

2.2.2 Presence detector

The check-in, presence, or demand detector is located at the ramp cordon line. The check-in detector notifies the controller that a vehicle is waiting on ramp and to activate the green cycle.

2.2.3 Passage Detector / Merge detector

The passage detector sometimes also known as merge detector is an optional component that senses the presence of vehicles in the primary merging area of the ramp. This prevents waiting vehicle from passing the ramp signal while the front vehicle still stopping in the merging area for some unexpected reasons.

2.2.4 Queue detector

The queue detector is located at the end of a ramp. The queue detector prevents vehicles from spilling over onto the surface street network.

2.2. 5 Downstream / Upstream detector

The detector collects the information of upstream or downstream traffic condition to feedback to ramp controller.

2.2.6 High Occupancy Vehicle detector

High Occupancy Vehicle detector collects the information of the trucks and buses entering the freeway. HOVs do not have to wait for the signal as these vehicles often have slow acceleration.

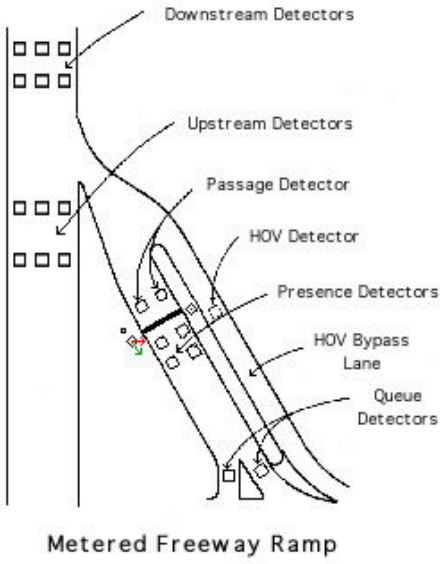


Figure 2.1 Traffic-responsive ramp metering system (Caltrans, 1991)

2.3 Ramp Metering Algorithm (Taale et al., 2000)

2.3.1 RWS Algorithm

The RWS algorithm is based on the flows on the motorway and on-ramp and the speed of the traffic on the motorway. The measurements are smoothed to reduce variations. The strategy aims at a good use of the capacity available. The heart of the algorithm consists of the following calculations. The number of vehicles allowed to enter the motorway is calculated with (Taale et al., 2000)

$$r_k = C - I_{k-1} \tag{2.1}$$

Where:

r_k is the number of vehicles allowed to enter the motorway in time interval k

C is the pre-specified capacity of the motorway downstream the on-ramp and the variable

I_{k-1} is the measured and smoothed flow upstream the on-ramp in the previous time interval

The cycle time of the metering system is then calculated with (Taale et al., 2000)

$$t = \frac{n * 3600}{r_k} \tag{2.2}$$

Where t is the cycle time in seconds and n the number of lanes on the on-ramp.

This calculated cycle time is compared with a minimum and maximum value and if necessary these values are used instead of the calculated one. If there is a queue on the on-ramp the calculated cycle time can be overruled with the minimum cycle time (Taale et al., 2000). According to U.K Handbook of Ramp Metering (2004), the metering rate generated by this algorithm is unstable when there are changes in the up-stream flow rate e.g. system disturbances, vehicles merging difficulties as well as slow vehicles.

2.3.2 ALINEA Algorithm (Joseph, 2003)

The ALINEA algorithm was developed by the Technical University of Munich and tested by INRETS on the Boulevard Périphérique (Paris) and by Rijkswaterstaat on the A10-West motorway (ring road of Amsterdam) in the framework of the DRIVE I project CHRISTIANE. This strategy tries to keep the occupancy downstream the on-ramp on a certain pre-specified value: the occupancy set point. The rules for switching the system on and off are the same as for the RWS algorithm. Only the calculation of the number of vehicles allowed to enter the motorway is different. It is based on the occupancy downstream the on-ramp. The formula is: (Joseph, 2003)

$$r_k = r_{k-1} + K[O_s - O_{k-1}] \tag{2.3}$$

Where:

r_k is the number of vehicles allowed to enter the motorway in time interval k

r_{k-1} is the number of vehicles allowed to enter the motorway in the previous time interval

K a constant

O_s the occupancy set-point

O_{k-1} the occupancy measured downstream the on-ramp in the previous time interval

The cycle time calculation and the other metering conditions are the same as for the RWS algorithm (Taale et al., 2000). Similar to RWS algorithm, ALINEA algorithm does not take ramp queue spill-back situation into account and because of this the algorithm generates over-restrictive metering rates causing the unbalance between freeway congestion and ramp queue length (Joseph, 2003).

2.4 Fuzzy Ramp Metering

Fuzzy logic can handle approximate information in a systematic way, making it ideal for controlling nonlinear systems and for modelling complex systems where an inexact model exists or systems where ambiguity or vagueness is common (Bogenberger, 2000). Fuzzy Logic has been used widely in designing controllers for real dynamic plants, industrial process. The strength of the theory of fuzzy sets lies in its capability of rendering powerful conceptual basis for the modelling and analysis of such processes to which the human approach is characterised by rough approximations (Ho, 1996).

Tay and Meldrum (2000) stated four main reasons why Fuzzy logic is well suited for ramp metering

a. It can utilize incomplete or inaccurate data.
b. It can generate more smooth outputs rather than oscillatory metering rate.
c. In local design point of view, it does not require extensive system modelling
d. It is easy to tune by changing weighting factor and the parameters of membership functions.

Most conventional controllers are only as good as the system model and usually force nonlinear systems into a linear context. Fuzzy logic controller can handle non linear systems with unknown models; it has a distinctive advantage over conventional controllers for the ramp metering problem (Bogenberger, 2000). A typical fuzzy control system consists of a rule base, membership functions, and an inference procedure. A membership function is defined as the mathematical function that estimates the degree of elements membership in a fuzzy set. The rule base, defined as the set of rules in the fuzzy logic algorithm, incorporates human expertise. Since

rules are easy to define, alter, or eliminate, fuzzy logic allows simple development and modification. Implementations of fairly simple fuzzy logic ramp controllers in Seattle (Taale et al., 1996) and Amsterdam (Taylor et al, 1998) have already proven these capabilities.

2.5 Fuzzy Logic Signal Control for Intersection

Tan et al. (1996) described a fuzzy logic controller for a single junction that should mimic human intelligence. The FLC determines the time that the traffic light should stay in a certain state, before switching to the next state. The order of states is predetermined, but the controller can skip a state if there is no traffic in a certain direction. The amount of arriving and waiting vehicles area quantized into fuzzy variables, like many, medium and none. The activation of the variables in a certain situation is given by a membership function, so where there are 5 cars in the queue, this might result in an activation of 25% of 'many' and 75% of 'medium'. Fuzzy rules are used to determine if the duration of the current state should be extended. In experiments the fuzzy logic controller showed to be more flexible than fixed controllers and vehicle actuated controllers, allowing traffic to flow more smoothly, and reducing waiting time. A disadvantage of the controller seems to be its dependence on the preset quantification values for the fuzzy variables. They might cause the system to fail if the total amount of traffic varies.

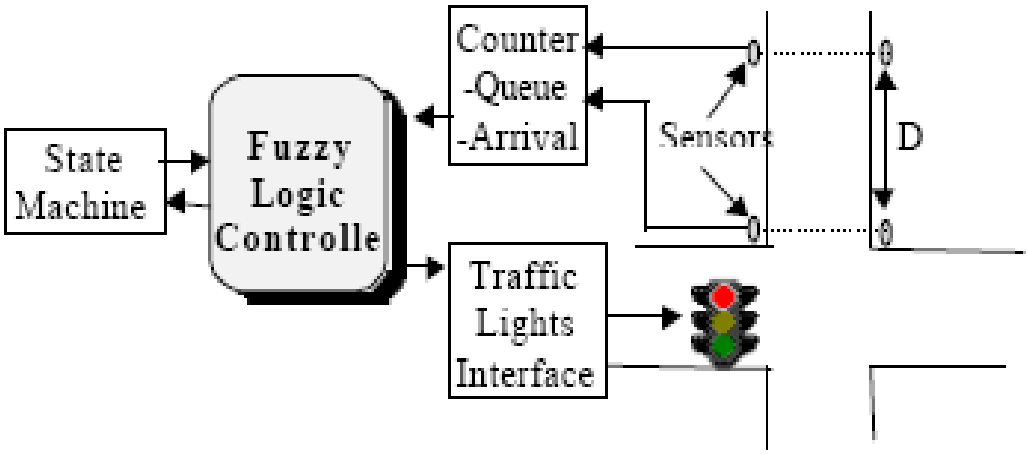


Figure 2.2 General structure of fuzzy traffic light control system (Tan et al., 1996)

Lee et al. (1995) studied the use of fuzzy logic in controlling multiple junctions. Controllers received extra information about vehicles at the previous and next junctions, and were able to promote green waves. The system outperformed a fixed controller, and was at its best in either light or heavy traffic. The controller could easily handle changes in traffic flow, but required different parameter settings for each junction.

Choi et al. (2002) also use fuzzy logic controllers, and adapted them to cope with congested traffic flow. Comparisons with fixed fuzzy-logic traffic light controllers indicated that this enhancement could lead to larger traffic flow under very crowded traffic conditions.

2.6 Existing Fuzzy Logic Control Models for Intersection

Pappis and Mamdani (1977) presented a FLC for a traffic junction of two one-way streets. The FLC uses three linguistic input variables and one linguistic output. The fuzzy 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 calculated using the FLC is the fuzzy output. This FLC was simulated at less critical intersections. Pappis's Fuzzy Logic approach reduced the average delay at an intersection of two one-way roads of various levels of traffic flow rates, when compared to with a fixed-cycle traffic controller. Pappis's model was different from Mamdai's approach; it employed one fixed set of object evaluation control rules, in which a control action is inferred from the evaluation of the resulting state of the system. Pappis's model works reasonably well when the flow rates (average number of vehicles arriving) in the two traffic streams are constant and match the traffic conditions described by the 'if-part' of fuzzy rules. However, the flow rates are functions of time and likely to span a wide range in practices; hence so is the traffic condition. A set of fixed rules is therefore inadequate to provide a comprehensive evaluation of the time-varying traffic condition and infer the optimal control actions.

Favilla et al. (1993) proposed a 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. There are both statistical and fuzzy adaptive strategies.

The FLC determines the extension time of the vehicles in the approach that has the green phase and the vehicles in the red approach. This FLC was tested for the intersection of two major avenues in the city of Sao Paulo. However the tested intersection is specific and is therefore difficult to apply adaptive strategies to other road networks.

Jamshidi et al. (1993) 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. The FLC uses 26 different fuzzy rules that are invoked every second.

Simulations are not confined to one-way road systems, but the signal program has only two states, i.e. it is restricted to two fixed directions of traffic. The controller is unable to respond to main traffic flows that change and turn.

Hoyer and Jumar (1994) proposed a FLC that focused on controlling an intersection with twelve main directional traffic flows**.** The controller operates with ten fuzzy inputs and two output variables. The chosen inputs were traffic density of different lanes and elapsed time since the last state change. Outputs were the extension time of the current phase and the selection of the next phase. Three-state or four state controls. In this way, the advantages of both control schemes may be combined. This FLC cannot choose one among many phases freely, thus more delay time may occur when traffic densities of each approach are highly uneven.

Trabia, Kaseko and Ande (1999) discussed a fuzzy logic controller for an isolated four-approach intersection with through and left turning movements. The controller was used to measure approach flows and estimate approach queues at regular intervals. Based on this, the current signal phase was terminated or extended by the 2-stage fuzzy logic controller. During the first stage, the controller estimated the traffic intensity on each approach. This intensity information was then used in the second stage to determine whether to extend or terminate the current phase.

2.7 Traffic Signal Phasing Selection

A signal phase is defined to be a unique set (or grouping) of traffic signal movements, where a movement is controlled by a number of traffic signal lights that change colour at one time (known as a signal group). When the traffic signal controller changes phase, one or more traffic movements will be stopped and one or more different movements will start (Austroads, 2002).

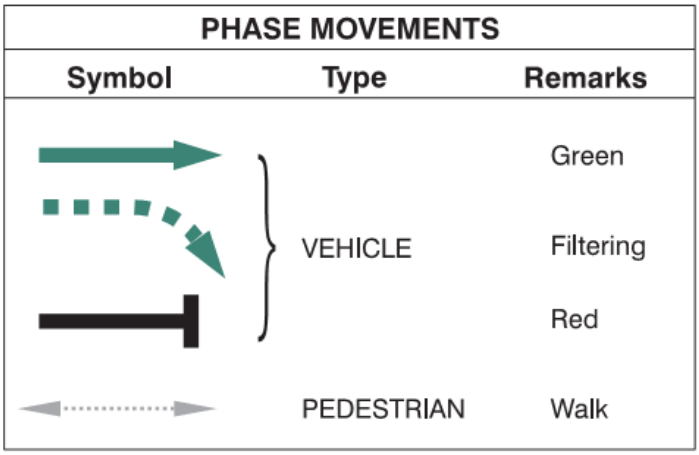


Figure 2.3 Phase movement symbols (Austroads, 2002)

2.7.1 Protected Right-Turn

A protected right-turn movement is signalled by a right-turn green arrow in the arrow phase. While the movement is protected, it may turn with no opposing through traffic i.e. the motorist does not have to consider oncoming traffic as this has been stopped (Austroads, 2007).

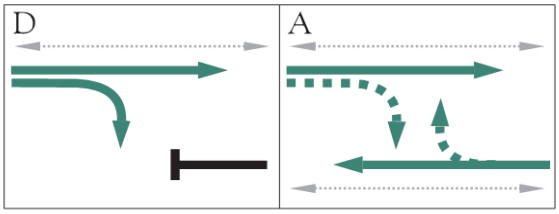
2.7.2 Permitted Right-Turn

A permitted right-turn movement is a turning movement where motorists are allowed to filter through gaps in the opposing through traffic (Austroads, 2007).

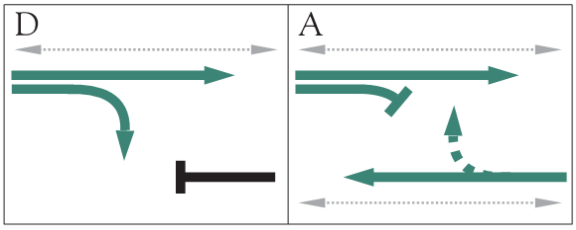
2.7.3 Leading Right-Turn

A leading right-turn phase is one where the right-turn phase precedes the main phase (where traffic flows in both directions). If the right turn is allowed to filter in the main phase then this is protected/permitted phasing (in US terminology). Otherwise, if the right turn is not allowed to filter it is a protected only phase (Austroads, 2002).

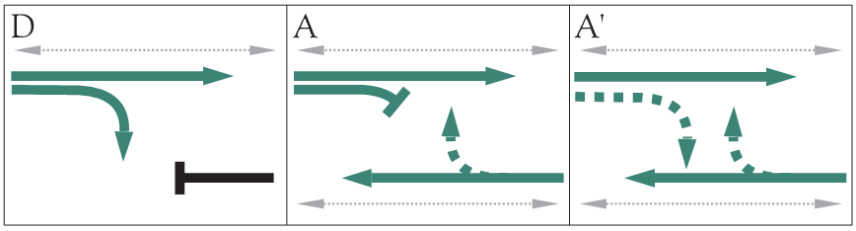
a. Right turn filtering in through phase



b. Right turn not filtering in through phase

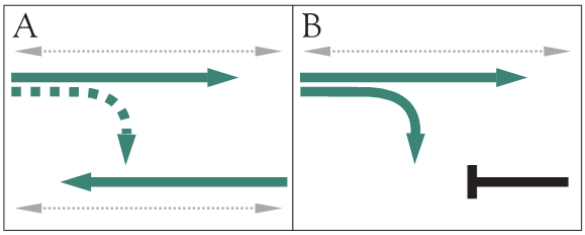


c. Right turn held on red and filtering



2.7.4 Lagging Right-Turn Phase

A lagging right-turn phase is one where the right-turn movement signal follows the main phase. If the right turn is allowed to filter in the main phase then this is permitted/protected phasing (in US terminology). Otherwise, if the right turn is not allowed to filter it is a protected only phase.



This phasing cannot be used at cross-roads where the other right turn filters. The driver of a vehicle waiting to turn right (as shown below as the problem vehicle) can see the signals facing him turn red at the end of A phase. With all other phasing arrangements, this indicates that the opposing traffic is also being stopped and the vehicle can turn in the inter-green. However, in this case, the opposing through traffic overlaps from phase A to phase B and the problem vehicle cannot assume that it is safe to turn during the inter-green. This phase is only suitable for T-junction (Austroads, 2002).

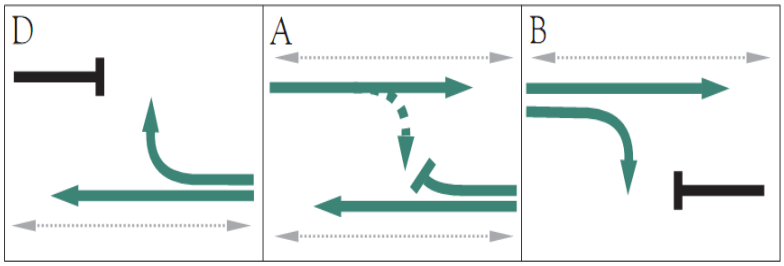
2.7.5 Split Phase

A split phase is where the movements of opposing traffic flow in totally separate phases. The right-turn movement flows at the same time as the associated through movement. Note that a green arrow signal is provided for the right-turn movement to indicate that it can flow unopposed (Austroads, 2002).

2.7.6 Protected phase

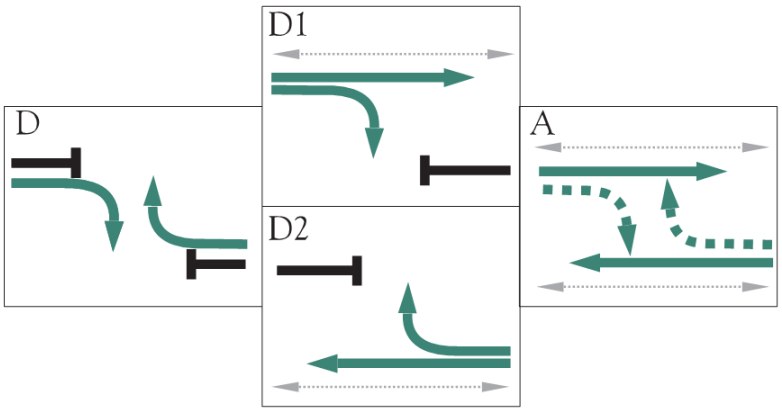
When controlled by circular signals alone, right-turning traffic will filter through gaps in the opposing traffic. Should this practice be considered unsafe due to sightline requirements, speed or other factors, the turn must be stopped by a red arrow signal.

a. Option 1 - Lead / Lag

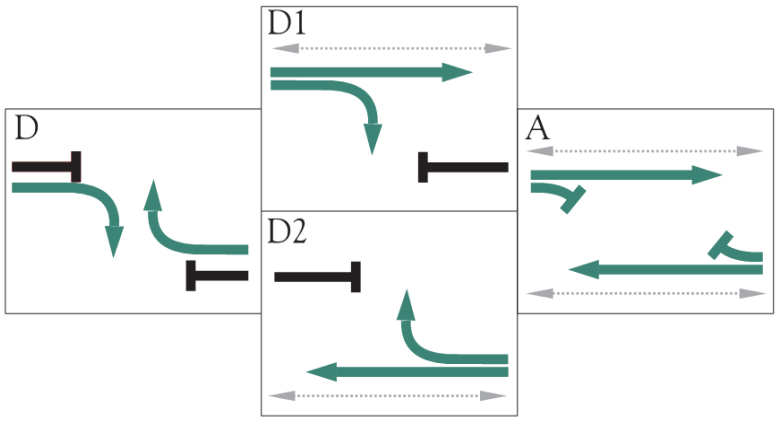


b. Option 2 - Diamond overlap

i) Right turn filtering in through phase



ii) Right turn not filtering in through phase

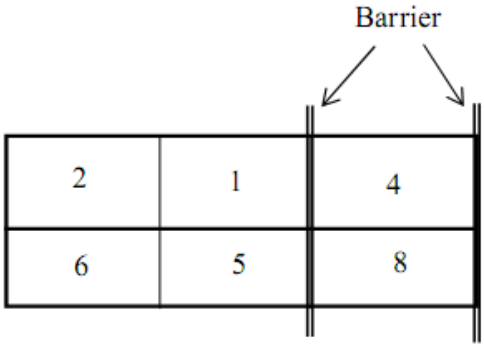


2.8 Diamond Interchange Signal Control

The basic feature of a diamond interchange is the two closely spaced intersections that connect motorway entrance and exit ramps to the arterial roads. Signalization of these two intersections is necessary when demand flows are sufficiently high. The signalization of a diamond interchange presents several major challenges. Most of these involve the heavy turning movements and the possibility of spill back queuing vehicles between the internal link connecting the two intersections (Bonneson and Lee, 2000). Diamond interchanges in USA usually follow the three-phase and four-phase scheme.

2.8.1 Three-phase plan

According to Fang (2004), three-phase plan divide the diamond interchange into two separate intersections with each having three phases. One phase is for through movements on the surface arterial roads, the second one for through movements from the internal links and the last one is for ramp movements.



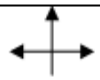
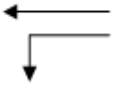
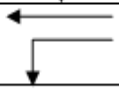
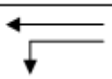
(Bonneson & Lee, 2000)

Figure 2.4 Basic Three-Phase Diamond Interchange Ring Structure

Each ramp junction is uniquely controlled by the phase timing functions in one ring as shown in Figure 2.4. From the above figure, the plan can have 5 possible signal combination pairs (2 and 6, 2 and 5, 1 and 6, 1 and 5, 4 and 8) while it requires the simultaneous termination of arterial phases before switching to the ramp phases (Fang, 2004).

Fang (2004) (cited on Munjal (1973) and Messer et al (1977)) discussed diamond interchange pre-timed signalization and proposed some phase plans. These common pre-timed diamond interchange phase plans are shown in the Table 2.1. The three-phase plan may possibly create queuing problems on the inside link if the left-turning vehicles from the ramps that enter the inside link can't be discharged by the next phase. Plan 4 in Table 2.1 could be the worst performing in terms of queue accumulation. Nevertheless, three-phase plans provide a more efficient use of time, therefore, reduce delay.

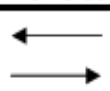
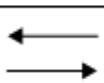
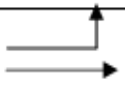
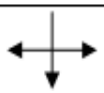
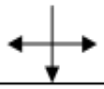
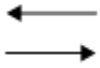
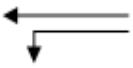
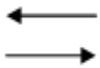
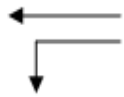
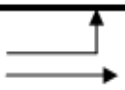
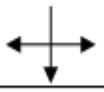
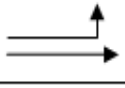
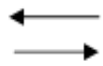
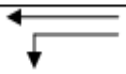
Table 2.1 Common Three-Phase Signal Plan (Messer et al, 1977)

Phase Plan		Phase Number	Left Side	Right Side
3-Phase	1 (Lead –Lead)	Phase 1		
		Phase 2		
		Phase 3		
	2 (Lag-Lead)	Phase 1		
		Phase 2		
		Phase 3		
	3 (Lead – Lag)	Phase 1		
		Phase 2		
		Phase 3		
	4 (Lag- Lag)	Phase 1		
		Phase 2		
		Phase 3		

2.8.2 Four-phase plan

Table 2.2 presents the most common four-phase pre-timed diamond interchange plans. In the four-phase with overlaps plan, vehicles entering and exiting from the internal link can happen at the same phase except that at overlaps Phase 1 and 4, through vehicles are allowed to enter the internal link while not be able to exit the link during the phase. The four-phase with overlaps strategy can possibly reduce queuing in the internal link of the interchange. However, because the plan requires four phases, longer cycle lengths are required which reduce the capacity of the intersection. The four-phase plan decreases the effective green time per cycle (g/C) ratio for the major movements and increases delay (Fang, 2004).

Table 2.2 Common Four-Phase Signal Plan (Messer et al, 1977)

Phase Plan	Phase Number	Left Side	Right Side
4-phase with overlaps	Overlap Phase 1		
	Phase 2		
	Phase 3		
	Overlap Phase 4		
	Phase 5		
	Phase 6		
4-phase	Phase 1		
	Phase 2		
	Phase 3		
	Phase 4		

2.9 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 plan 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 does not response with real-time demand fluctuation. Hence, there is a need to develop a Fuzzy Logic control for the entire diamond interchange including the on-ramp.

The literature was reviewed on various local traffic responsive ramps including fuzzy logic ramp metering model developed by Klaus Bogenberger. The fuzzy logic ramp metering reacts to local traffic conditions periodically therefore the installation of loop detectors is needed to collect real time traffic information. These detectors can be moved around but they should be able to detect the traffic information for up-stream and down-stream of the motorway and on the entrance of the on-ramp. A typical FLC does not need to be based on any traffic model however the fuzzy rules are based on human knowledge and expertise (Niittymaki, 2001).

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.

Lastly, the literature review covers the traffic signal phasing selection process with definitions of protected, permitted and leading right turn movements.

CHAPTER 3

METHODOLOGY

In this chapter, SH1 Upper Highway Diamond Interchange is used as a testing site for the development the FLDI algorithm. This algorithm could be applied to all diamond interchanges around New Zealand.

3.1 SH1 Upper Highway Diamond Interchange (FLDI algorithm developing base)

The SH1 Upper Highway Diamond Interchange is one of the most important interchanges in Northern Auckland region as it connects the SH18 with Northern SH1 motorway together to form a "Western Ring Route". When the Western Ring Route is completed in 2015, it will allow traffic to completely bypass Auckland CBD and the Harbour Bridge by linking the Southern Motorway (SH1) at Manukau City to the South-Western Motorway (SH20), the North-Western Motorway (SH16), and the Upper Harbour Highway (SH18) before rejoining the Northern Motorway (SH1) at SH1 Upper Highway Diamond Interchange.

Figure 3.1 Aerial view of SH1 Upper Highway Diamond Interchange (Google Map)

3.2 Methodology for Fuzzy Ramp System

3.2.1 Fuzzification

Fuzzification is a process that converts each numerical (analogue) input into a set of degrees of membership by membership functions. Input data could be acquired by four detectors installed at different locations:

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 is to detect the queue occupancy, which is also called queue detector. It is placed 180m from South bound on ramp entrance.
d. The check-in detector is the detector located at the ramp metering stop bar. Check-in detector is to detect the check-in occupancy. The location of the detector is 470 meters from the South-bound on-ramp entrance.

The detectors used in this project are the common single loop and advanced loop inductive sensors similar to the ones installed on the SH1 Upper Highway Diamond Interchange. Each detection interval is one minute in order to smooth the input signal and allow the system to adapt quickly to the traffic conditions. Fuzzification translates each numerical input into a set of fuzzy classes so we get different classes of speed, flow and occupancy according to the traffic quality of the mainline traffic flow. For example, the local occupancy, local flow and local speed are described by the terms "low", "medium" and "high" and the degree of activation indicates how true that class is on a scale of 0 to 1 (Bogenberger, 2000).

Table 3.1 Terms of the Fuzzy sets of inputs and outputs

Local Speed	Low	Medium	High
Local Flow	Low	Medium	High
Metering Rate	Low	Medium	High
Local Occupancy	Low	Medium	High
Downstream V/C	High		
Downstream Speed	High		
Check-In Occupancy	High		
Queue Occupancy	High		
Interchange Queue Occupancy	High		

a. 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% (Figure 3.2).
b. Local occupancy is from 0 to 30% and described as three Gaussian fuzzy set, low, medium and high, with an overlap of 50 (Figure 3.3).
c. Local flow rate is from 0 to 4000vehs/h and described as three Gaussian fuzzy set, low, medium and high, with an overlap of 50% (Figure 3.4).
d. 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 (Figure 3.5).
e. Downstream speed is from 0 to 100 km/h and activated at 50km/h and ended at 80km/h. A sigmoid function is used as the membership function (Figure 3.6).
f. Check-in occupancy is from 0 to 50% and activated from 10% to 30%. A sigmoid function is used as the membership function (Figure 3.7).
g. Queue occupancy is from 0 to 50% and activated from 10% to 30% (Figure 3.8).
h. Interchange Queue occupancy is from 0 to 90% and activated from 30% to 90%. A sigmoid function is used as the membership function (Figure 3.9).

Metering rate as the output of fuzzy algorithm has also been converted to fuzzy sets (Figure 3-9), which is from 240vehs/h to 900vehs/h and described as three triangular fuzzy set, low, medium and high, with an overlap of 50%. The metering rate is being scaled down (Figure 3-10) by using this equation

$$Scaled\ metering\ rate = \frac{metering\_rate - 900}{900 - 240} \tag{3.0}$$

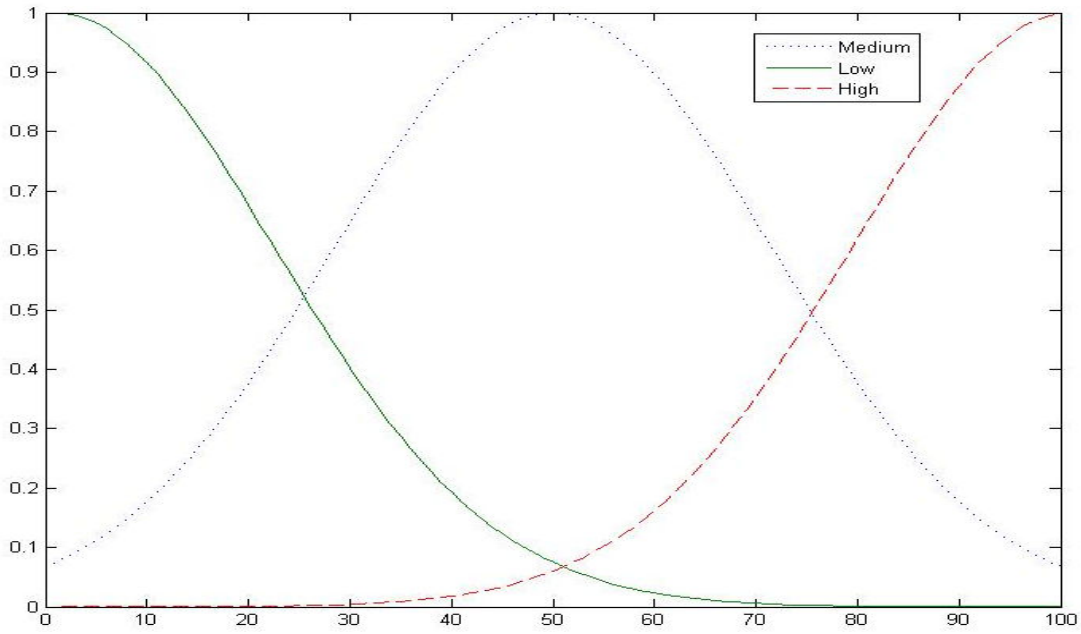


Figure 3.2 Fuzzy sets for the Local Speed (Low, Medium and High)

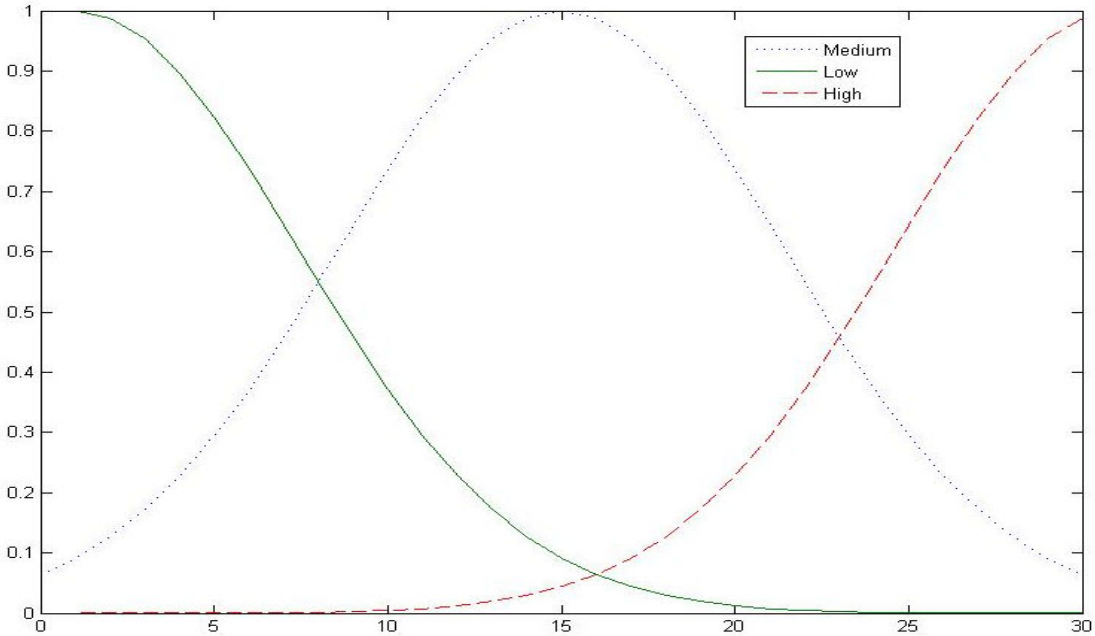


Figure 3.3 Fuzzy sets for the Local Occupancy (Low, Medium and High)

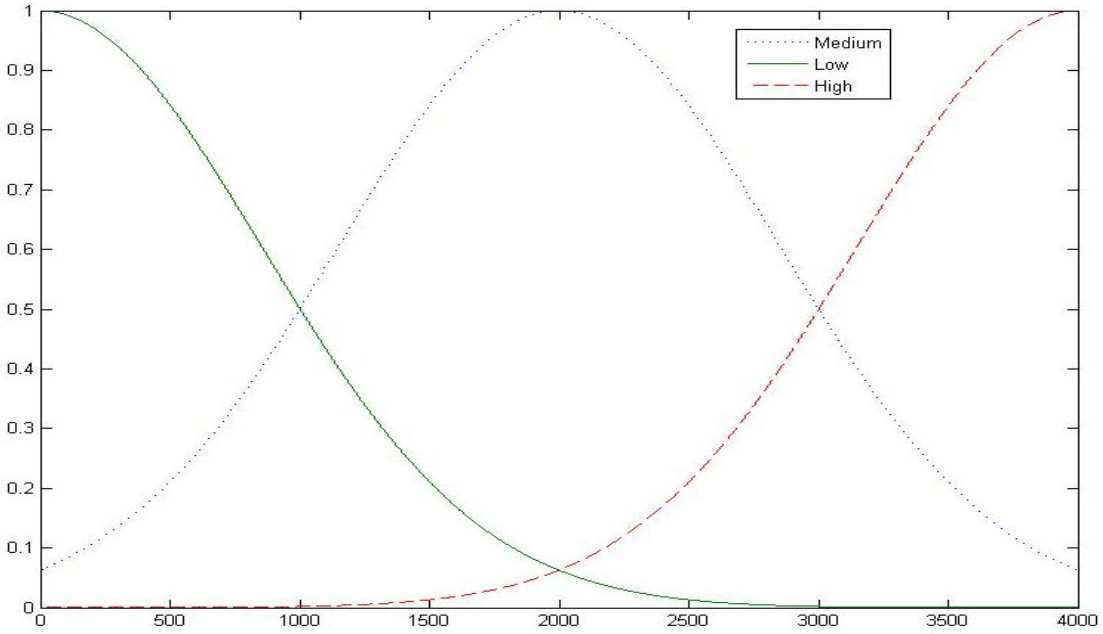


Figure 3.4 Fuzzy sets for the Local Flow Rate (Low, Medium and High)

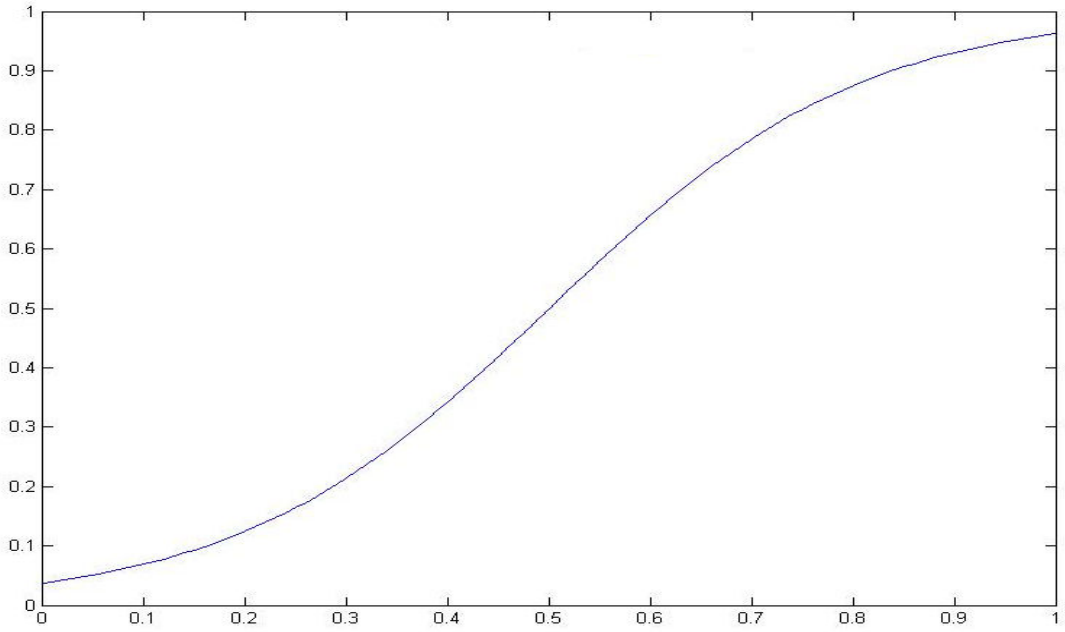


Figure 3.5 Fuzzy set for the Downstream Volume-Capacity Ratio (High)

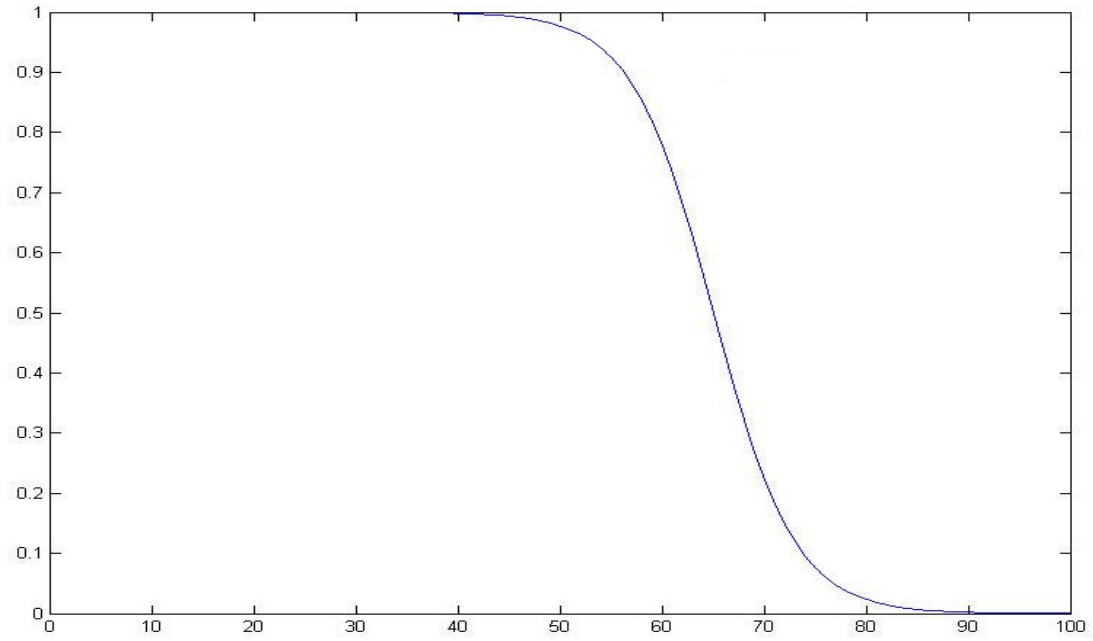


Figure 3.6 Fuzzy set for the Downstream Speed (Low)

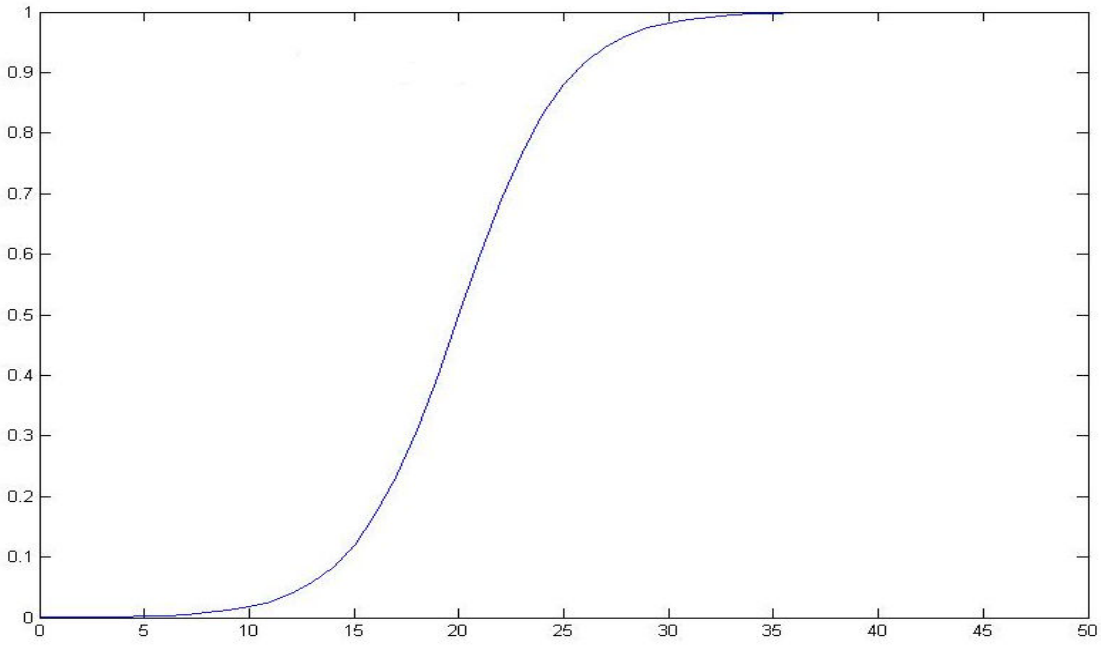


Figure 3.7 Fuzzy set for the Check-in Occupancy (High)

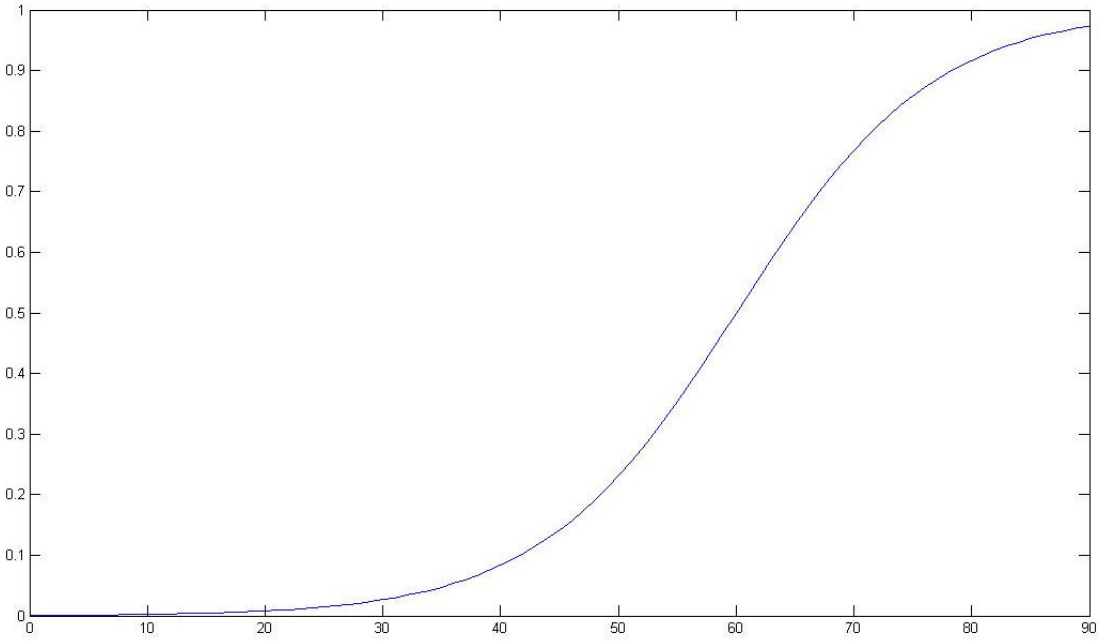


Figure 3.8 Fuzzy set for the Queue Occupancy (High)

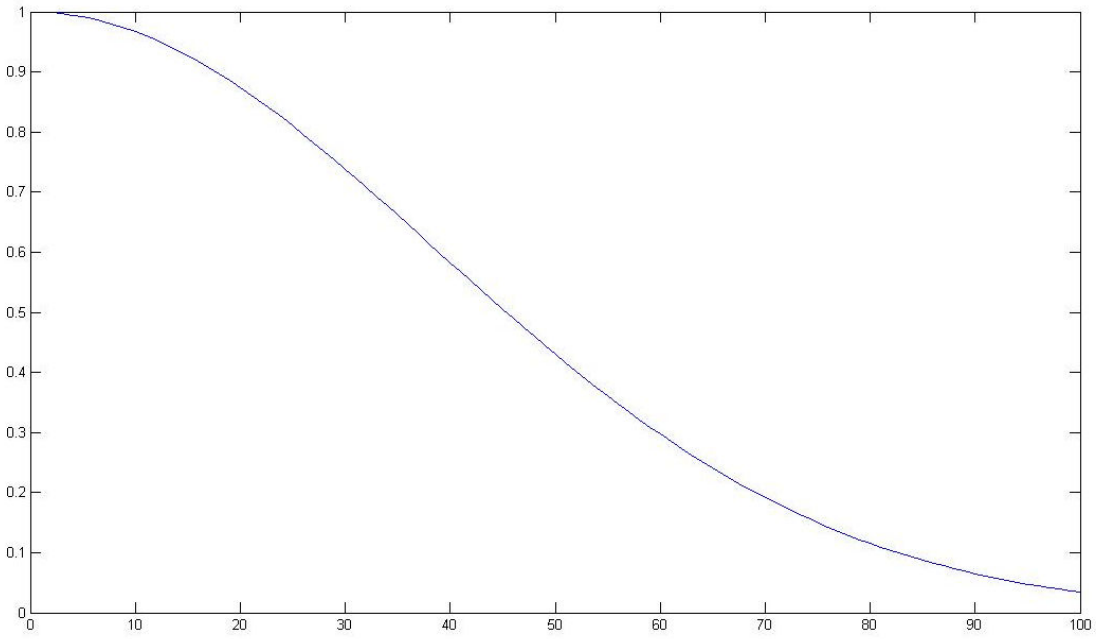


Figure 3.9 Fuzzy set for the Interchange Queue Occupancy (High)

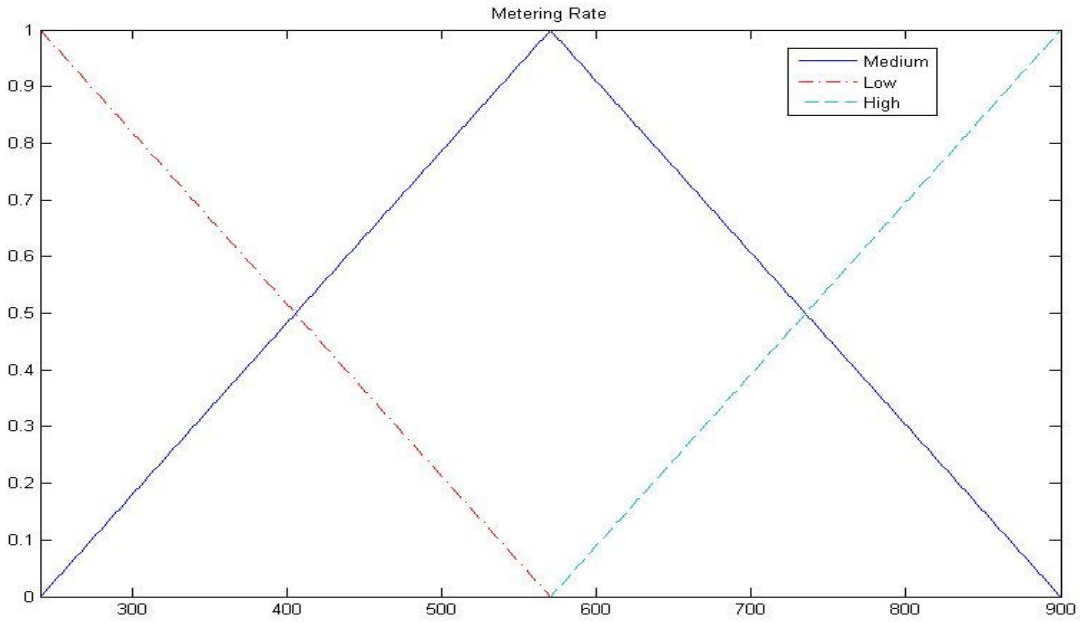


Figure 3.10 Fuzzy sets for the metering rates (Low, Medium and High)

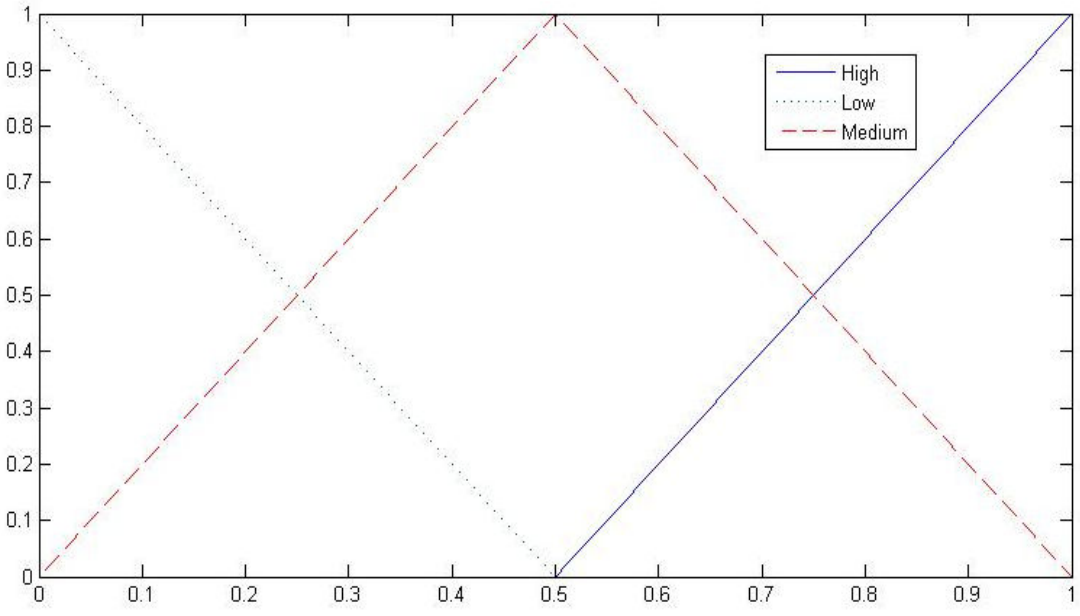


Figure 3.11 Fuzzy sets for the scaled metering rates (Low, Medium and High)

3.2.2 Inference

Fuzzy inference is the process of formulating the mapping from a given input to an output using fuzzy logic. The mapping then provides a basis from which decisions can be made, or patterns discerned. Fuzzy sets and fuzzy operators are the subjects and verbs of fuzzy logic. These if-then rule statements are used to formulate the conditional statements that comprise fuzzy logic. 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 3.2). 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).

Table 3.2 Rule base for Fuzzy Ramp Metering

RULE	RULE CONDTION		RULE OUTCOME	RULE WEIGHT
	IF	AND	THEN	
1	Local occupancy is LOW		Metering rate is HIGH	**1.5**
2	Local occupancy is MEDIUM		Metering rate is MEDIUM	**1.5**
3	Local occupancy Is HIGH		Metering rate is LOW	**2.0**
4	Local speed is LOW	Local flow is HIGH	Metering rate is LOW	**2.0**
5	Local speed is MEDIUM	Local occupancy is HIGH	Metering rate is MEDIUM	**1.0**
6	Local speed is MEDIUM	Local occupancy is LOW	Metering rate is HIGH	**1.0**
7	Local speed is HIGH	Local flow Is LOW	Metering rate is HIGH	**1.0**
8	Downstream speed is VERY LOW	Downstream v/c is VERY HIGH	Metering rate is LOW	**3.0**
9	Check-in occupancy is VERY HIGH	Queue occupancy is VERY HIGH	Metering rate is HIGH	**3.0**
10	Interchange Queue Occupancy is HIGH	Queue Occupancy Is HIGH	Metering rate is HIGH	**3.0**

The main point of setting the first three rules (Rule 1, 2 and 3) is 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 can be seen as a prediction of the downstream bottleneck behaviour. The v/c ratio is a measure of the degree of saturation. It is a performance measure used in analysis of signalized ramp.

Rules 1 and 6 have relatively higher weightings, the reason for this is 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 fuzzy logic control has to extend the metering rate to the maximum, to avoid the long queue on the ramp as well as preventing the traffic spill back to the connected intersection. The last rule (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 detectors are placed on various places on the connected intersections.

3.2.3 Defuzzification

For a fuzzy system whose final output needs to be in a crisp (non-fuzzy) form (specific metering rate, cycle time), a conversion from the final combined fuzzy conclusion into a crisp one is needed. This step is called defuzzification. For the defuzzification process in the ramp metering algorithm, a continuous centroid method is used, this helps to suppress parameter variations and stochastic disturbances, and the fuzzy logic algorithm produces an appropriate output given uncertain or incomplete information, e.g. from the loop detectors. The crisp output of the controller is the metering rate (cycle time) of the controlled on-ramp.

The equation for continuous centroid method (Pappis and Mamdani, 1977):

$$\frac{\sum_{i=1}^{N} w_i c_i I_i}{\sum_{i=1}^{N} w_i I_i} \tag{3.1}$$

Where:

w_i is the centroid and c_i is the area of the output class of the i^{th} rule

N is the number of output classes

I_i is the area of the i^{th} output class

3.3 Phase Selection Logic for SH1 Upper Highway Diamond Interchange

The State Highway 1 Upper Highway Interchange has a special phase selection logics. The logic is specially designed and optimized for this diamond interchange by Peter Evans, Senior Traffic System Engineer at Transit New Zealand.

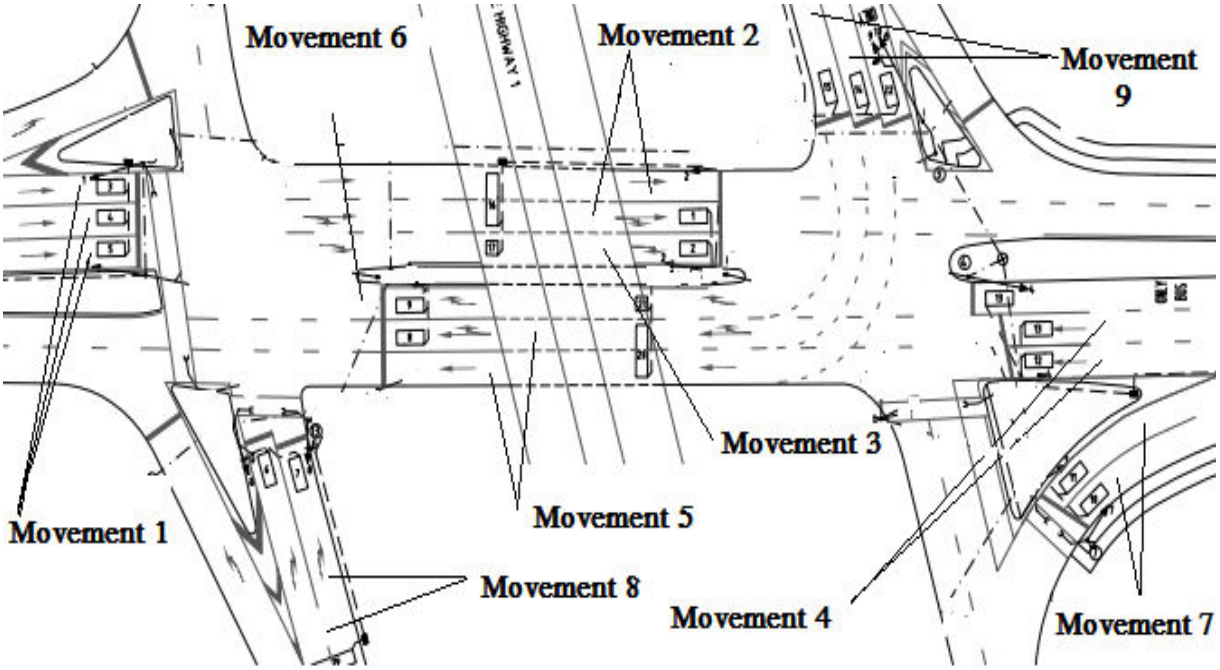


Figure 3.12 Geometric layout of SH1 Upper Highway Diamond Interchange (Transit NZ)

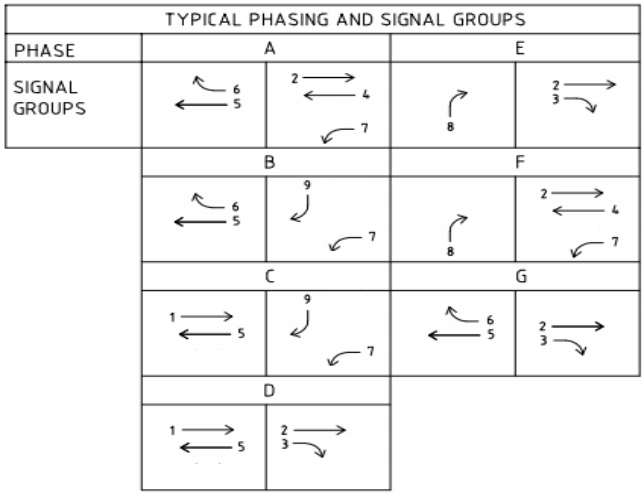


Figure 3.13 Phasing and signal groups for the interchange (Transit NZ)

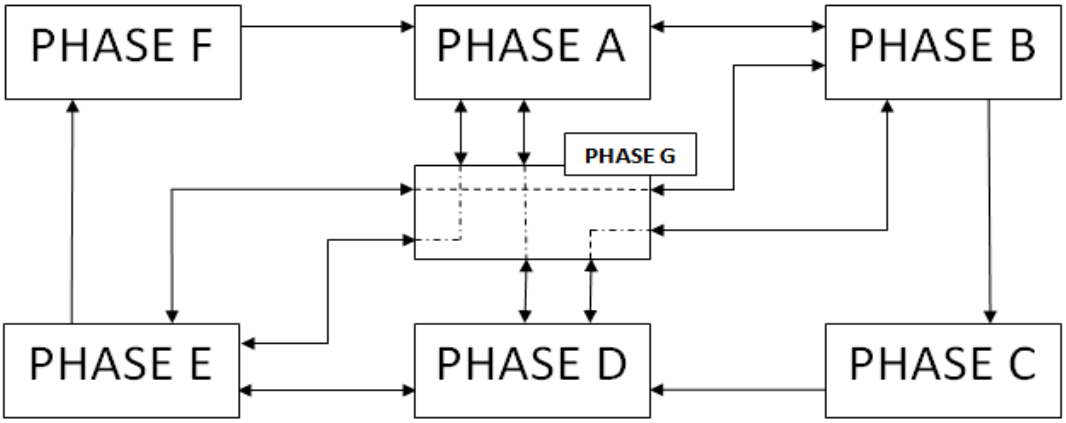


Figure 3.14 Phasing and logic

i) Starting from Phase A (running)

Phase A → Phase B (if there are cars waiting on any of detector 14, 15, 22, 24) **→ Phase C** (if there are cars waiting on any of detector 3, 4, 5, 18) **→ Phase D** (no condition)

Phase A → Phase G (if there are cars waiting on any detector 6, 7, 19, P3, P4 and there is no car waiting on any of detector 14, 15, 22, 24) **→ Phase D** (no condition required)

Phase A → Phase G (if there are cars waiting on 6, 7, 19, P3, P4 and there is no car waiting on any of detector 1, 2, 3, 4, 5, 14, 15, 17, 18, 22, 24) **→ Phase E** (no condition)

i) Starting from Phase B (running)

Phase B → Phase A (if there are cars on 12, 13, 16, 23, P1, P2 and there is no car on 1, 2, 3, 4, 5, 6, 7, 17, 18, 19, P3, P4)

Phase B → Phase G (if there are cars on 1, 2, 17, and no car on 3, 4, 5, 18) **→ Phase D** (no condition)

Phase B → Phase G (if there are cars on 6, 7, 19, P3, P4 and no car on 1, 2, 3, 4, 5, 17, 18) **→ Phase E** (no condition)

ii) Starting from Phase C (running)

Phase C → Phase D (no condition)

iii) Starting from Phase D (running)

Phase D → Phase G (if there are cars on 8, 9, 10, 11, 12, 13, 21, 23, P1, P2 and no car on 6, 7, 19, P3) → **Phase A**

Phase D → Phase E (if there are cars on 6, 7, 19, P3)

iv) Starting from Phase E (running)

Phase E → Phase F (if there are cars on 10, 11, 12, 13, 23) → **Phase A** (no condition)

Phase E → Phase G (if there are cars on 8, 9, 21, P1, P2 and there is no car on 10, 11, 12, 13, 23) → **Phase A** (no condition)

Phase E → Phase G (if there are cars on 14, 15, 22, 24 and there is no car on 8, 9, 10, 11, 12, 13, 21, 23, P1, P2) → **Phase B**

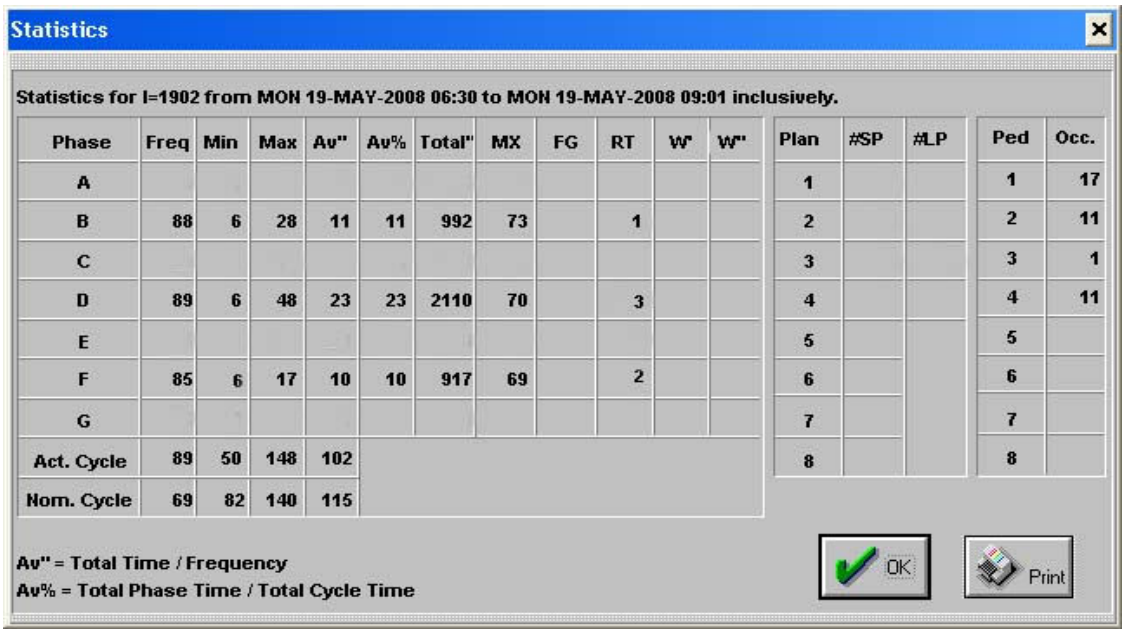
Phase E → Phase D (if there are cars on 3, 4, 5, 18, 20 and there is no car on 8, 9, 10, 11, 12, 13, 14, 15, 21, 22, 23, 24, P1, P2)

v) G Phase Logic

G Phase is a transitional phase providing for mid block clearance when phases are skipped during a signal cycle. G phase effectively provides a flexible early cut off period better suited for congested. A demand for G phase will simultaneously lodge a call for another phase as defined by the phasing and logic diagram (Figure 3-13). Once G phase is entered, exit must be via the predefined path. Paths are shown as dotted lines through G phase in the phasing and logic diagram. All calls for G phase are to be cancelled when the demand conditions are no longer met.

vi) Phasing operation during peak hours

For SH1 Upper Highway Diamond Interchange traffic actuated controller, the green time for each phase is determined by the volume of traffic on the corresponding street and may vary from cycle to cycle. A maximum and minimum green time are predetermined and set within the controller. The minimum initial green time (6 seconds) is set to be adequate for the number of vehicles waiting between the stop line and detector. Each additional vehicle which triggers the detector during green phase calls for a vehicle interval extension. The Green time extension ranges are based on the report obtained from the IDM report for SH1 Upper Highway Diamond Interchange on 19/05/2008 (Figure 3.15).



Phase	Freq	Min	Max	Av"	Av%	Total"	MX	FG	RT	W'	W"
A											
B	88	6	28	11	11	992	73		1		
C											
D	89	6	48	23	23	2110	70		3		
E											
F	85	6	17	10	10	917	69		2		
G											
Act. Cycle	89	50	148	102							
Nom. Cycle	69	82	140	115							

Plan	#SP	#LP
1		
2		
3		
4		
5		
6		
7		
8		

Ped	Occ.
1	17
2	11
3	1
4	11
5	
6	
7	
8	

Figure 3.15 SCATS IDM Phase run report from 6:30-9:01 on 15/07/2008 (ATMS)

According to Figure 3.15, during peak hours from 6:30-9:30AM on weekdays only 3 phases are available to run instead of 7 phases like other periods during the day. The phase operate in a fixed sequence starting from Phase B then Phase F and ends with Phase D. Each phase has a minimum green time of 6 seconds with different maximum green times (28 seconds for Phase B, 48 seconds for Phase D and 17 seconds for Phase F). For all other periods, the diamond interchanges operate based on the mentioned Phase Logic above (Figure 3.14).

3.4 Methodology for Fuzzy Logic Phase Timing

Vehicle detectors are installed on “upstream-line” and “stop-line”. The number of approaching vehicles for each approach during given time interval can be estimated using the detectors. Fuzzy variables considered are arriving vehicles, queuing vehicles and extension. Extension is the output of the fuzzy variable, it is the extension needed for the green light on the arrival side. In this model, the minimum green time is given for arriving traffic, at the end of green period if the impulses are receiving from detectors and the maximum queue length of other flows that are waiting in red signal. After that the extension process is stopped and the next phase is selected considering according to Figure 3.16.

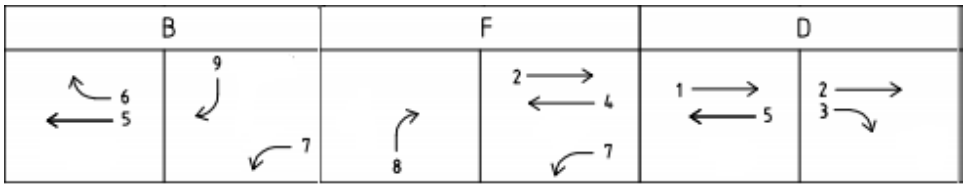


Figure 3.16 Phase plan for SH1 Upper Highway Diamond Interchange during peak hours

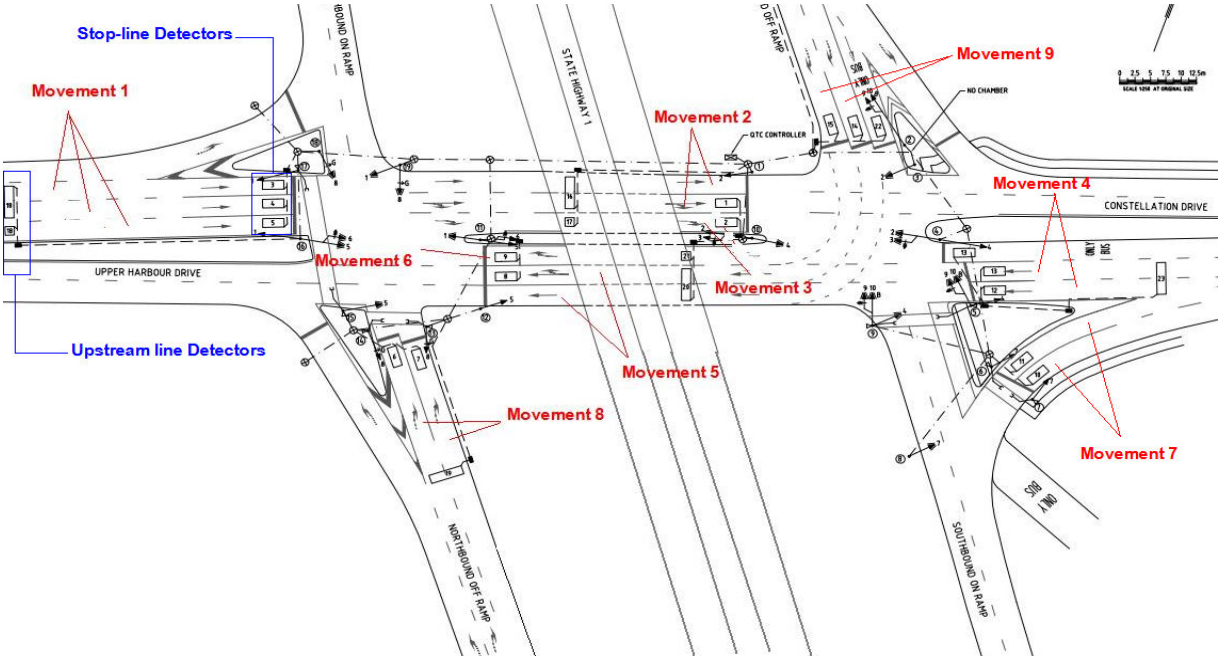


Figure 3.17 Movements and detectors layout

3.4.1 Fuzzy Logic Phase Timing for Phase B

In the SH1 Upper Highway Diamond Interchange, Fuzzy Logic Phase Timing is designed to calculate the Green Time Extension for each individual phase. Initially the traffic controller will run Phase B with minimum green time of 6 seconds. Then fuzzy controller determines whether to extend or terminate the current green phase after a minimum green time has been elapsed. The minimum green time for Phase B is 6 seconds. If Phase B is running (green) then this would be considered as the arrival side while the Phase D and F are queue sides. Arrival is the total amount of vehicles on movements 5, 6, 7, 9. Queue is the total amount of vehicles on movements 1, 2, 3, 4, 8 (Figure 3.16). The extension range is based on the SH1 Upper Highway Diamond Interchange Actuated Traffic Plan (Figure 3.15).

Table 3.3 Fuzzy rules and its weighting for Phase B

Rule number	IF (Arrival)	AND (Queue)	THEN (Extension)	Rule 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**

The graphical representations of the membership functions of the variables are presented in Figure 3.18 and 3.19. It can be observed that the y-axis is the degree of the membership of each of the fuzzy variable and the x-axis is the quantity of vehicles. For the output fuzzy variable (Figure 3.20) is the length of time to be extended in seconds. From Figure 3.19, it can be observed that 28 vehicles have been assigned as “Many” fuzzy subsets in this simulation which have a full membership. For the output fuzzy variable (Figure 3.20), a strong “Long” fuzzy

subset with of membership of "1" would be in the region of 42 seconds whereas a strong "Medium" fuzzy subset would be in the region of 20 seconds. The configuration of these membership functions is done according to observation of the system and the operation of the SH1 Upper Highway Diamond Interchange daily operation. However, the width and the centre of the membership functions of these fuzzy subsets can be easily changed and configured according to different traffic conditions. For example, if the interchange is too congested, the number of vehicles in the fuzzy subset "Medium" and "Many" is needed to be increased. Meanwhile, for a less congested interchange the width of the membership functions can be reduced. In fuzzy logic phase timing control in order to smooth the transition from one control action to another the fuzzy subsets need to be overlapped. If there is no overlapping in the fuzzy subsets then the control action would resemble step-like control. The green time extension in seconds for the output of fuzzy logic phase timing controller. These timings are generated automatically by the fuzzy logic controller and they are dependent on the settings of the membership functions and rules.

The general structure of a fuzzy logic phase timing control system is illustrated as in Figure 3.16. In each movement there are two detectors (electromagnetic sensors) placed on the road for each lane. The first detector behind each traffic lights (stop-line detector) counts the number of cars passing the traffic lights, and the second detector which is located behind the first detector counts the number of cars coming to the intersection. The number of cars queuing in front of a particular traffic light is determined by calculating the difference of the reading between the two detectors. Based on the distance between the upstream-line and stop-line detectors (Figure 3.17) the maximum amount of cars for each movement can be calculated by these two equations

$$A_L = \frac{D}{L_1 + L_2} \tag{3.2}$$

$$A_M = A_L \times N \tag{3.3}$$

Where:

A_L is the total amount of cars per lane

A_M is the total amount of cars per movement

D is the distance between the Up-stream and Stop-line detector of the lane (m)

L_1 is the average size of a vehicle (car), in this case is 4m (defined by Transit NZ)

L_2 is the acceptance gap between two vehicles, in this case is 0.5m (defined by Transit NZ)

N is the number of lanes

Fuzzy modelling includes different inference process. Mamdani (max–min)'s inference method is used in Fuzzy Logic Phase Timing model. In max–min process, the minimum membership values of the used rules outputs are selected (i.e. the minimum membership value of each input sub-set), then maximum of the minimums are determined (i.e. the maximum membership value of each output sub-set is selected). This process is applied to all valid rules and a triangular shape is obtained.

The membership functions for the arriving vehicles (A) at the approach having green phase are *few = 0 to 14, small = 0 to 28, medium = 14 to 42* and *many = 28 to 42*.

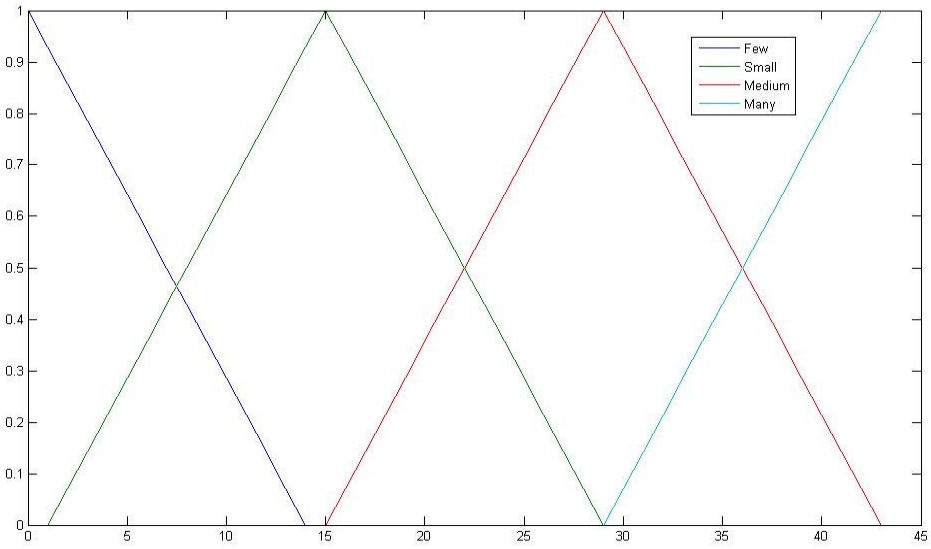


Figure 3.18 Phase B Arrival memberships

The membership functions for the queuing vehicles (Q) at the next approach having red phase are *few = 0 to 19, small = 0 to 38, medium = 19 to 38* and *many = 38 to 57.*

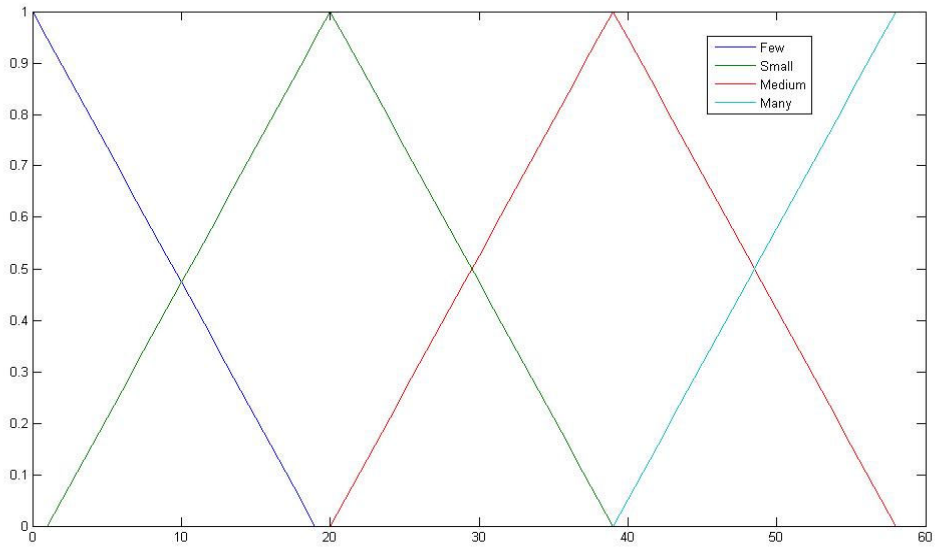


Figure 3.19 Phase B Queue memberships

The membership functions for output fuzzy variable extension (EXT) are *very short = 0 to 10 sec, short = 0 to 20 sec, medium = 10 to 30 sec* and *long = 20 to 30 sec.*

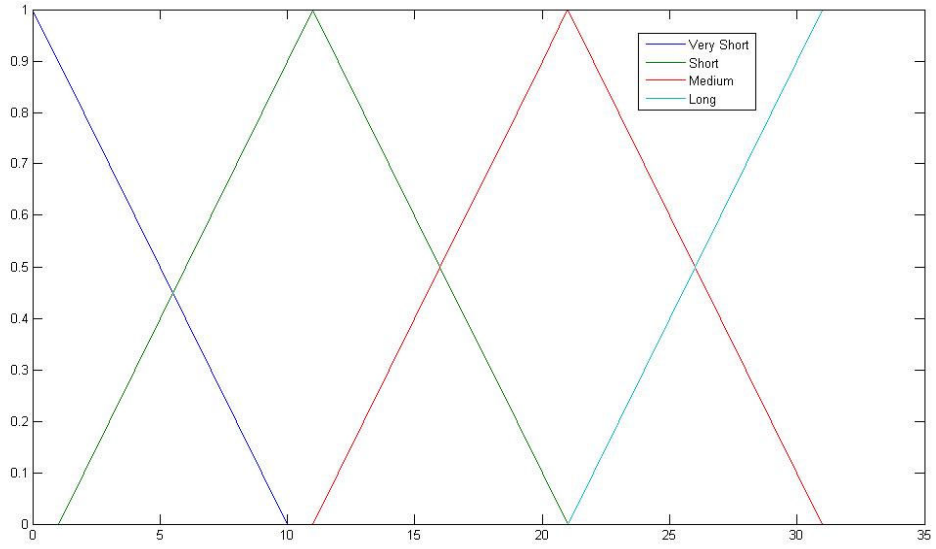


Figure 3.20 Phase B Fuzzy variable extension

A continuous centroid method is used for the defuzzification process, this helps to suppress parameter variations and stochastic disturbances, and the fuzzy logic algorithm produces an appropriate output given uncertain or incomplete information, e.g. from the loop detectors. The crisp output of the controller is the variable extension green time of the controlled traffic lights.

3.4.2 Fuzzy Logic Phase Timing for Phase D

Similar to Fuzzy Logic Phase Timing for D Phase, If Phase D is running (green) then this would be considered as the arrival side while the Phase B and F are queue sides. Arrival is the total amount of vehicles on movements 1 and 5 (since 2 and 3 are linked to 1). Queue is the total amount of vehicles on movements 4, 6, 7, 8 (Figure 3.16). The green time extension range is also based on the SH1 Upper Highway Diamond Interchange Actuated Traffic Plan (Figure 3.16). The fuzzy rules and rule's weights are similar to Fuzzy Logic Phase Timing for B Phase (16 rules)

The membership functions for the arriving vehicles (A) at the approach having green phase are few = 0 to 19, small = 0 to 38, medium = 19 to 38 and many = 38 to 57.

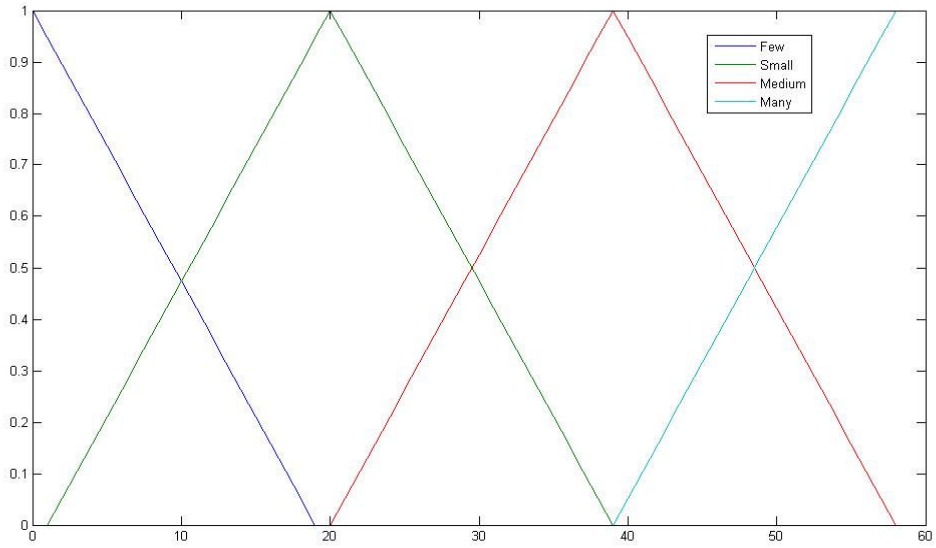


Figure 3.21 Phase D Arrival memberships

The membership functions for the queuing vehicles (Q) at the next approach having red phase are *few = 0 to 24, small = 0 to 48, medium = 24 to 72* and *many = 48 to 72.*

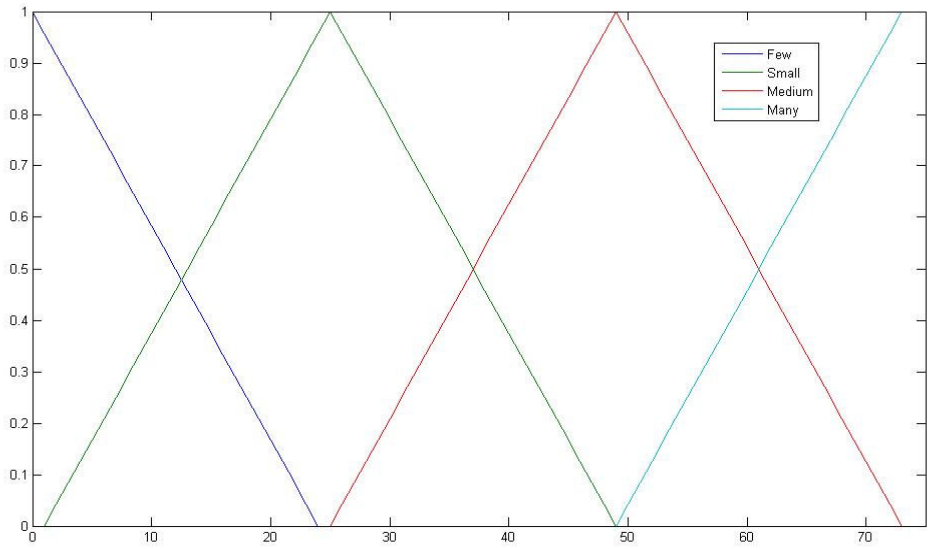


Figure 3.22 Phase D Queue memberships

The membership functions for output fuzzy variable extension (EXT) are *very short = 0 to 16 sec, short = 0 to 32 sec, medium = 16 to 48 sec* and *long = 32 to 48 sec*.

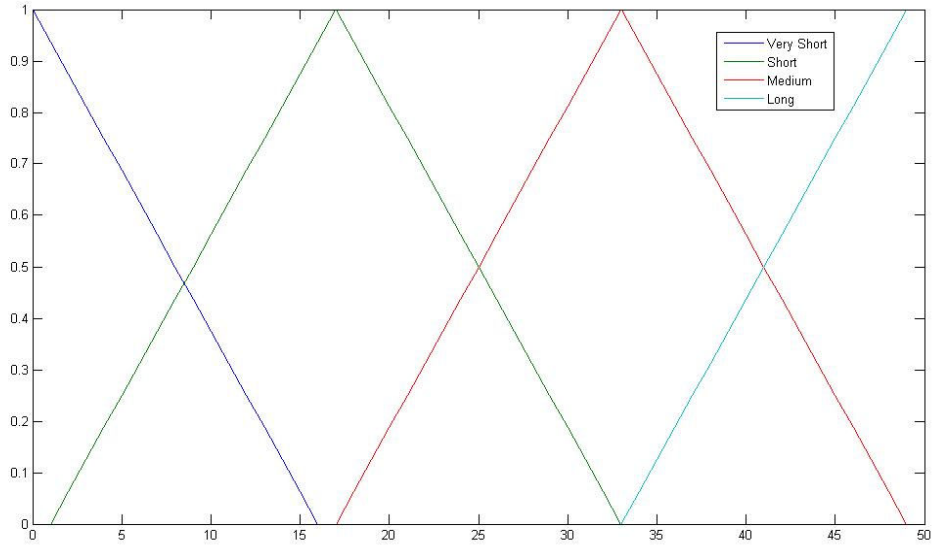


Figure 3.23 Phase D Fuzzy variable extension

In the fuzzy logic controller once the appropriate rules are fired, the degree of membership of the output fuzzy variable i.e., Extension time, is determined by encoding the fuzzy Arrival and Queue subsets. Similar to Fuzzy D Phase Timing, the max-min implication technique is used. Once the membership degree of each output fuzzy variable is determined, all of the rules that are being fired are then combined and the actual crisp output is obtained through defuzzification. Continuous centroid method is also employed for this fuzzy logic algorithm defuzzification process. The crisp output of the controller is the variable extension green time of the controlled traffic lights for the movements (associated with Phase D).

3.4.3 Fuzzy Logic Phase Timing for Phase F

Similar to Fuzzy Logic Phase Timing for Phase D and B, the traffic controller will run Phase F after the completion of Phase B with minimum green time of 8 seconds. If Phase F is running (green) then this would be considered as the arrival side while the Phase D and B are queue sides. Arrival is the total amount of vehicles on movements 2, 4, 7, 8. Queue is the total amount of vehicles on movements 1, 5, 6, 9 (Figure 3.16). The fuzzy rules and its weighting are similar to Phase B (Figure 3-2). The membership functions for the arriving vehicles (A) at the approach having green phase are *few = 0 to 19*, *small = 0 to 38*, *medium = 19 to 38* and *many = 38 to 57*.

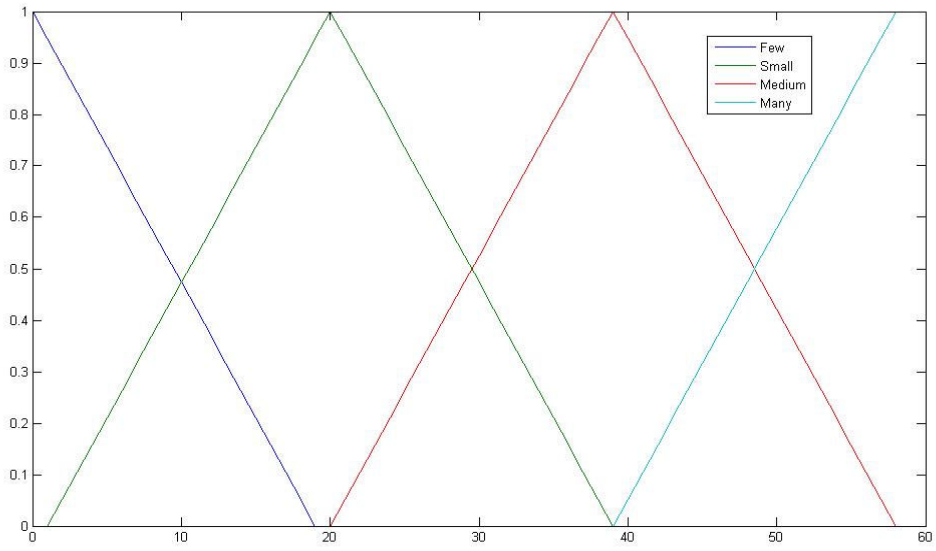


Figure 3.24 Phase F Arrival memberships

The membership functions for the queuing vehicles (Q) at the next approach having red phase are *few = 0 to 24*, *small = 0 to 48, medium = 24 to 72* and *many = 48 to 72*.

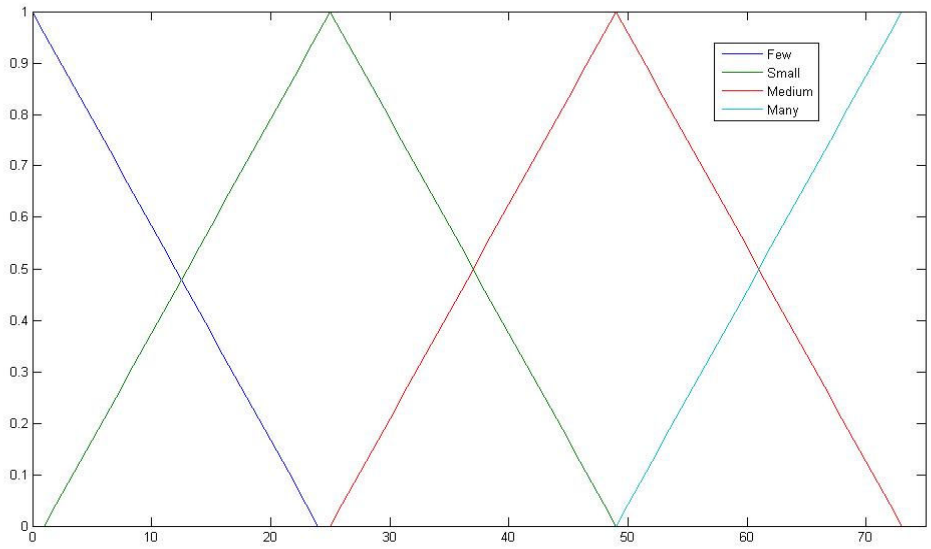


Figure 3.25 Phase F Queue memberships

The membership functions for output fuzzy variable extension (EXT) are *very short = 0 to 6 sec*, *short = 0 to 12 sec, medium = 6 to 18 sec* and *long = 12 to 18 sec.*

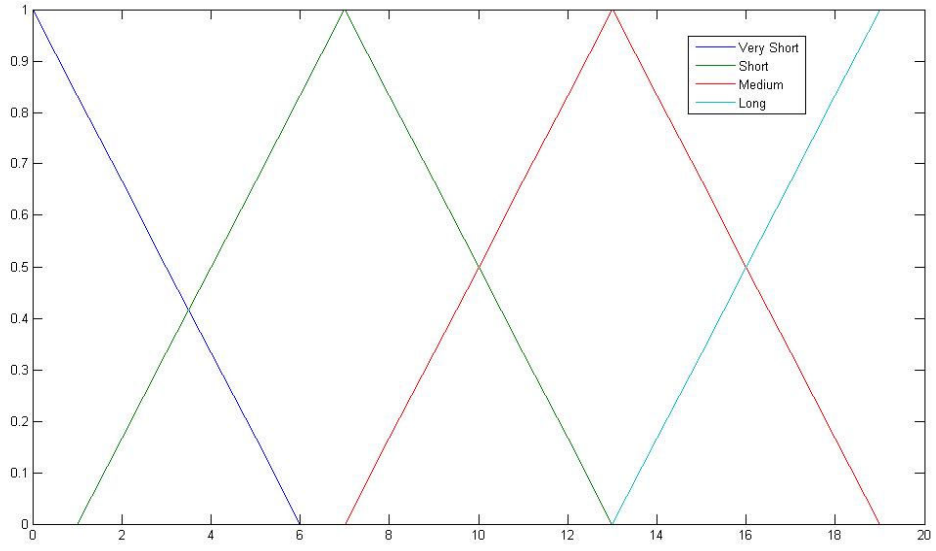


Figure 3.26 Phase F Fuzzy variable extension

Table 3.4 Fuzzy Green Time Extension for Phase D, B and F

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 Fuzzy Green Time Extension		***18.52 sec***	Average Fuzzy Green Time Extension		***12.69 sec***	Average Fuzzy Green Time Extension		***6.28 sec***

These traffic demands are generated randomly by AIMSUN to test the Fuzzy Logic Phase Timing algorithm reaction to various demands scenarios. There are 90 scenarios (30 for each Phase) which cover most of the traffic conditions from very low to extremely high traffic demands. Overall, the fuzzy controller adapts well to the changes in traffic conditions e.g. when Total Arrival = 7 and Total Queue is 47 the Green Time Extension is 3.82 seconds (on top of the minimum green time).

CHAPTER 4

ALGORITHM IMPLEMENTATION

4.1 Basic Parameters for Implementing FLDI

To implement the FLDI method developed in Chapter 3, it is important to determine the values of some parameters such as number of detectors used for each movement, distance of between the up-stream and stop-line detectors.

As stated in section 3.4.1 the amount of vehicles in a queue can be calculated using the equations 3.2 and 3.3. The counts provide the inputs for the Fuzzy Phase Timing to generate the appropriate green time extension for the current phase based on the local traffic condition. By considering different queue length at the stop-line, the following parameters are determined in order to implement the FLDI algorithm:

a) The distance between the up-stream and stop-line detectors depends on the type of the road e.g. if the road is a main stream one then the distance between the two detectors should be longer than the non main stream roads.

b) The detectors detection interval should be identical to the minimum green phase time which means if the minimum green time is 6 seconds then the detection interval is also 6 seconds. However, the detection interval should be in the sensible range as if the interval is too long or too short it will cause the in appropriate green time extension generated by FLDI as the inputs may be out of the working range.

c) Each lane should have two detectors and it is recommended that each lane's detectors should be isolated from the others as sometimes vehicles may change lane inside the detection zone.

d) During the Arrival vehicles count only one detector in each lane do the counting. If the movements are linked to each other, in other words if the movements are on a straight line and they are in the same phase then only one detector from one movement do the counting.

4.2 Detectors Setup

Figure 4.1 presents a detectors placement layout for implementing the DP algorithm. A diamond interchange has 8 lane groups including North-bound, South-bound, East-bound external, West-bound external, East-bound internal, West-bound internal, East-bound right turning and West-bound right turning. Detectors are located upstream some distance, different for all lane groups. These detectors should have the capabilities of measuring the number of vehicles passing through and their speeds.

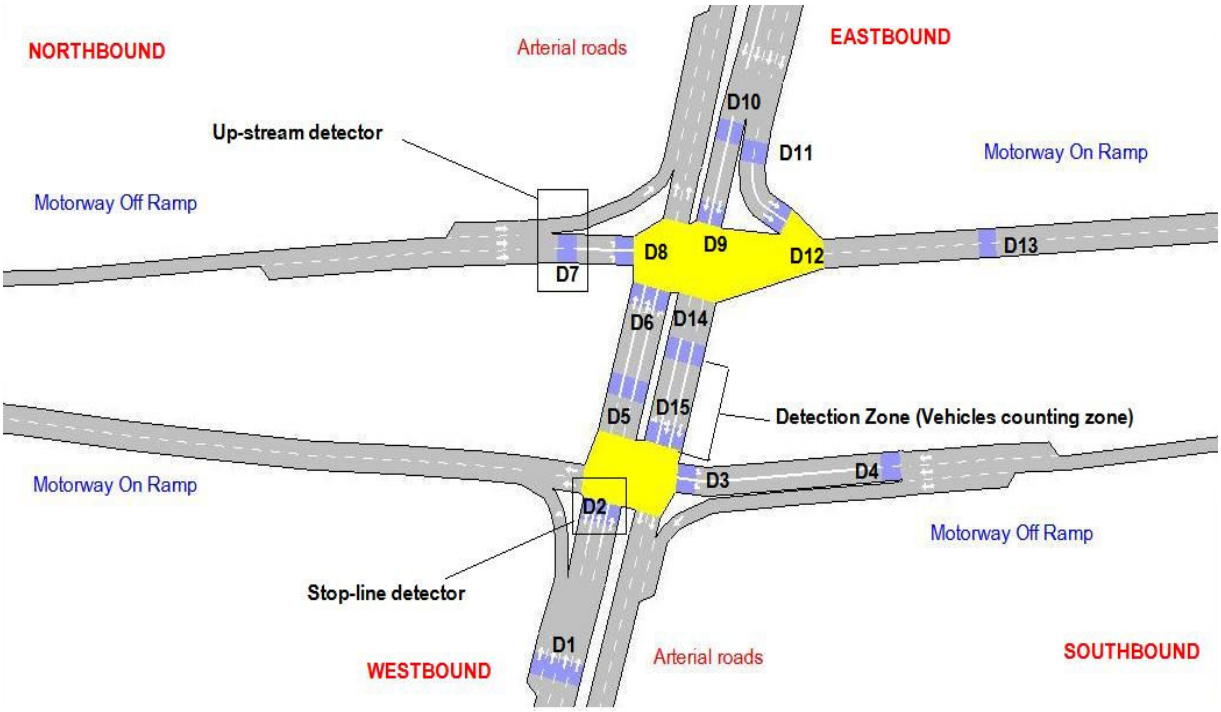


Figure 4.1 Diamond Interchanges Detectors Placement Layout

4.3 FLDI's Modules and Integration

The FLDI algorithm is made of three main modules; they are Phase Selection Module (Fixed and Logic), Fuzzy Phase Timing and Fuzzy Ramp Metering (Figure 4.2). These modules are linked together by the main algorithm; the reason why the modules are designed separately is to make the FLDI similar to an open system which means other algorithms such as genetic fuzzy or Matsuoka oscillators controls can be integrated into FLDI easily.

Moreover, different diamond interchanges have different phase plans therefore the Phase Selection Module is designed to work with both cases; Phase Logic Selection and Fixed Phase Selection.

The Phase Selection Module is similar to the state machine which controls the sequence of states that the fuzzy logic traffic controller should cycle through. There is one state for each phase of the traffic light. There is one default state which takes place when no incoming traffic is detected. This default state corresponds to the green time for specific phase approach, usually to the main stream approach. In the Phase Logic, a phase can be skipped if there is no vehicle queues for the corresponding approach.

The Fuzzy Ramp Metering module is carefully designed so it can takes the operation of the connected intersections into account before generating the ramp metering rate. The Interchange Queue Occupancy Membership will prevent the spill back phenomenon when there are too many vehicles waiting to get onto the ramp from the connected intersection and also when the queue on the ramp is too long which may lead to the congestion of the connected intersection (If Interchange Queue Occupancy is HIGH and Ramp Queue Occupancy is HIGH then the Ramp Metering rate is set to HIGH). The Interchange Queue Occupancy detectors are actually up-stream detectors used by the Fuzzy Phase Timing.

4.4 Framework of FLDI Model Implementation

The framework and procedure for implementing the FLDI control is presented in Figure 4.2. Inputs include initial signal phase, initial queue length, and continuously undated vehicle information at detectors. Initial queues in the beginning of each movement can be calculated using the counting function. For example, the FLDI counts the number of arrival and queue vehicles for the first 6 seconds of the current running phase. This counts process only happens once for each phase. The Fuzzy Phase Timing for each phase will generate the right green time extension based on these inputs.

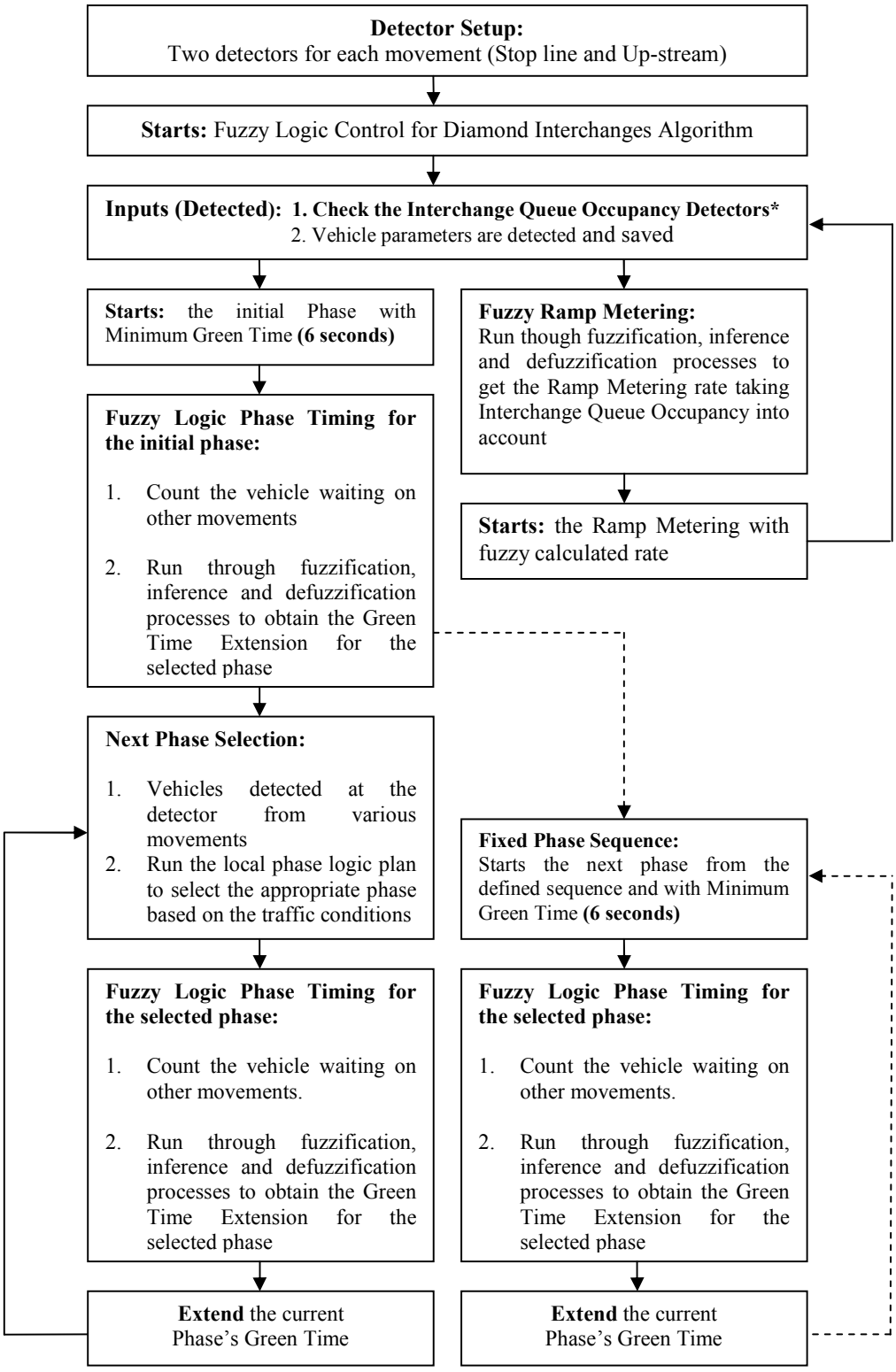


Figure 4.2 Framework of FLDI Algorithm

CHAPTER 5

SIMULATION STUDY

5.1 Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network

AIMSUN version 6 (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Network) is a microscopic, stochastic model for simulating traffic on 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 and signal optimisation interfaces like TRANSYT 7F, 12.

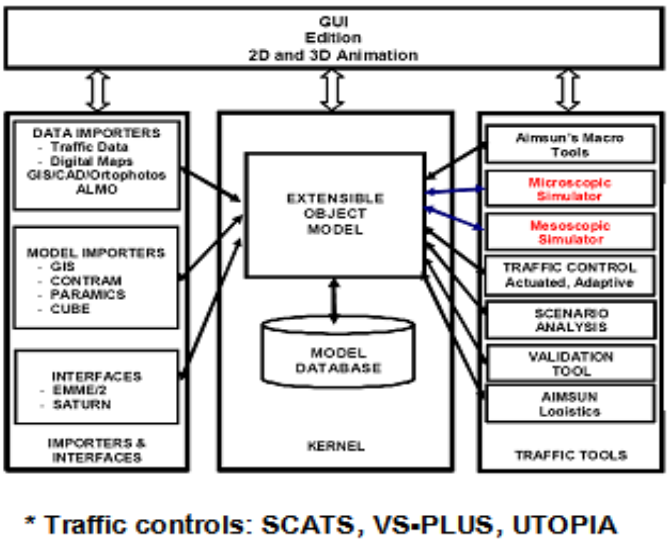


Figure 5.1 AIMSUN 6 environment (Aimsun, 2008)

5.2 AIMSUN Expansion

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 5.2 illustrates the conceptual structure of how Aimsun working with user application by means of Aimsun API module:

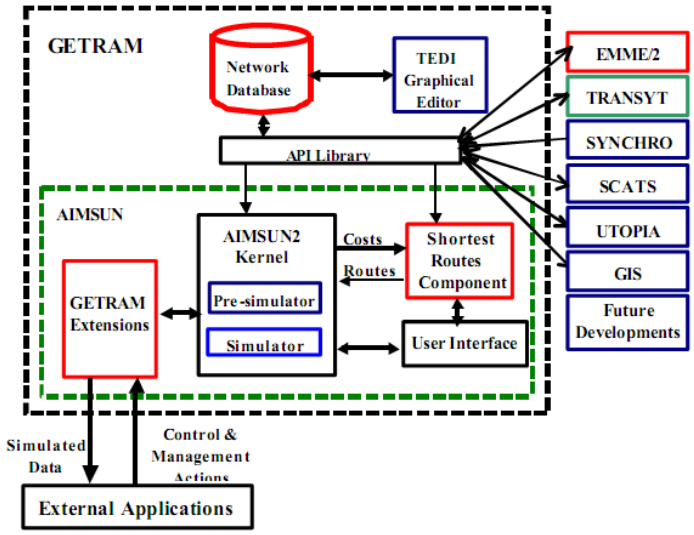


Figure 5.2 GETRAM/AIMSUN conceptual architecture (Barceló, 1995)

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 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 cycle, 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 cycle, 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 5.3.

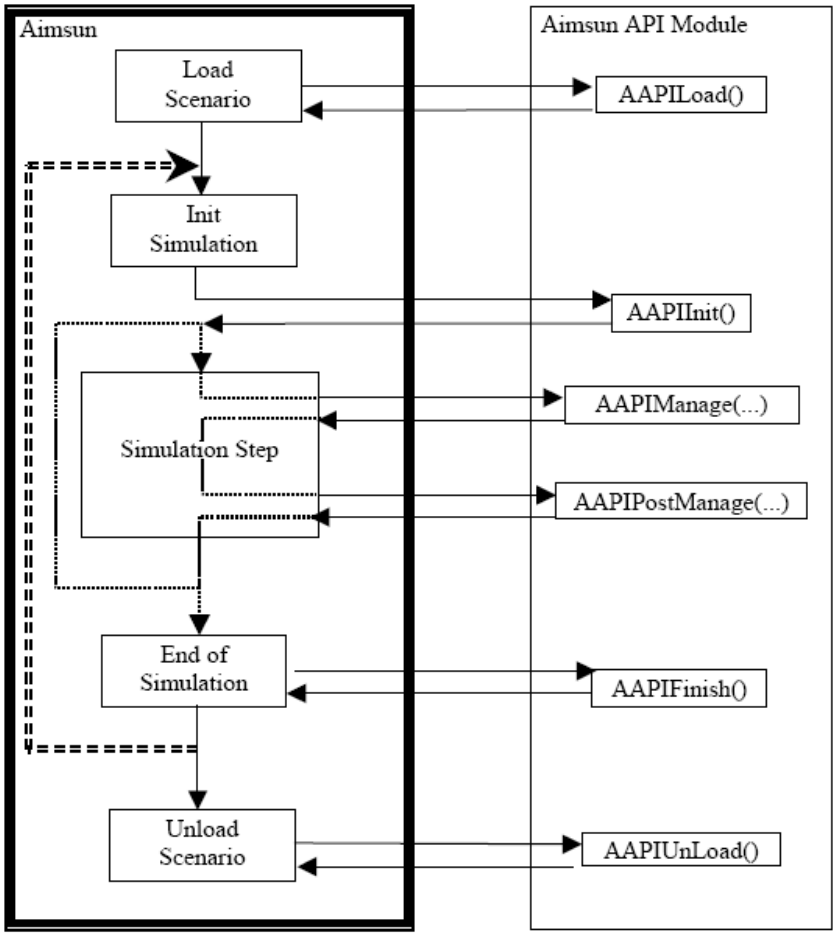


Figure 5.3 Interaction between Aimsun and Aimsun API module (Aimsun, 2008)

The proposed ramp metering algorithms programmed in Microsoft Visual C++ will be implemented in Aimsun simulator through AAPIManage () and AAPIPostManage () function by means of Microsoft Visual Studio 2005, where a Dynamic Link Library (DLL) will be generated and integrated to the simulator.

5.3 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 (J. Barceló et al, 1995):

a. Green time metering, with parameters green time and cycle time. It is modelled as a traffic light.
b. Flow metering, with parameters platoon length and flow (veh/h). The meter 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 disable 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.

5.4 Study Area and Assumptions

In this study, SH1 Upper Highway Diamond Interchange is used as a testing site for the FLDI model implementation. The simulations are 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 pedestrian crossing, bicycle and motorbike are used in this simulation.

The simulation will mimic the real-world peak hour period from 8:00-9:00 on Monday morning.

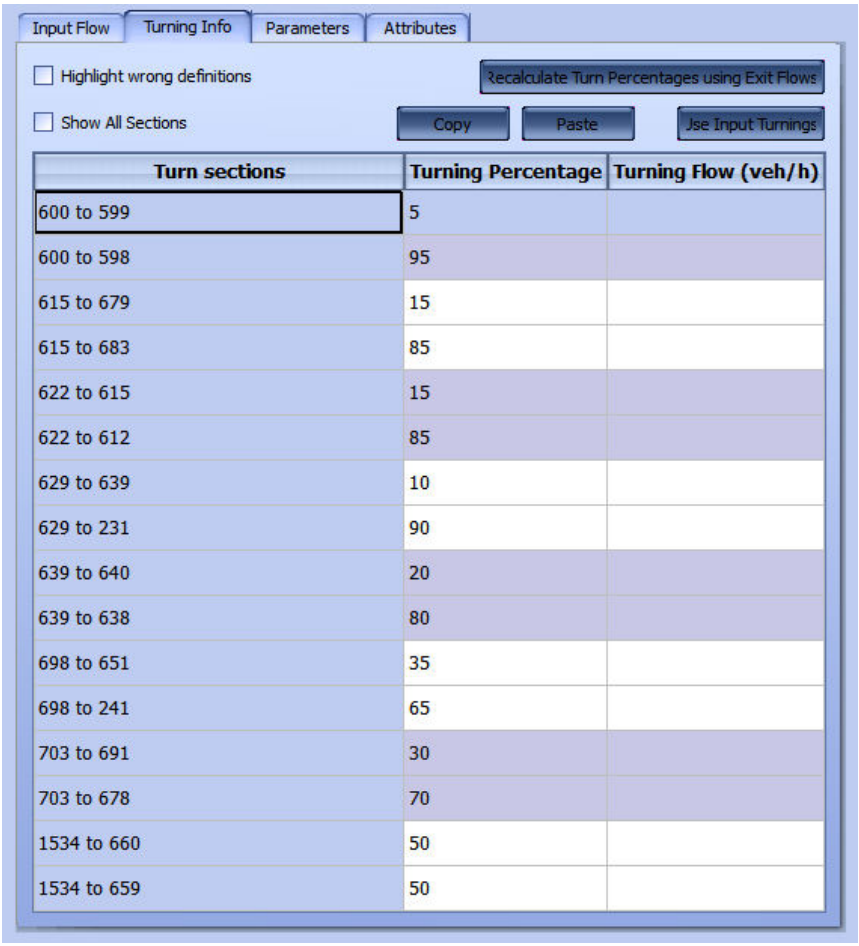
The simulation only covers three main phases (Phase B, F and D) based on the IDM report (Figure 3.15). The order of the phases is fixed starting with Phase B then moves to Phase F and ends with Phase D then repeat the process until the simulation ends. Only one ramp metering is used in this simulation (South-bound).

The traffic demands used in this simulation are partially based on the real-world data obtained from Transit NZ and North Shore City Council. Transit NZ only recorded the data for South-bound on-ramp and State Highway 1 for North-bound and South-bound. North Shore City Council only recorded the data for Constellation Drive and Upper Harbour Highway (connects to SH18). The turning movements at Constellation Drive and Upper Harbour Highway Intersections are setup based on observation and traffic survey on 15/07/2008.

Figure 5.4 SH1 Upper Highway Diamond Interchange in AIMSUN

5.5 Traffic Turning Setup

Currently there is no data recorded SH1 Upper Highway Diamond Interchange vehicle turning information. Therefore the traffic turning information in this simulation is based on estimation and observation 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 traffic junction next to the Constellation Park N Ride Bus station's entrance, and the last one was placed near the pedestrian crossing by the Upper Highway traffic junction.



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 5.5 Turning Information

Figure 5.6 Flow percentages for sections 612, 615 and 622

Figure 5.7 Flow percentages for sections 598, 599, 600, 615, 679 and 683

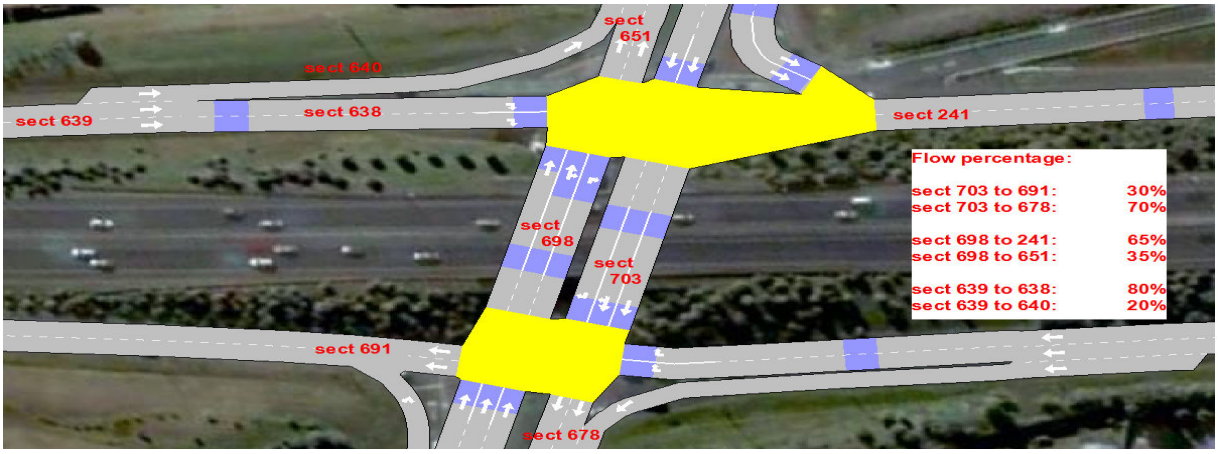


Figure 5.8 Flow percentages for sections 241, 638, 639, 640, 651, 698 and 703

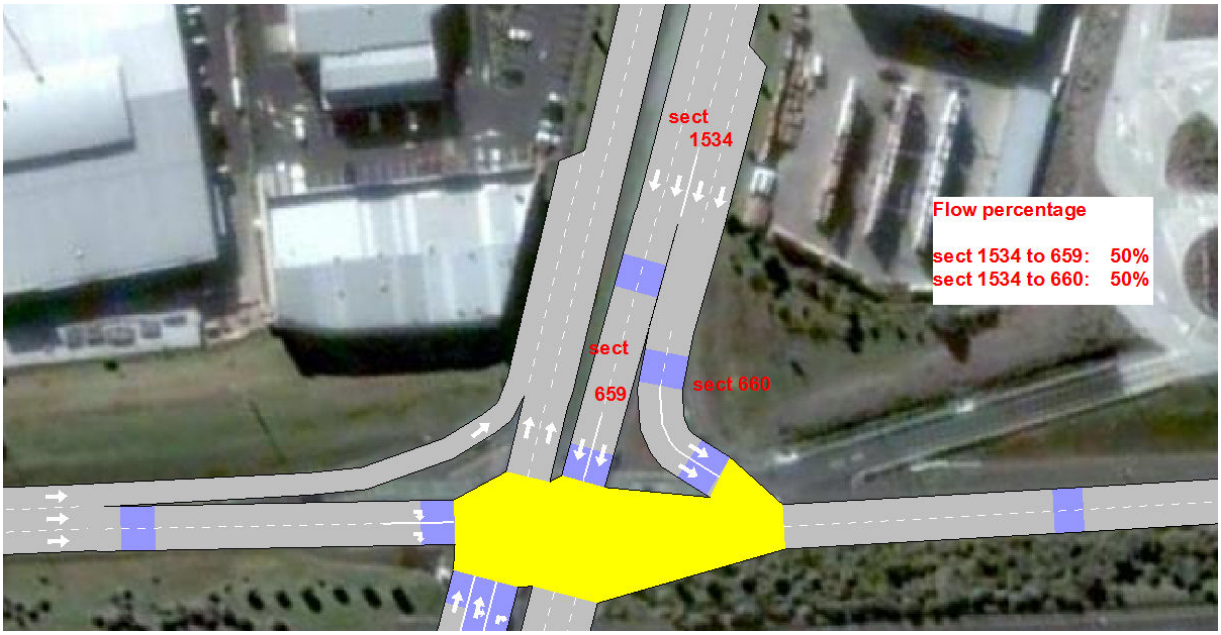


Figure 5.9 Flow percentages for sections 659, 660 and 1534

Figure 5.10 Flow percentages for sections 231, 629 and 639

5.6 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.

Totally eight traffic demands are used in this simulation: 3348 vehicles/hour (Table 51), 3752 vehicles/hour (Table 5.2), 4592 vehicles/hour (Table 5.3), 6272 vehicles/hour (Table 5.4), 7952 vehicles/hour (Table 5.5) and 8783 vehicles/hour (Table 5.6). 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 Tristam – 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 free way/motorway capacity is more than 5000 vehicles/hour.

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	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 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	

5.7 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.

5.8 Parameter Setup

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.

On ramp speed limit for cars: 80 - 100 km//hr
On ramp speed limit for heavy trucks (HOV): 80 - 100 km/hr
Arterial roads speed limit for cars: 50-60 km/hr
Arterial roads speed limit for heavy trucks (HOW): 50-60
Speed limit on the motorway for cars and trucks: 80 - 100km/hr
Maximum vehicle acceleration for passenger car: 1.08 ms^{-2}
Maximum vehicle acceleration for heavy trucks (HOV): 0.40 ms^{-2}
Maximum vehicle deceleration for passenger car: -1.72 ms^{-2}
Maximum vehicle deceleration for heavy trucks (HOV): -1.19 ms^{-2}

Capacity on ramp: maximum 1700 vehicles/hour/lane
Capacity on arterial roads: maximum of 1000 vehicles/hour/lane
Capacity on signalised streets: maximum of 1000 vehicles/hour/lane
Capacity on motorway: max 2500 vehicles/hour/lane

5.9 Detector Type

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.

5.10 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
Reaction time:

- Simulation step: 0.75 second
- Reaction time at stop: 1.35 seconds
- Route choice model: fixed using Travel Time in Free Flow models

5.11 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.

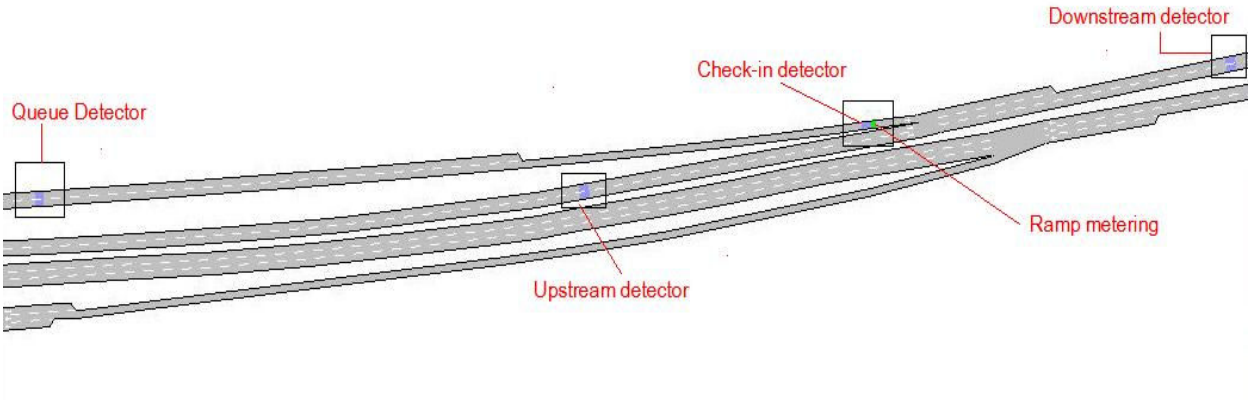


Figure 5.11 Detectors setup for Fuzzy Ramp system in AIMSUN

5.12 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.

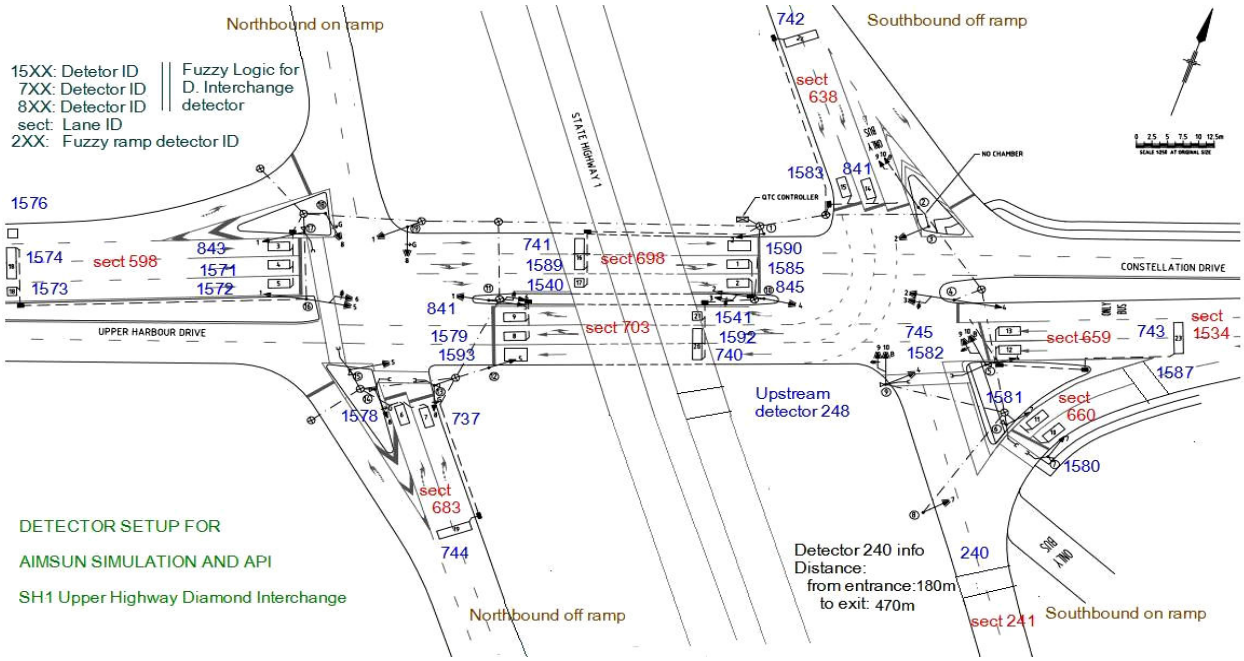


Figure 5.12 Detectors setup for SH1 Upper Highway Diamond Interchange in AIMSUN

5.13 Geometric Information

For this simulation, the motorway geometric layout was obtained as a high-resolution JPEG image from Google Earth Professional Edition. The file showing curbs lines and edges of the road pavement. The map was geometrically accurate, having been produced by aerial photogrammetric by Google Inc. and North Shore City Council. Additional information on the widths and number of lanes was obtained from Transit NZ.

5.14 Fuzzy Logic Phase Timing and Phase Control API

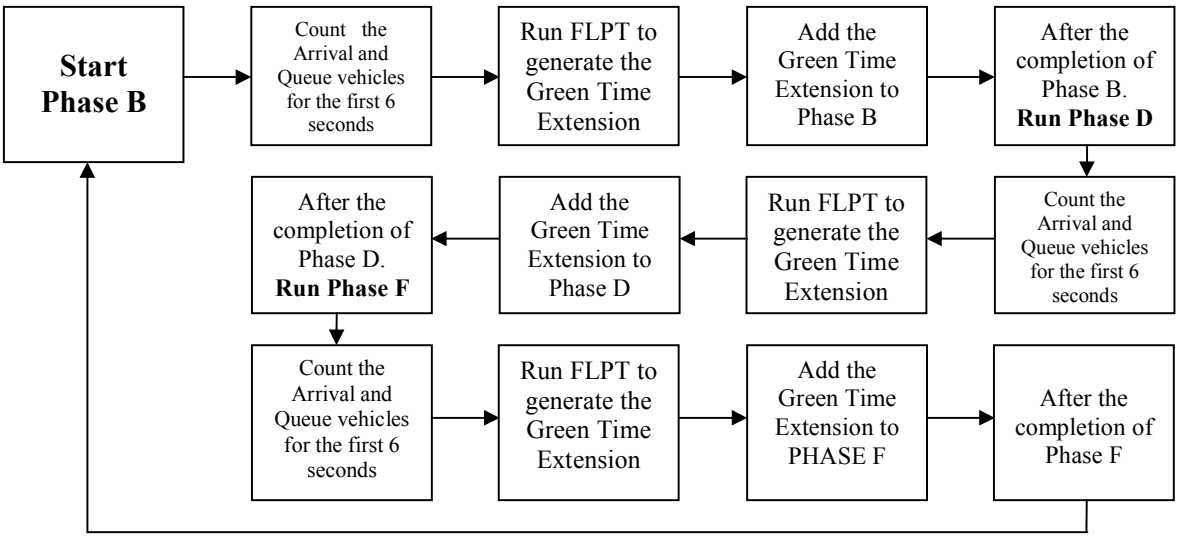


Figure 5.13 SH1 Upper Highway Diamond Interchange Simulation Procedures

The Phase Control algorithm (Figure 5.13) works as a state machine which controls the sequence of phases. Initially, the algorithm triggers Phase B with the minimum phase duration of 6 seconds. During the first 6 seconds period in Phase B, the algorithm counts the arrival and queue vehicles in the two connected intersections. These counts will be used as the Arrival and Queue inputs for the Fuzzy Phase Timing. Current phase detection function is developed to get the current running phase in the AIMSUN and check whether the counting has been run on the current phase. If the function detects a phase change and the counting has not been done for the new phase then it will starts counting the vehicles for the first 6 seconds of the new phase. The Fuzzy Phase Timing uses the Arrival and Queue counts to generate the appropriate green time extension for the current running phase. The green time extension can be any values but has to be smaller than the maximum defined green time in AIMSUN. The purpose of defining a maximum green time in AIMSUN is to enable the Phase Control algorithm to trigger the next appropriate phase so that all the movements in the diamond interchanges have been covered.

5.15 Counting the Arrival and Queue vehicles API

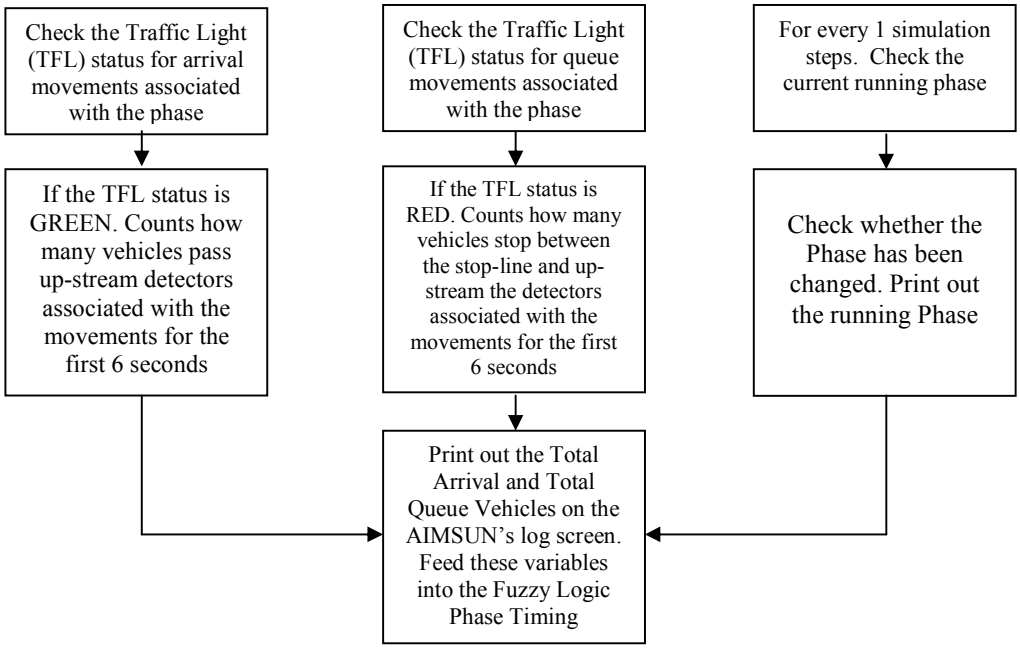


Figure 5.14 Counting Arrival and Queue Vehicles Procedures in AIMSUN

As mentioned above the Arrival and Queue counting algorithm is part of the Phase Control. The algorithm check the traffic light status in AIMSUN for arrival movements associated with the current running phase. If the status is Green, the algorithm counts how many vehicles pass the up-stream detectors associated with the current movements for the first six seconds. On the other side, all the remaining movements that are waiting for their turns (Red) are considered as queue vehicles (counted as Queue vehicles). Every simulation steps, the algorithm check for the current running phase, if it detects a change in phase then it will do a further check to see whether the counting algorithm has been initiated or not. This is to make sure that the counting only been done once in the first 6 seconds of the running phase. Finally, the algorithm prints out the current running phase along with the Total Arrival and Queue counts on the AIMSUN's log screen and feed these values into the Fuzzy Logic Phase Timing algorithm as Arrival and Queue inputs. The maximum green time for each phase is not the same therefore each phase has their own Fuzzy Logic Phase Timing with different ranges for the Arrival and Queue memberships. Therefore the Fuzzy Logic Phase Timing green time extension for each phase is also different from the others.

5.16 Performance Measures of Effectiveness

The FLDI 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 simulator) from both the motorway speed limit and driving characteristics of each individual vehicle.

5.16.1 Motorway MOE

Total travel time: is the total time accumulated by all the vehicles while travelling on the freeway 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.

5.16.2 Ramp Delay/ Queue MOE

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

5.16.3 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)

5.17 Simulation Pictures

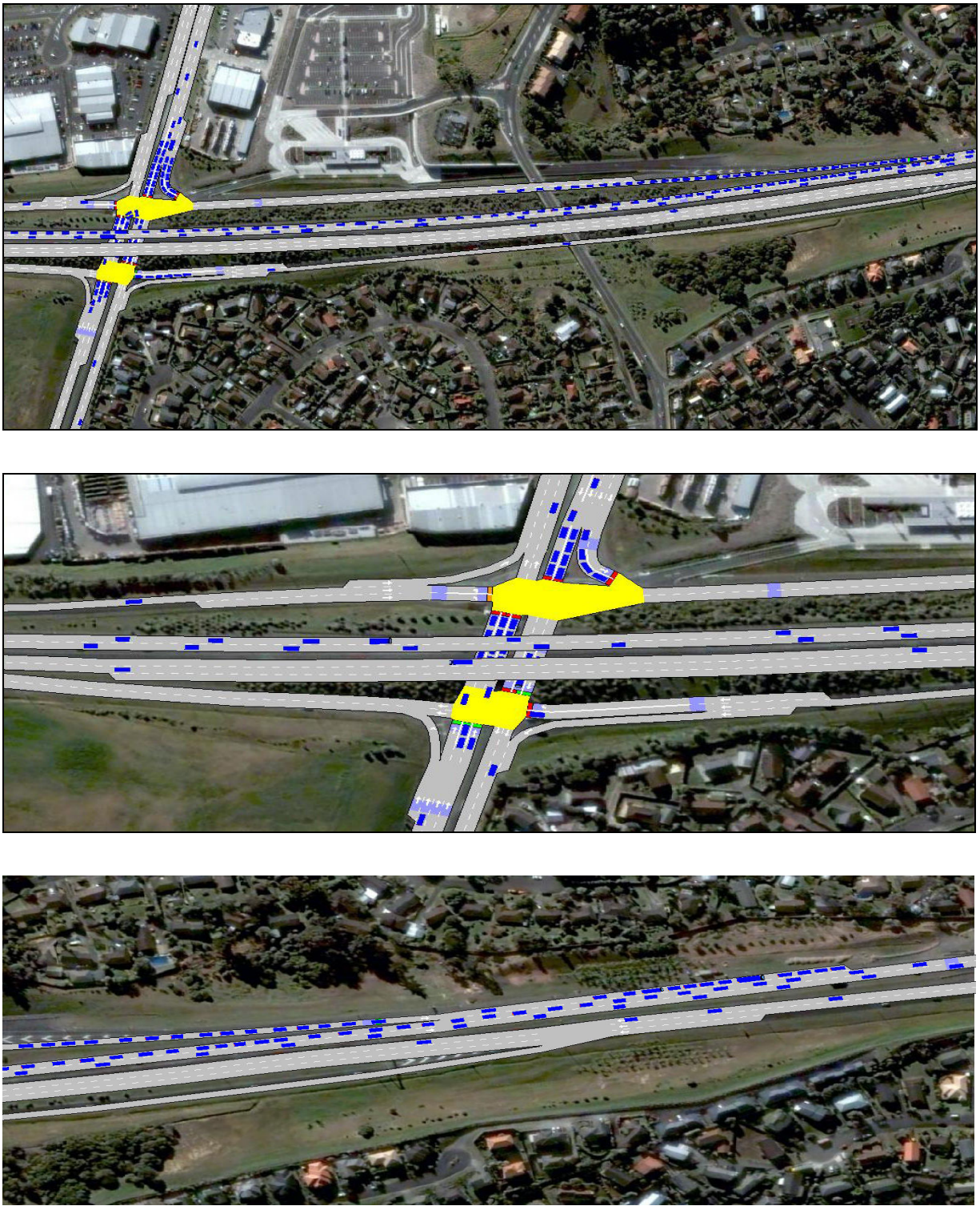


Figure 5.15 State Highway 1 Upper Highway Interchange AIMSUN simulation pictures

CHAPTER 6

RESULTS AND ANALYSIS

The performance of the Fuzzy Diamond Interchange Control with Fuzzy Ramp Metering (FLDI) can be 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 performances of the Downstream, Ramp and overall System's performance which including the performances of the two connected intersections. This can be done by using AIMSUN where all the controllers are implemented on to the SH1 Upper Highway Diamond Interchange model. 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 (Table 5.1 to Table 5.8). 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 order to make the comparisons between the FLDI and the other models, identical conditions have to be set during the simulations. The simulation runs for one hour representing the peak hour period in the morning from 8:00-9:00AM (Monday). In each scenario, there are three simulation runs, one for each model.

6.1 Analysis of Scenario 1 and 2 (Table 5.1, 5.2 and Table 5.7, 5.8)

Scenario 1 and 2 are considered as Free Flow traffic conditions with Total Traffic Demand ranges from 3348 to 3752 vehicles per hour. According to the results obtained from the AIMSUN's simulation (Table 5.9), Fuzzy Diamond Interchange Control with Fuzzy Ramp Metering (FLDI) outperformed the Actuated Diamond Interchange Control with No Metering (ADINM) and Actuated Diamond Interchange Control with Fuzzy Ramp Metering (ADIFM) models. The FLDI 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% (Scenario2) in comparison to the ADINM model.

Moreover, the FLDI 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%) as shown in Figure 6.1 and 6.2 below.

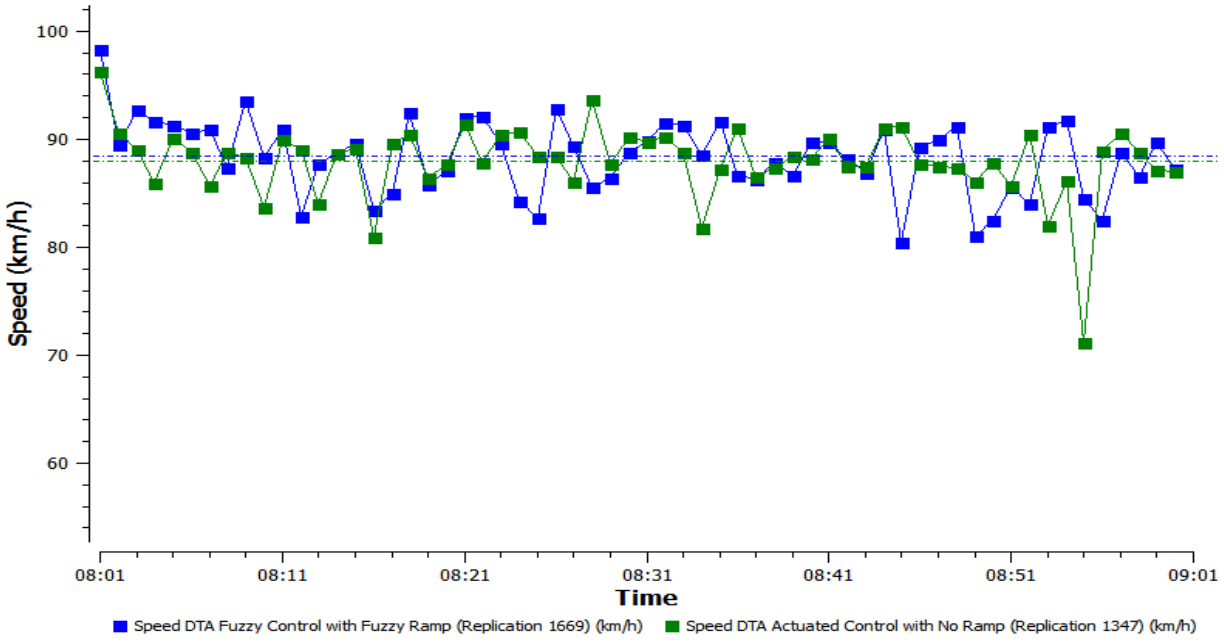


Figure 6.1 Scenario 1 downstream average speed (FLDI and ADINM)

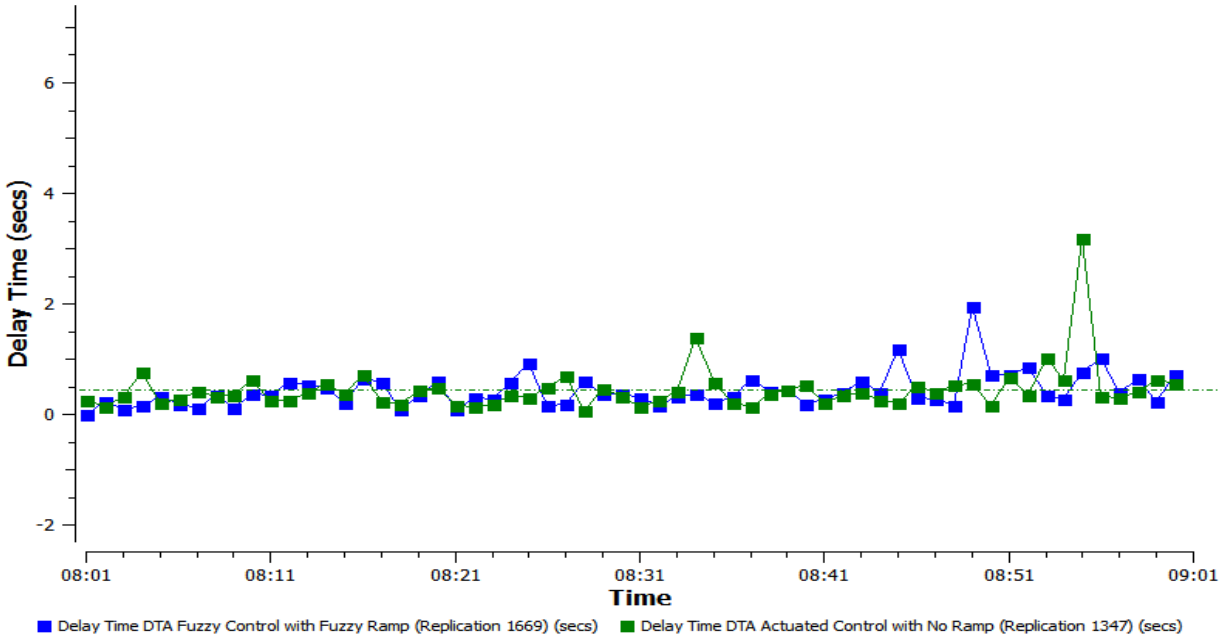


Figure 6.2 Scenario 1 downstream average delay (FLDI and ADINM)

Similar to Scenario 1, in Scenario 2 the FLDI 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 FLDI increases the ramp metering rate enabling more vehicles getting into the motorway causing a slight congestion in the downstream movements. However, the FLDI reduced the system total travel time by 17.29% (from 58.59 hours to 48.46 hours) and average delay time by -70.13%.

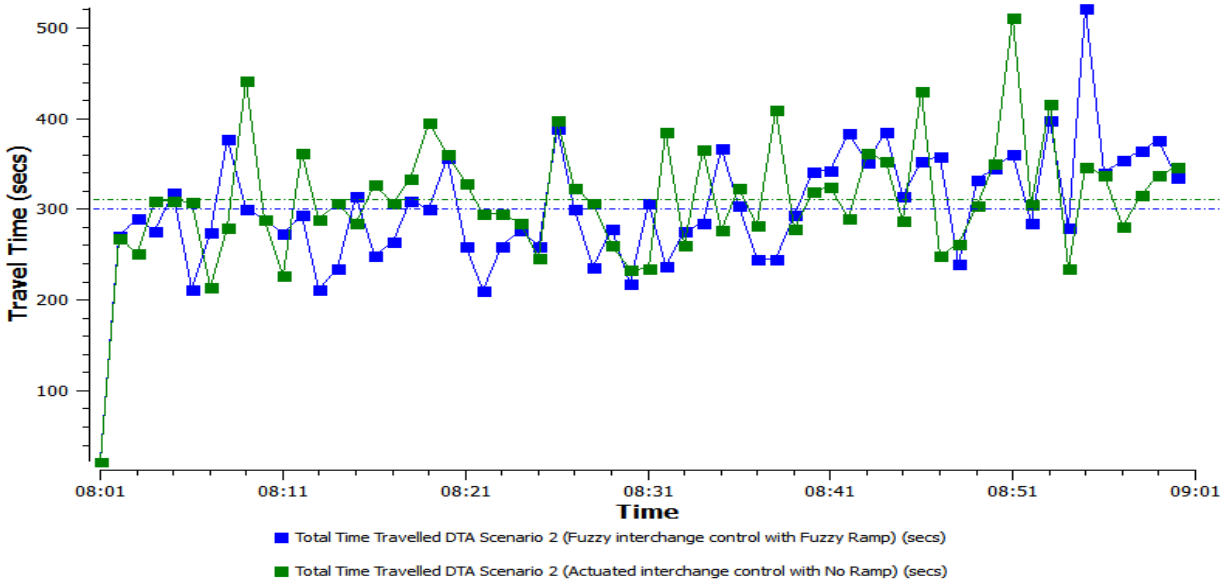


Figure 6.3 Scenario 2 downstream total travel time (FLDI and ADINM)

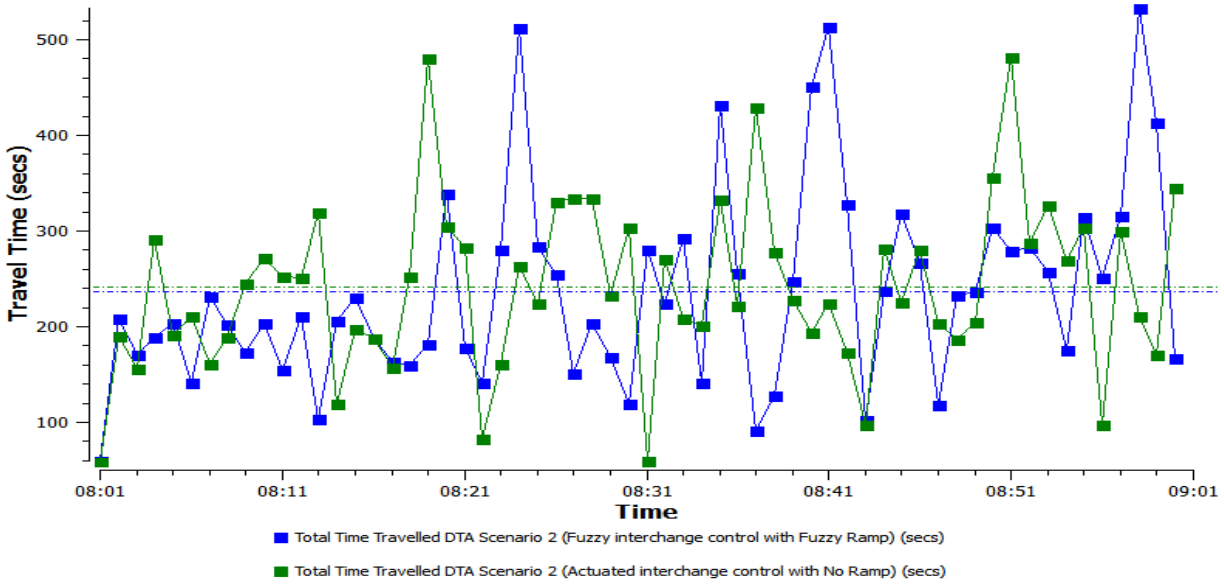


Figure 6.4 Scenario 2 ramp total travel time (FLDI and ADINM)

Table 5.7 Measure of effectiveness between models (3348 vehicles per hour – Scenario 1)

	Downstream Measure of Effectiveness				Ramp Measure of Effectiveness			System Measure of Effectiveness		
	Total Travel Time (sec/km)	**Average Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Average Speed (km/hr)**	**Total Ramp Travel Time (sec/km)**	**Average Ramp Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Total Travel Time (hour)**	**Average Delay Time (sec/veh/km)**	**Average Flow Rate (veh/hr)**
Actuated diamond interchange control with No Ramp Metering **(A)**	14702.54	0.45	1865	86.99	9363.93	2.15	455	42.75	21.28	2454
Actuated diamond interchange control with Fuzzy Ramp Metering **(B)**	13715.94	0.44	1745	88.23	8885.48	3.85	383	36.94	11.17	2388
Fuzzy interchange control with Fuzzy Ramp Metering **(C)**	14528	0.42	1858	88.40	9340	3.08	422	35.86	5.16	2477
Percentage of change										
B versus A	-6.71%	-2.22%	-6.43%	1.43%	-5.11%	79.07%	-15.82%	-13.59%	-47.51%	-2.69%
C versus A	-1.19%	-6.67%	-0.38%	1.62%	-0.26%	43.26%	-7.25%	-16.12%	-75.75%	0.94%
C versus B	5.92%	-4.55%	6.48%	0.19%	5.12%	-20.00%	10.18%	-2.92%	-53.80%	3.73%

Table 5.8 Measure of effectiveness between models (3752 vehicles per hour – Scenario 2)

	Downstream Measure of Effectiveness				Ramp Measure of Effectiveness			System Measure of Effectiveness		
	Total Travel Time (sec/km)	**Average Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Average Speed (km/hr)**	**Total Ramp Travel Time (sec/km)**	**Average Ramp Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Total Travel Time (hour)**	**Average Delay Time (sec/veh/km)**	**Average Flow Rate (veh/hr)**
Actuated diamond interchange control with No Ramp Metering **(A)**	18650.59	0.63	2315	86.14	14497.26	2.67	679	58.59	24	3314
Actuated diamond interchange control with Fuzzy Ramp Metering **(B)**	17718.27	0.78	2147	84.45	20122.68	10.71	651	62.96	32.71	3221
Fuzzy interchange control with Fuzzy Ramp Metering **(C)**	18058.13	0.66	2220	85.24	14208.78	5.34	577	48.46	7.17	3284
Percentage of change										
B versus A	-5.00%	23.81%	-7.26%	-1.96%	38.80%	301.12%	-4.12%	7.46%	36.29%	-2.81%
C versus A	-3.18%	4.76%	-4.10%	-1.04%	-1.99%	100%	-15.02%	-17.29%	-70.13%	-0.91%
C versus B	1.92%	-15.4%	3.40%	0.94%	-29.39%	-50.14%	-11.37%	-23.03%	-78.08%	1.96%

6.2 Scenario 3 and 4 Analysis (Table 5.3, 5.4 and Table 5.9, 5.10)

Scenario 3 and 4 are considered as moderate traffic flow. For Scenario 3, the FLDI continue shows its strength over the ADIFM and the ADINM models with a sharp decrease in the system average flow rate (from 40.04 to 17.09 sec/veh/km) and system total travel time (from 82.48 to 68.75 hours). In Scenario 4, the FLDI fails to reduce the STTT, the STTT increases from 135.40 hours to 210.07 hours (201.72 hours for ADIFM). The system average delay time increases significantly from 70.18 to 133.75 sec/veh/km. The system average flow rate almost the same with only slight decreases of 0.53% (5.69% for the ADIFM case). The reason why FLDI fails to reduce the STTT is because the ramp demand is relatively low at that moment and the motorway is under free flow condition. AISMSUN generates vehicles randomly and at some points the system total traffic demand could exceeds the road capacity and cause traffic delay. On the other hand vehicles from the ramp merging into the mainstream continuously interrupt the traffic platoon causing the formation of the congestion on the motorway. The FLDI's ramp metering is unable to response to the sudden change of the traffic flows, it could even generate a stricter metering rate that even cause extra traffic delay and worsen the traffic condition. The downstream total travel time decreases by 11% in the FLDI simulation.

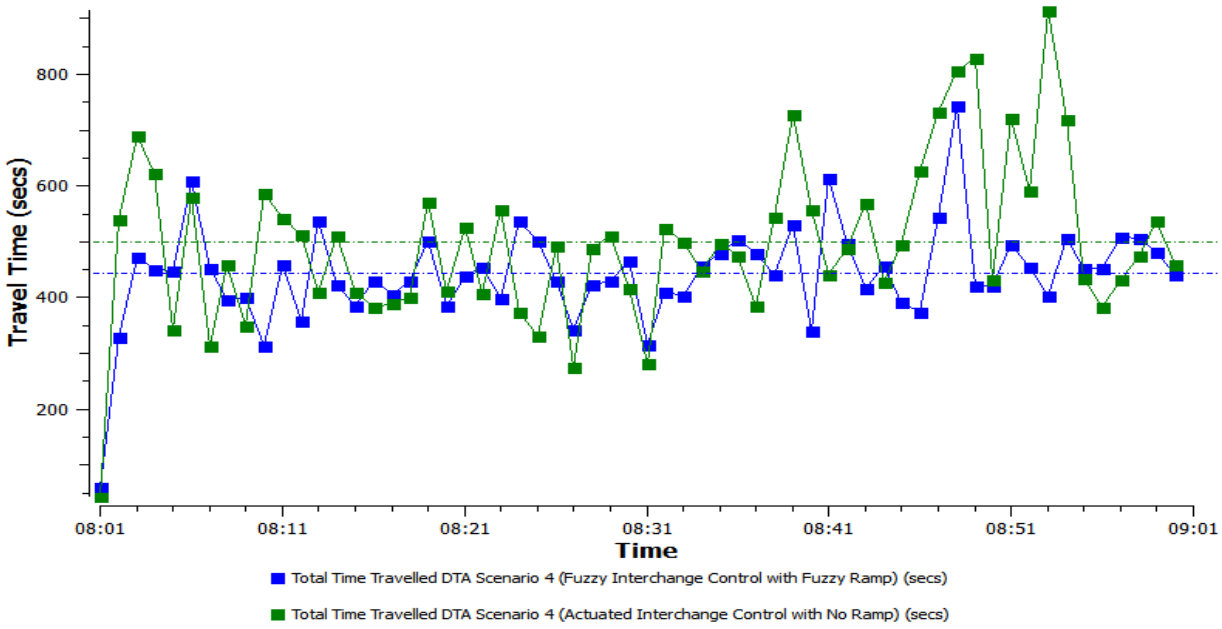


Figure 6.5 Scenario 4 Downstream Total Travel Time (FLDI and ADINM)

Table 5.9 Measure of effectiveness between models (4592 vehicles per hour – Scenario 3)

	Downstream Measure of Effectiveness				Ramp Measure of Effectiveness			System Measure of Effectiveness		
	Total Travel Time (sec/km)	**Average Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Average Speed (km/hr)**	**Total Ramp Travel Time (sec/km)**	**Average Ramp Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Total Travel Time (hour)**	**Average Delay Time (sec/veh/km)**	**Average Flow Rate (veh/hr)**
Actuated diamond interchange control with No Ramp Metering **(A)**	21239.96	0.80	2497	84.58	18265.97	2.90	759	82.48	40.04	3851
Actuated diamond interchange control with Fuzzy Ramp Metering **(B)**	20606.25	0.93	2460	83.05	23244.62	12.60	722	122.56	103.87	3796
Fuzzy interchange control with Fuzzy Ramp Metering **(C)**	21933.58	1.10	2563	82.09	50023.71	40.14	816	68.75	17.09	4037
Percentage of change										
B versus A	-2.98%	16.25%	-1.48%	-1.81%	27.26%	334.48%	-4.87%	48.59%	159.42%	-1.43%
C versus A	3.27%	37.50%	2.64%	-2.94%	173.86%	1284.14%	7.51%	-16.65%	-57.32%	4.83%
C versus B	6.44%	18.28%	4.19%	-1.16%	115.21%	218.57%	13.02%	-43.91%	-83.55%	6.35%

Table 5.10 Measure of effectiveness between models (6272 vehicles per hour – Scenario 4)

	Downstream Measure of Effectiveness				Ramp Measure of Effectiveness			System Measure of Effectiveness		
	Total Travel Time (sec/km)	**Average Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Average Speed (km/hr)**	**Total Ramp Travel Time (sec/km)**	**Average Ramp Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Total Travel Time (hour)**	**Average Delay Time (sec/veh/km)**	**Average Flow Rate (veh/hr)**
Actuated diamond interchange control with No Ramp Metering **(A)**	29914.49	1.82	3188	77.98	25356.02	4.42	1099	135.40	70.18	5240
Actuated diamond interchange control with Fuzzy Ramp Metering **(B)**	27467.68	1.59	3022	78.57	223102.93	236.05	865	201.725	138.50	4942
Fuzzy interchange control with Fuzzy Ramp Metering **(C)**	26623.90	1.34	3032	79.72	254338.76	271.43	868	210.07	133.75	5212
Percentage of change										
B versus A	-8.18%	-12.64%	-5.21%	0.76%	779.88%	5240.50%	-21.29%	48.98%	97.35%	-5.69%
C versus A	-11.00%	-26.37%	-4.89%	2.23%	903.07%	6040.95%	-21.02%	55.15%	90.58%	-0.53%
C versus B	-3.07%	-15.72%	0.33%	1.46%	14.00%	14.99%	0.35%	4.14%	-3.43%	5.46%

6.3 Scenario 5 and 6 Analysis (Table 5.5, 5.6 and Table 5.11, 5.12)

Scenario 5 and 6 represent for congested traffic with the total traffic demand exceeds 8000 vehicles per hour. This means that the total traffic demand in the system is higher than the road capacity. In the previous scenarios where the traffic demand reaches or nearly reaches the road capacity, the FLDI and ADIFM perform very well. However when congestion occurs, the FLDI and ADFM fail to reduce the STTT and system average delay time. At this point, the ADFM is less effective in stabilizing the traffic flow than the FLDI as ADFIM's Total Travel Time increases by 43.60% comparing to 35.34% by the FLDI. When the traffic demand reaches 8783 vehicles (Scenario 6) both the FLDI and ADIFM bounce back with a decrease of 3% in the system total travel time. The ADINM does not perform well during congestion period because the traffic from the on-ramp is not regulated therefore vehicles from the ramp continuously interrupting the motorway downstream flow causing further congestion. On the other hand, the FLDI model is designed to avoid the formation of congestion and in extreme traffic condition the algorithm also prevents the congestion getting worse. Figure 6.6 shows FLDI significantly improve the downstream average speed in comparison to ADINM model.

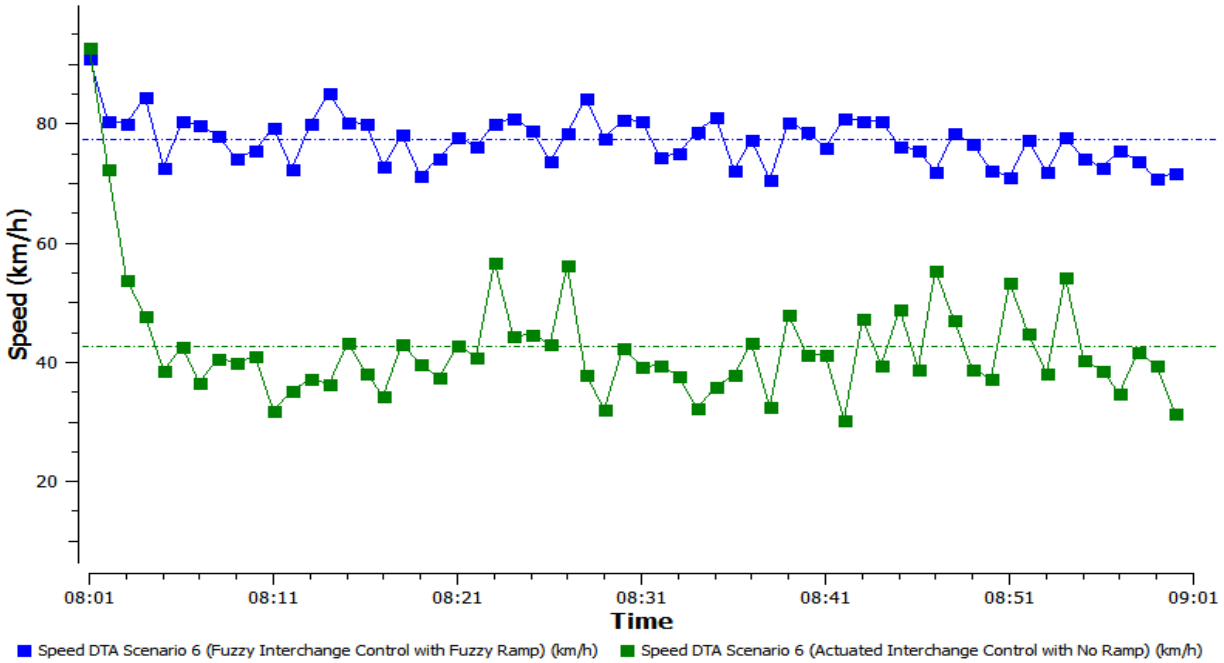


Figure 6.6 Scenario 6 downstream average speed (FLDI and ADINM)

Table 5.11 Measure of effectiveness between models (7952 vehicles per hour – Scenario 5)

	Downstream Measure of Effectiveness				Ramp Measure of Effectiveness			System Measure of Effectiveness		
	Total Travel Time (sec/km)	**Average Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Average Speed (km/hr)**	**Total Ramp Travel Time (sec/km)**	**Average Ramp Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Total Travel Time (hour)**	**Average Delay Time (sec/veh/km)**	**Average Flow Rate (veh/hr)**
Actuated diamond interchange control with No Ramp Metering **(A)**	38064.52	2.97	3606	73.48	24765.38	2.68	1162	183.62	88.37	6144
Actuated diamond interchange control with Fuzzy Ramp Metering **(B)**	31371.89	1.77	3389	77.12	267671	286.58	870	263.68	151.97	6041
Fuzzy interchange control with Fuzzy Ramp Metering **(C)**	30787.52	1.63	3399	77.74	243428.20	262.34	867	248.51	149.24	5812
Percentage of change										
B versus A	-17.58%	-40.40%	-6.02%	4.95%	980.83%	10593.28%	-25.13%	43.60%	71.97%	-1.68%
C versus A	-19.12%	-45.12%	-5.74%	5.80%	882.94%	9688.81%	-25.39%	35.34%	68.88%	-5.40%
C versus B	-1.86%	-7.91%	0.30%	0.80%	-9.06%	-8.46%	-0.34%	-5.75%	-1.80%	-3.79%

Table 5.12 Measure of effectiveness between models (8783 vehicles per hour – Scenario 6)

	Downstream Measure of Effectiveness				Ramp Measure of Effectiveness			System Measure of Effectiveness		
	Total Travel Time (sec/km)	**Average Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Average Speed (km/hr)**	**Total Ramp Travel Time (sec/km)**	**Average Ramp Delay (sec/veh)**	**Average Flow Rate (veh/hr)**	**Total Travel Time (hour)**	**Average Delay Time (sec/veh/km)**	**Average Flow Rate (veh/hr)**
Actuated diamond interchange control with No Ramp Metering **(A)**	115813.64	21.06	4027	41.54	248979.82	173.50	1326	285.93	130.28	7130
Actuated diamond interchange control with Fuzzy Ramp Metering **(B)**	34064.89	2.29	3476	75.17	274809.15	294.92	867	276.53	152.94	6272
Fuzzy interchange control with Fuzzy Ramp Metering **(C)**	31958.45	1.59	3527	77.38	274196.05	294	870	277.17	152.59	6414
Percentage of change										
B versus A	-70.59	-89.13	-13.68	80.96	10.37	69.98	-34.62	-3.29	17.39	-12.03
C versus A	-72.41	-92.45	-12.42	86.28	10.13	69.45	-34.39	-3.06	17.12	-10.04
C versus B	-6.18	-30.57	1.47	2.94	-0.22	-0.31	0.35	0.23	-0.23	2.26

6.4 FLDI and ADIFM Overall Results Analysis

Figure 6.7 shows the percentage of change in total travel time (TTT) of FLDI and ADIFM for six scenarios with total traffic demand ranges from 3348 to 8783 vehicles per hour.

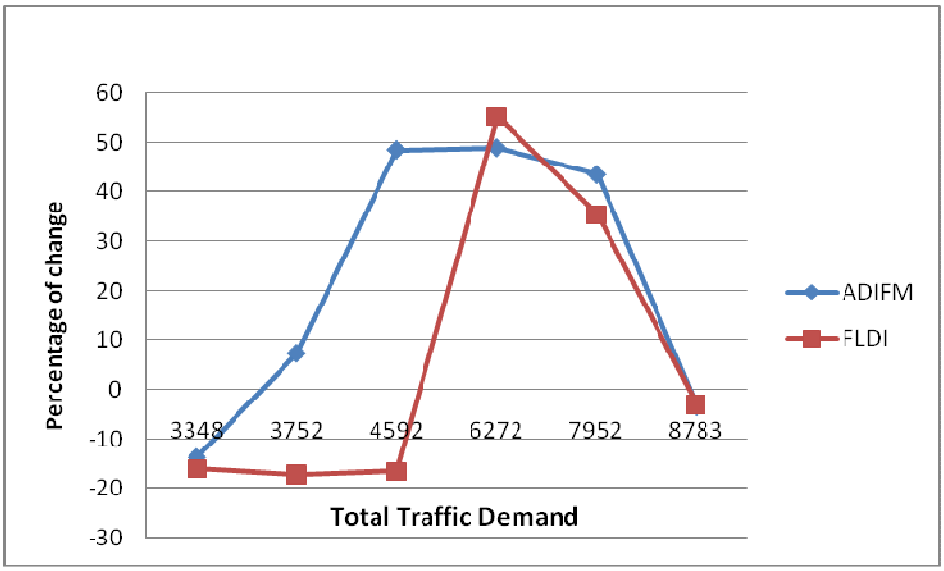


Figure 6.7 Percentage of change in system total travel time (ADIFM and FLDI)

From Figure 6.7, when the total traffic demand ranges from 4592 to 7952 vehicles per hour the FLDI and ADIFM failed 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. An when the total traffic demand reaches 7952 vehicles the performance of FLDI and ADIFM are very close. When the total traffic demand reaches 8784 vehicles, both FLDI and ADIFM reduced the TTT by 3%. However, during the non congestion periods (3348 to 4592 vehicles per hour), FLDI performed better than ADIFM model. The FLDI reduced 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, 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 connected intersection 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 main stream to avoid the vehicles on ramp spill back into the connected intersection causing the traffic congestion on the intersections.

Figure 6.8 shows the percentage of change in downstream average speed of FLDI and ADIFM for six scenarios with total traffic demand ranges from 3348 to 8783 vehicles per hour.

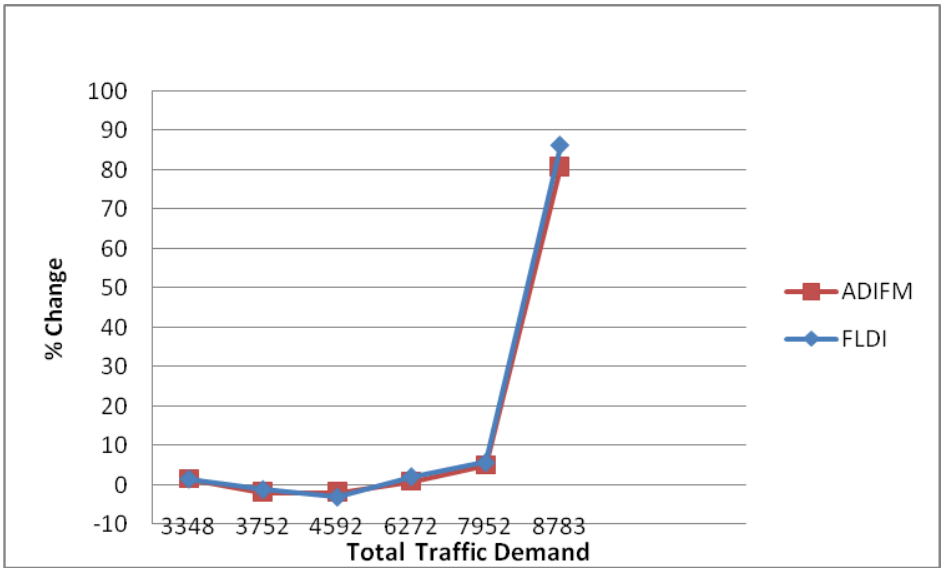


Figure 6.8 Percentage of change in downstream average speed (ADIFM and FLDI)

According to Figure 6.7 above, when the traffic is in free flow conditions (3348 to 4592 vehicles per hour), the FLDI and ADIFM failed to increase the downstream speed. The reason why both models failed to increase the downstream speed is because ramp demand is relatively low and the motorway is under free flow condition in most cases. Although total traffic demand could sometimes exceed the road capcity and cause traffic delay, low ramp demand continoulsy interrupt the traffic conditions which leads to traffic congestion. The FLDI and ADIFM's ramp metering rate are unable to response to the change of the traffic flow and they could even generate a stricter metering rate that may even cause extra delay and make the traffic condition worsen. This would lead to the decrease in the downstream speed. Meanwhile, during the congested periods when the total traffic demand exceed 6272 vehicles per hour, the FLDI and ADIFM improved the speed significantly. The FLDI performed better than ADIFM by 1.47% when traffic demand reaches 8783 vehicles per hour.

Figure 6.9 shows the percentage of change in downstream average delay (DAD) of FLDI and ADIFM for six scenarios with total traffic demand ranges from 3348 to 8783 vehicles per hour.

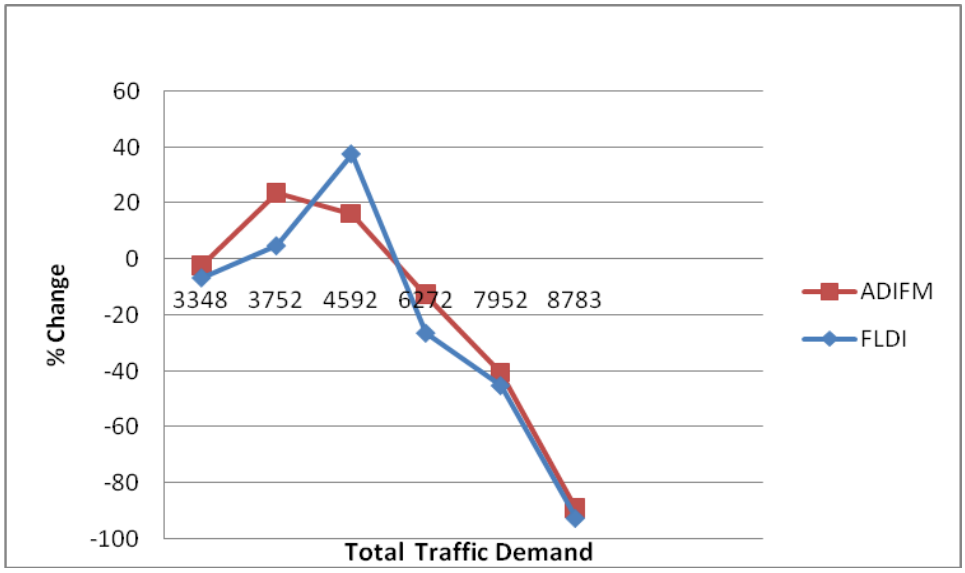


Figure 6.9 Percentage of change in downstream average delay (ADIFM and FLDI)

Similar to the downstream average speed, the FLDI and ADIFM failed to reduce the delay when the total traffic demand is less than 6727 vehicles per hour during the non congestion periods. Situation starts to change once the total traffic demand exceed 6272 vehicles per hour as FLDI performed better than the ADIFM by further reducing the average delay of downstream vehicles. The FLDI algorithm performs better than the ADIFM when the traffic is congested. This makes sense as the fuzzy membership functions are designed to cope with different traffic conditions and emphasised on the heavily congested traffic conditions. The FLDI is more flexible than the ADIFM models, the flexibility here involves the number of vehicles sensed at the incoming intersections and (Constellation Drive and Upper Harbour Highway) and the extension of the green time with different extension range for each particular phase. In the actuated controller (which is an enhanced version of fixed-time controller, the green time is extended whenever there is a presence of a vehicle). However, these times are fixed in advance up to a maximum time limit. Meanwhile the FLDI's green time extension values are not fixed therefore the system reacts better when there are changes in the traffic conditions.

6.5 Summary

In this chapter, Fuzzy Logic Diamond Interchange Control with Fuzzy Ramp Metering, Actuated Diamond Interchange Control with No Ramp Metering and Actuated Diamond Interchange Control with Fuzzy Ramp are built and tested in AIMSUN environment with six different traffic scenarios ranges from free flow to congestion. From the comparison results, FLDI outperformed 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.

CHAPTER 7

SENSITIVITY ANALYSIS

Sensitivity analysis is the study of how the variation (uncertainty) in the output of an algorithm can be apportioned, qualitatively or quantitatively, to different sources of variation in the input of a model (Saltelli et al., 2008). This chapter conducts two types of sensitivity analysis to characterise the proposed FLDI control algorithm.

7.1 Positions of Detectors for On-ramp and Motorway

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. 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 (Shaflik, 1995). As mentioned earlier in Chapter 4, 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:

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 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) will be used as the comparison factor. According to Figure 7.1, 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.

Table 7.1 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%

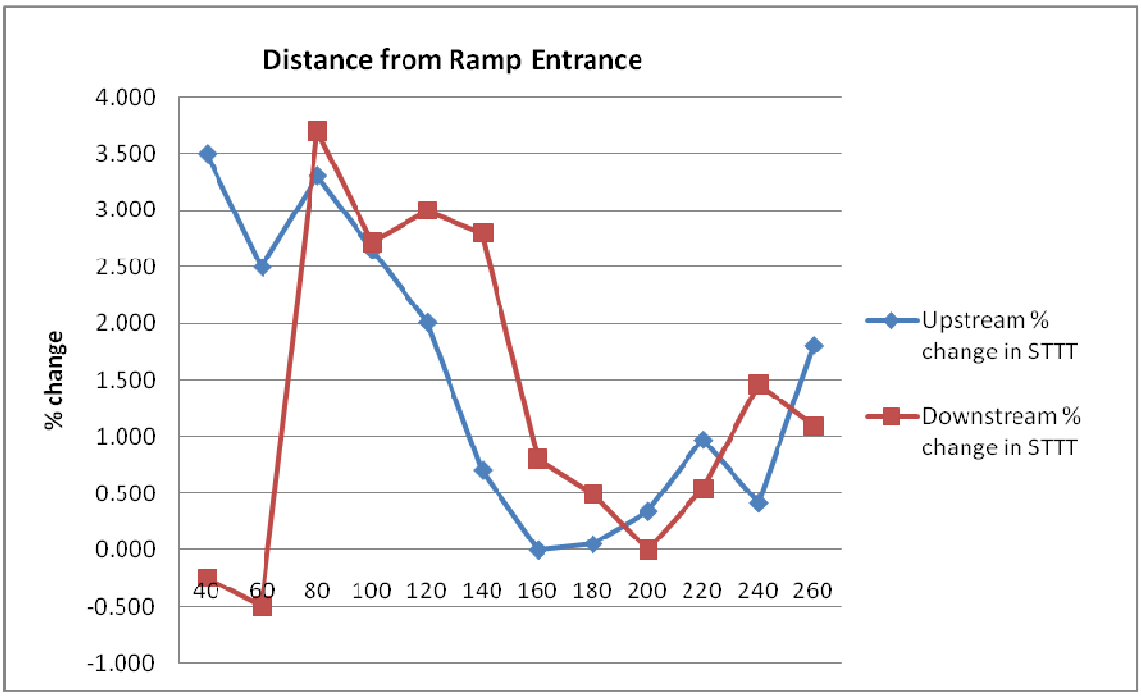


Figure 7.1 Percentage of change in TTT vs. Positions of upstream/downstream detector

7.2 Minimum Green Time Variations

This 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 will be 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 Figure 7.2, the optimum minimum green time is around 6 seconds.

Table 7.2 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

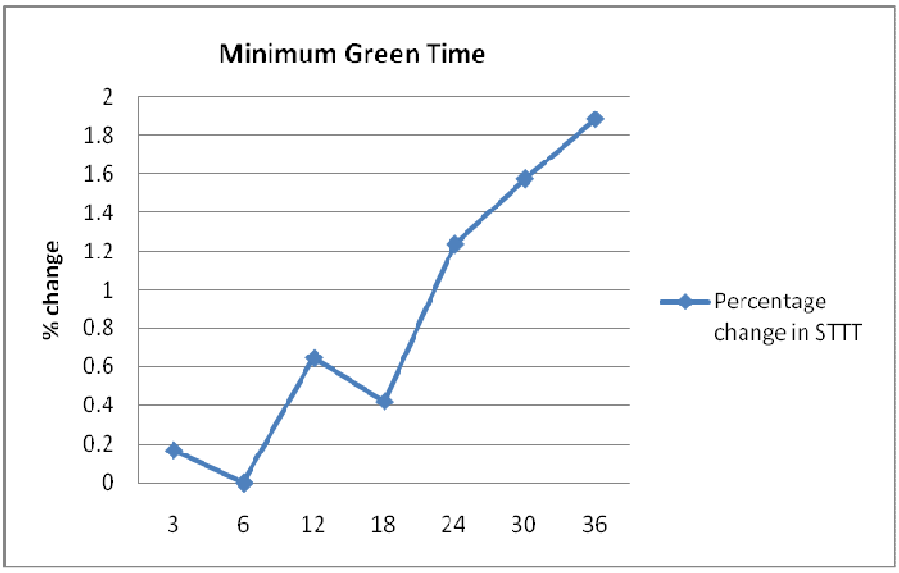


Figure 7.2 Percentage of change in STTT vs. Minimum Green Time

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 Conclusion

Currently there are only a few control models developed for Diamond Interchange such as Dynamic Programming model developed by Fang (2004). This thesis proposed one of the world first adaptive approaches to design a Diamond Interchange. The basic of Fuzzy Logic Diamond Interchange Control (FLDI) algorithm is "to model control strategy based on human expert knowledge rather than model of the process itself" (Niittymaki, 2001).

Simulation experiments were carried out to compare the performance of the FLDI controller with Actuated Interchange Control with No Ramp and Actuated Interchange Control with Fuzzy Ramp Metering. The flow rate of the simulation is varied according to real life traffic conditions.

It can be observed from the simulation results showed that the developed FLDI provides better performance in terms of total travel time, downstream average speed and average delay. Less delay and travel time will not only reduce the fuel consumption but also reduce air and noise pollution. On the other hand the FLDI adapted well to different traffic scenarios; this is because the FLDI is more flexible than the other two. The flexibility here involves the number of vehicles sensed at the incoming intersections (Constellation Drive and Upper Harbour Highway) and the extension of the green time with different extension range for each particular phase. The actuated controller is actually an enhanced version of fixed-time controller, the green time is extended whenever there is a presence of a vehicle and these times are fixed in advance up to a maximum time limit. For example when a car is detected, the green time is extended 5-10 seconds until the maximum time limit is reached. In the FLDI controller, the extension time is not fixed values. They are all fuzzy variables such as very short, short, medium and long. The numbers of cars detected at the input of the fuzzy controllers are also converted to fuzzy values, such as few, small, medium and many. Most importantly, the fuzzy controller has the advantage if performing according to linguistic rules base on human expertise. The other advantage of the FLDI model

over the two other controllers is the spill-back prevention algorithm, this algorithm is designed to increase the metering flow rate to the maximum when there are many cars waiting to get onto the ramp from the connected intersection and also when the queue on the ramp is too long and may lead to the blockage of the connected intersection. Moreover, the FLDI also consider car sizes such as automobile, larger cars and trucks.

8.2 Future Work

A new Fuzzy Logic Diamond Interchange Control algorithm may be developed by considering the pedestrians, 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.

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

REFERENCES

Aimsun6 API Manual, TSS-Transport Simulation Systems, Spain, 2008.

Barceló J., Casas J., Ferrer J. L., and García D., "Modelling Advanced Transport Telematic Applications with Microscopic Simulators: the Case of AIMSUN2", Paper presented at the Fourth International Conference on Applications of Advanced Technologies in Transportation Engineering, Capri, Italy, 27-30 June 1995.

Bell, M.G.H., "A probability approach to the optimization of traffic signal systems in discrete time", Transportation and Traffic Theory, Elsevier, London, 1990, pp. 619-633.

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

Bogenberger, K., Vukanovic, S., Keller, H., "A Neuro-Fuzzy Algorithm for Coordinated Traffic Responsive Ramp Metering," IEEE 4th International Conference on Intelligent Transportation Systems, California, 2001, pp. 2-4.

Bonneson J.A., and Lee S., "Actuated Controller Settings for the Diamond Interchanges with Three-Phase Operation", TTI/ITS RCE-01/01, TTI, College State, Texas, 2000, pp. 53.

Chen L., May A., "Freeway Ramp Control Using Fuzzy Set Theory for Inexact Reasoning", Institute of Transportation Studies, UC Berkeley, USA, 1988.

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

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

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

Highway Capacity Manual 2000 (HCM), Transportation Research Board, National Research Council, Washington DC, 2000.

Ho T. K., “Fuzzy logic traffic control at a road junction with time-varying flow rates”, Electronics Letters, Vol. 32, August 1996, pp. 6-8.

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

Hughes, J, “AIMSUN2 Simulation of a Congested Auckland Freeway”, Master thesis at The University of Auckland, New Zealand, 2000.

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.

Joseph, R., “Evaluation of Coordinated and Local Ramp Metering Algorithms using Microscopic Traffic Simulation”, Master Thesis, University of Rhode Island, 2003.

Lin F. B, Cooke D., and Vijayakumar. S., “Use of Predicted Vehicle Arrival Information for Adaptive Signal Control", Transportation Research Record (1112), Transportation Research Board, Washington, D.C., 1987, pp. 89-98.

M. D. T., “Evaluation and Improvement of the Stratified Ramp Metering Algorithm Through Microscopic Simulation - Phase II”- 2005-48 Final Report for Minnesota Department of Transportation.

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

Microsimulator and Mesosimulator in Aimsun 6 User's Manual, TSS-Transport Simulation Systems, Spain, 2008.

Miller, A. J., "A Computer Control System for Traffic Networks", 1963, pp. 200-220.

Munjal, P.K., "An Analysis of Diamond Interchange Signalization", HRB, Highway Research Board 349, 1971, pp. 47-64.

Niittymaki J., "Installation and experiences of field testing a fuzzy signal controller", European Journal of Operation Research 131, Volume 131, Issue 2, pp. 273-281, 1 June 2001.

Ramp Metering Design Guidelines, California Department of Transportation, January 1991.

Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D. Saisana, M., and Tarantola, S., "Global Sensitivity Analysis", The Primer, John Wiley & Sons, 2008.

Shaflik C., "Traffic signal detector locations – Proper positioning increases efficiency & safety", Technical paper prepared for CIVIL 589 – Traffic Flow Theory, Department of Civil Engineering, University of British Columbia, US, pp. 8-9, 1995

State Highway Traffic Data Booklet 2003-2008, Transit New Zealand, January 2009.

Taale H., Slager J., Rosloot J., "The Assessment of Ramp Metering Based on Fuzzy Logic", 3d ITS World Congress in Orlando, 1996, pp. 1-6.

Taale, H., Middelham, F., "Ten years of ramp-metering in the Netherlands", Road Transport Information and Control, Tenth International Conference on (Conf. Publ. No. 472), London, UK, 2000, pp. 3-5.

Tan K. K., Khalid M., Yusof R., "Intelligent Traffic Lights Control by Fuzzy Logic", Malaysian Journal of Computer Science, Vol. 9, No. 2, Dec. 1996, pp. 29-35.

Trabia M.B., M.S. Kaseko and M. Ande., "A two-stage fuzzy logic controller for traffic signals", Transportation Research Part C 7, 1999, pp. 353-367.

Traffic signal phasing selection manual, Austroads, Australia, September 2009.

Taylor C., Meldrum D., Jacobson L., "Fuzzy Ramp Metering - Design Overview and Simulation Results", Transportation Research Record 1634, Washington, 1998, pp. 4-10.

Taylor C., and Meldrum D., "Algorithm Design, User Interface, and Optimization Procedure for a Fuzzy Logic Ramp Metering Algorithm: A Training Manual for Freeway Operations Engineers," WA-RD Technical Report to be published, Washington State Department of Transportation, National Technical Information Service, 2000.

Pappis C. P., and Mamdani E. H., "A Fuzzy Logic Controller for a Traffic Junction", IEEE Transactions System, Man and Cybernetics, Vol.SMC-7, No.10, October 1977, pp. 707-717.

Pearce, V., "Real-time Remedy", Traffic Technology International, January 2001, pp. 61-63.

APPENDIX A

C++ CODE FOR FUZZY LOGIC DIAMOND INTERCHANGES

1. AAPI.cxx (Main file)

```
// SIGNALISED FUZZY LOGIC FOR A DIAMOND INTERCHANGE
// VAN C. PHAM - MASTER OF ENGINEERING IN MECHATRONICS
// SEAT - MASSEY UNIVERSITY - AUCKLAND - NEW ZEALAND

// Fuzzy Logic Phase Timming C++ Coding
// 2008-2009 - All rights reserved

#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, 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, 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. FLCD.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 Darr_veh_few (float Darr_veh_few) /*initial value 0 and 19*/
{             if (Darr_veh_few<0) {
                    return (0);}
              if (Darr_veh_few>19) {
                    return (0);}
              if (Darr_veh_few>=-19 && Darr_veh_few<=0) {
                    return (Darr_veh_few/19 +1);}
              if (Darr_veh_few>0 && Darr_veh_few <=19) {
                    return ((-1)*Darr_veh_few/19 + 1);}
       return (-1);
}


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


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


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

```
// Input 2 definition - Phase D Queue Membership functions

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


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


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


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

 float DflcExtensionRate (float Darr_veh, float Dq_veh)

{
   // Evaluate each rule
         float rule[16];
         rule[0]=MIN(Darr_veh_few(Darr_veh),   Dq_veh_few(Dq_veh));
         rule[1]=MIN(Darr_veh_few(Darr_veh),   Dq_veh_small(Dq_veh));
         rule[2]=MIN(Darr_veh_few(Darr_veh),   Dq_veh_medium(Dq_veh));
         rule[3]=MIN(Darr_veh_few(Darr_veh),   Dq_veh_many(Dq_veh));
         rule[4]=MIN(Darr_veh_small(Darr_veh), Dq_veh_few(Dq_veh));
```

```
        rule[5]=MIN(Darr_veh_small(Darr_veh), Dq_veh_small(Dq_veh));
        rule[6]=MIN(Darr_veh_small(Darr_veh), Dq_veh_medium(Dq_veh));
        rule[7]=MIN(Darr_veh_small(Darr_veh), Dq_veh_many(Dq_veh));
        rule[8]=MIN(Darr_veh_medium(Darr_veh), Dq_veh_few(Dq_veh));
        rule[9]=MIN(Darr_veh_medium(Darr_veh), Dq_veh_small(Dq_veh));
        rule[10]=MIN(Darr_veh_medium(Darr_veh), Dq_veh_medium(Dq_veh));
        rule[11]=MIN(Darr_veh_medium(Darr_veh), Dq_veh_many(Dq_veh));
        rule[12]=MIN(Darr_veh_many(Darr_veh), Dq_veh_few(Dq_veh));
        rule[13]=MIN(Darr_veh_many(Darr_veh), Dq_veh_small(Dq_veh));
        rule[14]=MIN(Darr_veh_many(Darr_veh), Dq_veh_medium(Dq_veh));
        rule[15]=MIN(Darr_veh_few(Darr_veh), Dq_veh_many(Dq_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_ext_class[4];
         f_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
      f_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
      f_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
      f_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_ext_class[i]*area*centroid;
                 den+=f_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. FLCB.cxx (Fuzzy Logic B Phase Timing)

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


//input1 definition - Phase D Arrival Membership functions

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


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


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


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

```
// Input 2 definition - Phase D Queue Membership functions

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


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


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


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

 float BflcExtensionRate (float Barr_veh, float Bq_veh)

{
   // Evaluate each rule
         float rule[16];
         rule[0]=MIN(Barr_veh_few(Barr_veh), Bq_veh_few(Bq_veh));
         rule[1]=MIN(Barr_veh_few(Barr_veh), Bq_veh_small(Bq_veh));
         rule[2]=MIN(Barr_veh_few(Barr_veh), Bq_veh_medium(Bq_veh));
         rule[3]=MIN(Barr_veh_few(Barr_veh), Bq_veh_many(Bq_veh));
         rule[4]=MIN(Barr_veh_small(Barr_veh), Bq_veh_few(Bq_veh));
```

```
        rule[5]=MIN(Barr_veh_small(Barr_veh), Bq_veh_small(Bq_veh));
        rule[6]=MIN(Barr_veh_small(Barr_veh), Bq_veh_medium(Bq_veh));
        rule[7]=MIN(Barr_veh_small(Barr_veh), Bq_veh_many(Bq_veh));
        rule[8]=MIN(Barr_veh_medium(Barr_veh), Bq_veh_few(Bq_veh));
        rule[9]=MIN(Barr_veh_medium(Barr_veh), Bq_veh_small(Bq_veh));
        rule[10]=MIN(Barr_veh_medium(Barr_veh), Bq_veh_medium(Bq_veh));
        rule[11]=MIN(Barr_veh_medium(Barr_veh), Bq_veh_many(Bq_veh));
        rule[12]=MIN(Barr_veh_many(Barr_veh), Bq_veh_few(Bq_veh));
        rule[13]=MIN(Barr_veh_many(Barr_veh), Bq_veh_small(Bq_veh));
        rule[14]=MIN(Barr_veh_many(Barr_veh), Bq_veh_medium(Bq_veh));
        rule[15]=MIN(Barr_veh_few(Barr_veh), Bq_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_ext_class[4];
         f_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
      f_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
      f_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
      f_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_ext_class[i]*area*centroid;
                 den+=f_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. FLCF.cxx (Fuzzy Logic F Phase Timing)

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


//input1 definition - Phase D Arrival Membership functions

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


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


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


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

```
// Input 2 definition - Phase D Queue Membership functions

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


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


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


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

 float FflcExtensionRate (float Farr_veh, float Fq_veh)

{
   // Evaluate each rule
         float rule[16];
         rule[0]=MIN(Farr_veh_few(Farr_veh), Fq_veh_few(Fq_veh));
         rule[1]=MIN(Farr_veh_few(Farr_veh), Fq_veh_small(Fq_veh));
         rule[2]=MIN(Farr_veh_few(Farr_veh), Fq_veh_medium(Fq_veh));
         rule[3]=MIN(Farr_veh_few(Farr_veh), Fq_veh_many(Fq_veh));
         rule[4]=MIN(Farr_veh_small(Farr_veh), Fq_veh_few(Fq_veh));
```

```
        rule[5]=MIN(Farr_veh_small(Farr_veh), Fq_veh_small(Fq_veh));
        rule[6]=MIN(Farr_veh_small(Farr_veh), Fq_veh_medium(Fq_veh));
        rule[7]=MIN(Farr_veh_small(Farr_veh), Fq_veh_many(Fq_veh));
        rule[8]=MIN(Farr_veh_medium(Farr_veh), Fq_veh_few(Fq_veh));
        rule[9]=MIN(Farr_veh_medium(Farr_veh), Fq_veh_small(Fq_veh));
        rule[10]=MIN(Farr_veh_medium(Farr_veh), Fq_veh_medium(Fq_veh));
        rule[11]=MIN(Farr_veh_medium(Farr_veh), Fq_veh_many(Fq_veh));
        rule[12]=MIN(Farr_veh_many(Farr_veh), Fq_veh_few(Fq_veh));
        rule[13]=MIN(Farr_veh_many(Farr_veh), Fq_veh_small(Fq_veh));
        rule[14]=MIN(Farr_veh_many(Farr_veh), Fq_veh_medium(Fq_veh));
        rule[15]=MIN(Farr_veh_few(Farr_veh), Fq_veh_many(Fq_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_ext_class[4];
         f_ext_class[0]=rule[0]+rule[1]+rule[2]+rule[3]+rule[6]+rule[7];
      f_ext_class[1]=rule[4]+rule[5]+rule[10]+rule[11]+rule[15];
      f_ext_class[2]=rule[8]+rule[9]+rule[13]+rule[14];
      f_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_ext_class[i]*area*centroid;
                 den+=f_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 (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 Darr_veh_few(float);
float Darr_veh_small(float);
float Darr_veh_medium(float);
float Darr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float Dq_veh_few(float);
float Dq_veh_small(float);
float Dq_veh_medium(float);
float Dq_veh_many(float);

//FLC declaration

float DflcExtensionRate (float, float);
```

6. FLCB.h (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 Barr_veh_few(float);
float Barr_veh_small(float);
float Barr_veh_medium(float);
float Barr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float Bq_veh_few(float);
float Bq_veh_small(float);
float Bq_veh_medium(float);
float Bq_veh_many(float);

//FLC declaration

float BflcExtensionRate (float, float);
```

7. FLCF.h (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 Farr_veh_few(float);
float Farr_veh_small(float);
float Farr_veh_medium(float);
float Farr_veh_many(float);

//input2: (q_veh) queueing vehciles declaration
float Fq_veh_few(float);
float Fq_veh_small(float);
float Fq_veh_medium(float);
float Fq_veh_many(float);

//FLC declaration

float FflcExtensionRate (float, float);
```

APPENDIX B
C++ CODE FOR FUZZY LOGIC RAMP METERING

1. AAPI.cxx (Main file)

```
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include <stdio.h>
#include "FLC.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. FLC.h (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 local_speed_med(float, float=32, float=75);
float local_speed_high(float, float=32, float=150);
float local_speed_low(float, float=32, float=0);
//input2 declaration
float local_flow_med(float, float=850, float=2000);
float local_flow_high(float, float=850, float=4000);
float local_flow_low(float, float=850, float=0);
//input3 declaration
float local_occ_med(float, float=6.4, float=15);
float local_occ_high(float, float=6.4, float=30);
float local_occ_low(float, float=6.4, float=0);
//input4 delcaration
float downstream_vc_high(float, float=6.5, float=0.5);
//input5 delcaration
float downstream_speed_low(float, float=-0.25, float=65);
//input6 delcaration
float checkin_occ_high(float, float=0.4, float=20);
//input7 declaration
float queue_occ_high(float, float=0.4, float=20);
//input8 declaration
float int_queue_occ_high(float, float=0.12, float=60);
//FLC declaration
float flcMeterRate(float *, float *, float *, float *, float *, float *, float *,
float*);
```

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

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

//input1 definition

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

//input2 definition

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

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

//input3 definition

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

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

//input4 definition

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



//input5 definition

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

//input6 definition

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

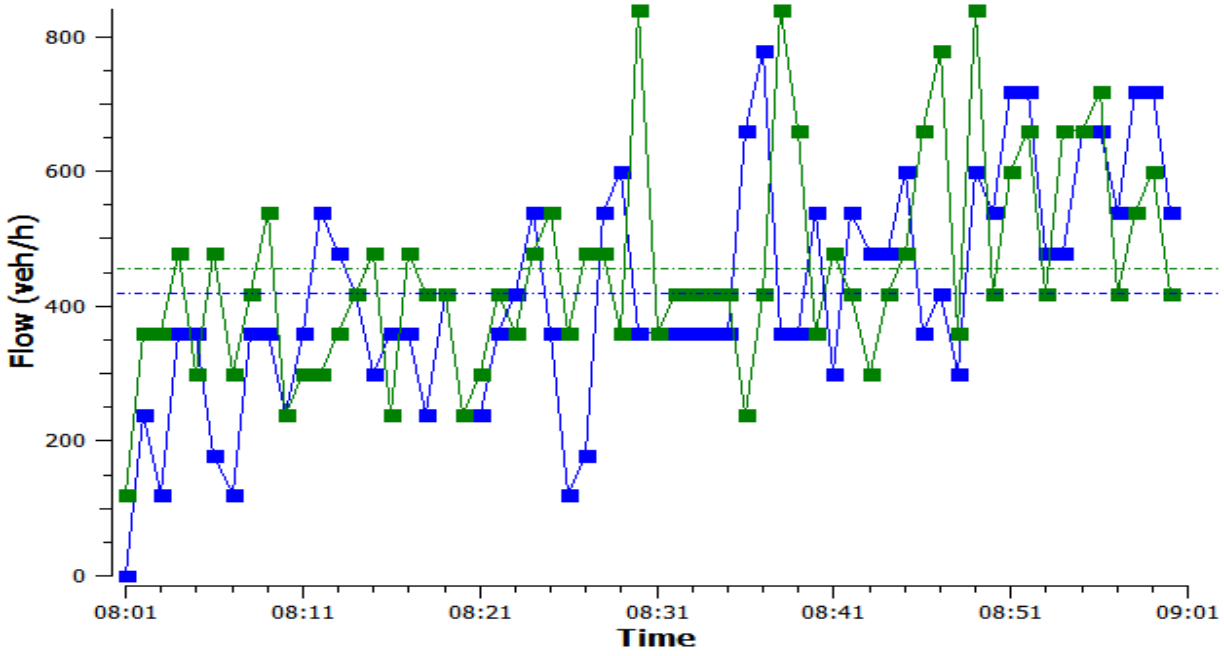
```
//input7 definition

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

//input 8 definition - Interchange Queue Occupancy

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

// FLC function

float flcMeterRate(float *local_occ, float *local_speed, float *local_flow, float
*downstream_vc, float *downstream_speed, float *checkin_occ, float *queue_occ, float
*int_queue_occ)
{

//RULES EVALUATION (10 rules)

float rule[10];
rule[0]=local_occ_low(*local_occ,*(local_occ+1),*(local_occ+2));
rule[1]=local_occ_med(*local_occ,*(local_occ+3),*(local_occ+4));
rule[2]=local_occ_high(*local_occ,*(local_occ+5),*(local_occ+6));
rule[3]=MIN(local_speed_low(*local_speed,*(local_speed+1),*(local_speed+2)),local_flow
_high(*local_flow,*(local_flow+5),*(local_flow+6)));
rule[4]=MIN(local_speed_med(*local_speed,*(local_speed+3),*(local_speed+4)),local_occ_
high(*local_occ,*(local_occ+5),*(local_occ+6)));
rule[5]=MIN(local_speed_med(*local_speed,*(local_speed+3),*(local_speed+4)),local_occ_
low(*local_occ,*(local_occ+1),*(local_occ+2)));
rule[6]=MIN(local_speed_high(*local_speed,*(local_speed+5),*(local_speed+6)),local_flo
w_low(*local_flow,*(local_flow+1),*(local_flow+2)));
rule[7]=MIN(downstream_speed_low(*downstream_speed,*(downstream_speed+1),*(downstream_
speed+2)),downstream_vc_high(*downstream_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)));
```

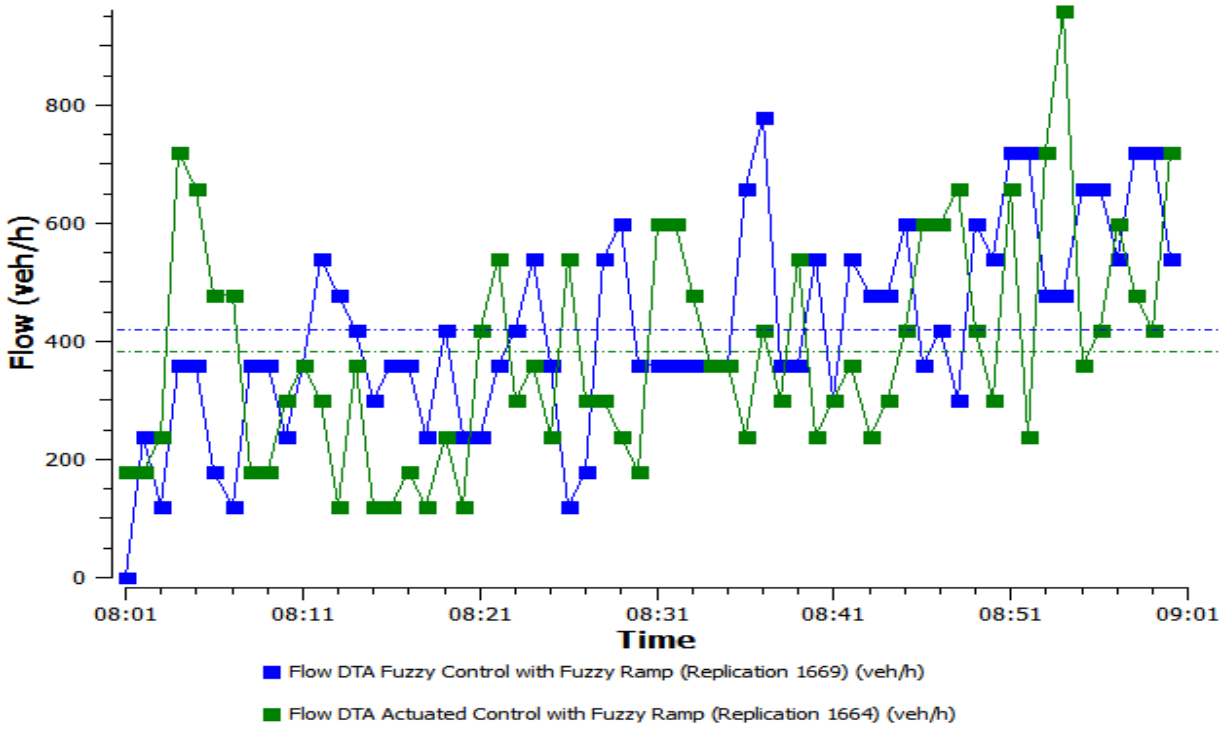
```
//The weighted sum of each rule class outcome

//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 C: COMPARISONS BETWEEN FLDI, ADINM AND ADIFM

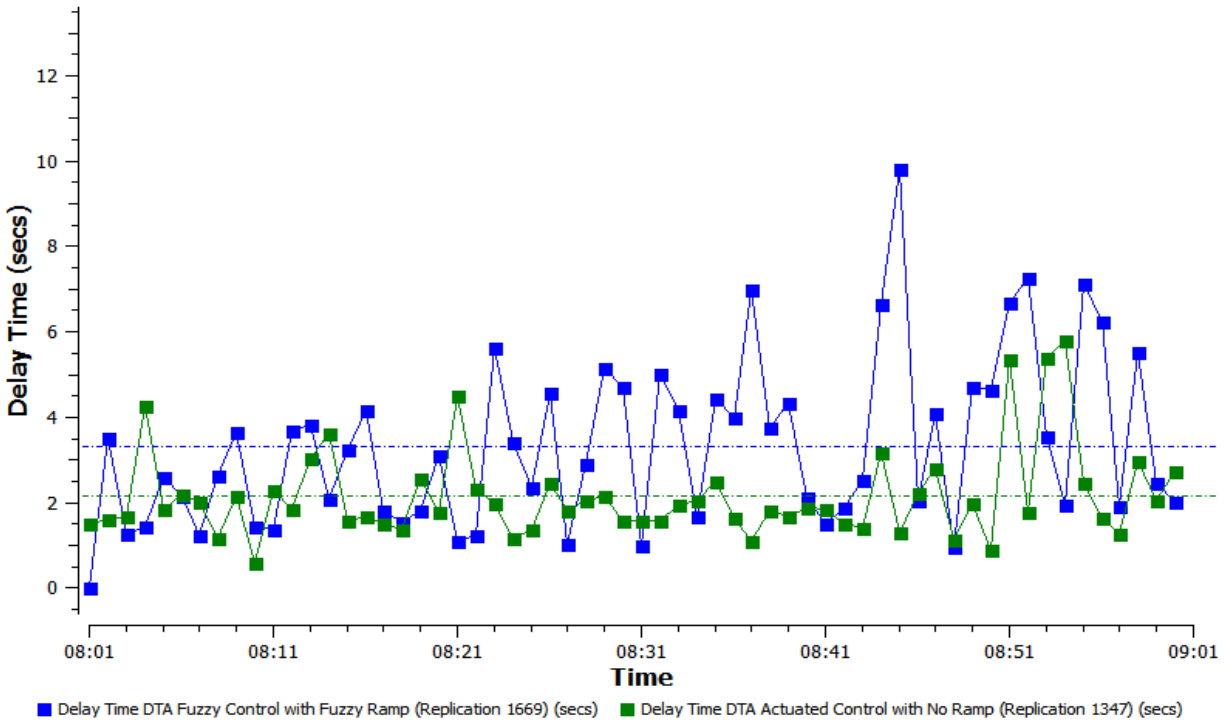
SCENARIO 1 RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS

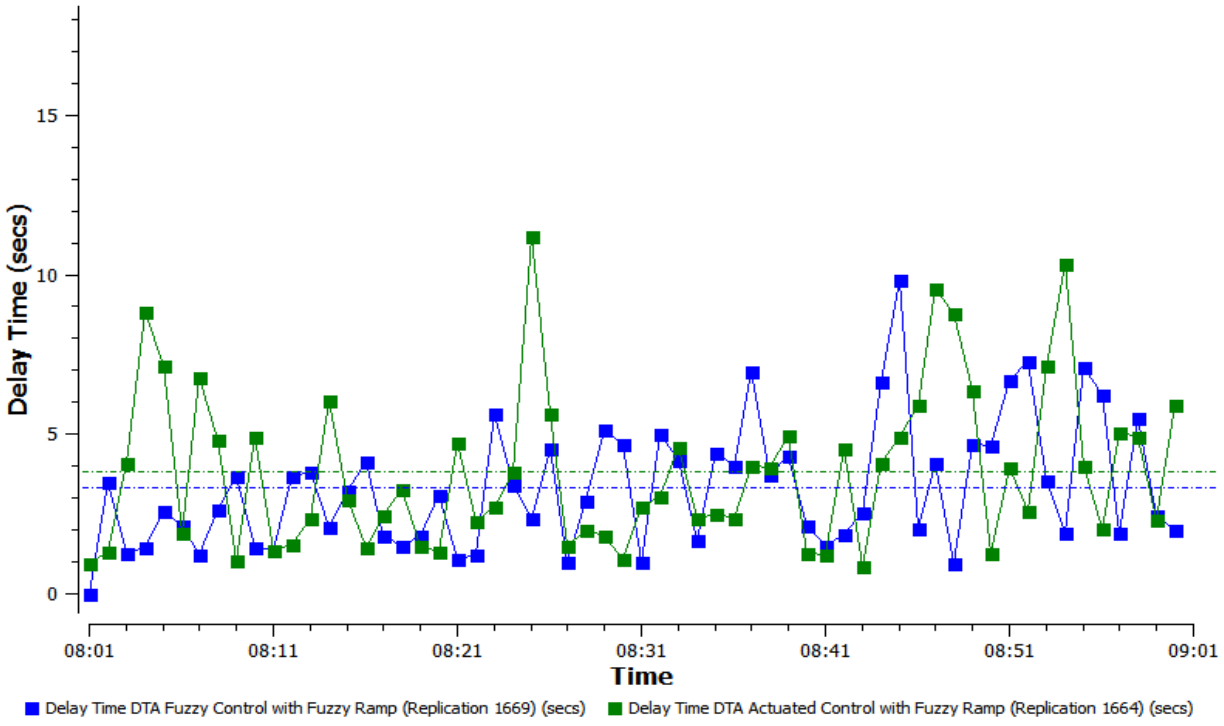
AVERAGE FLOW



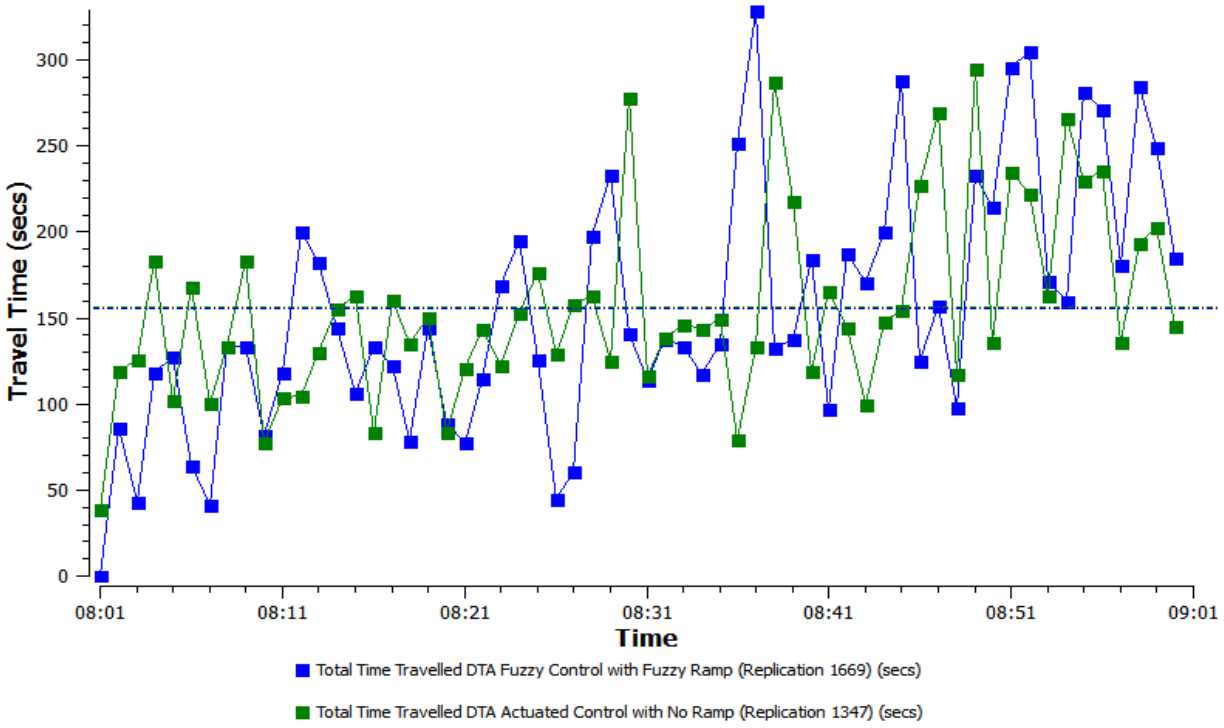


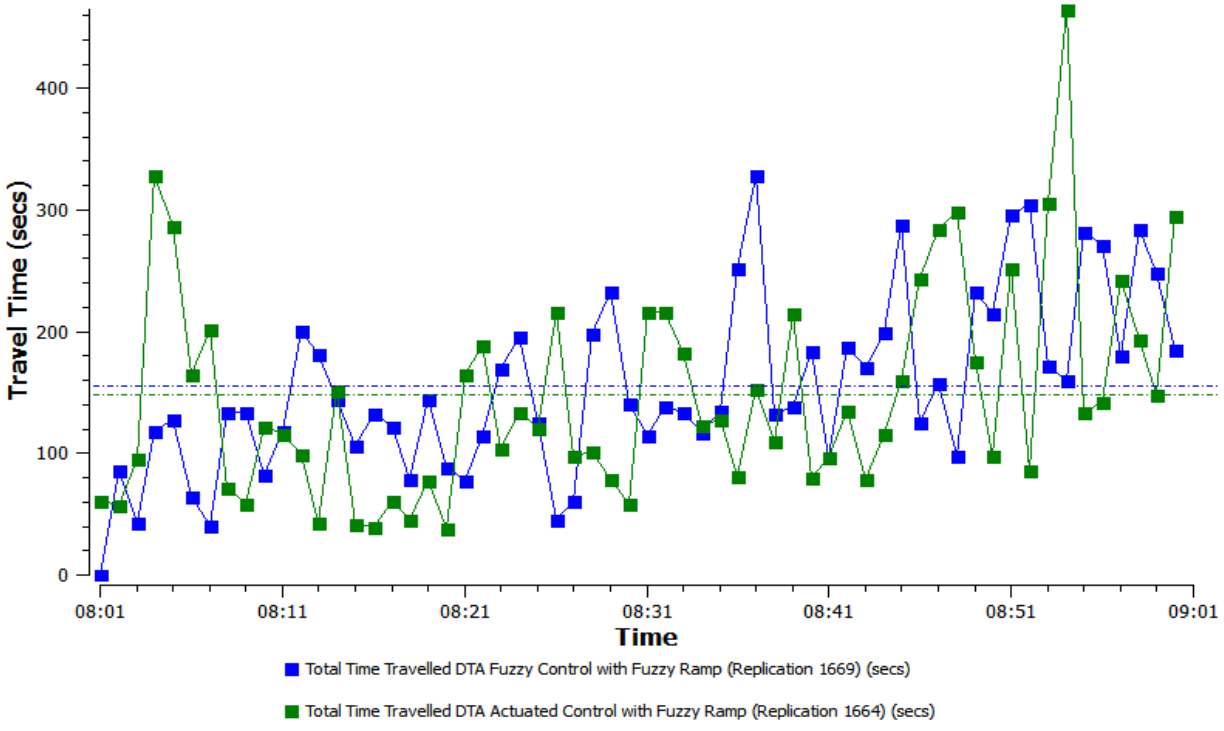
SCENARIO 1
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



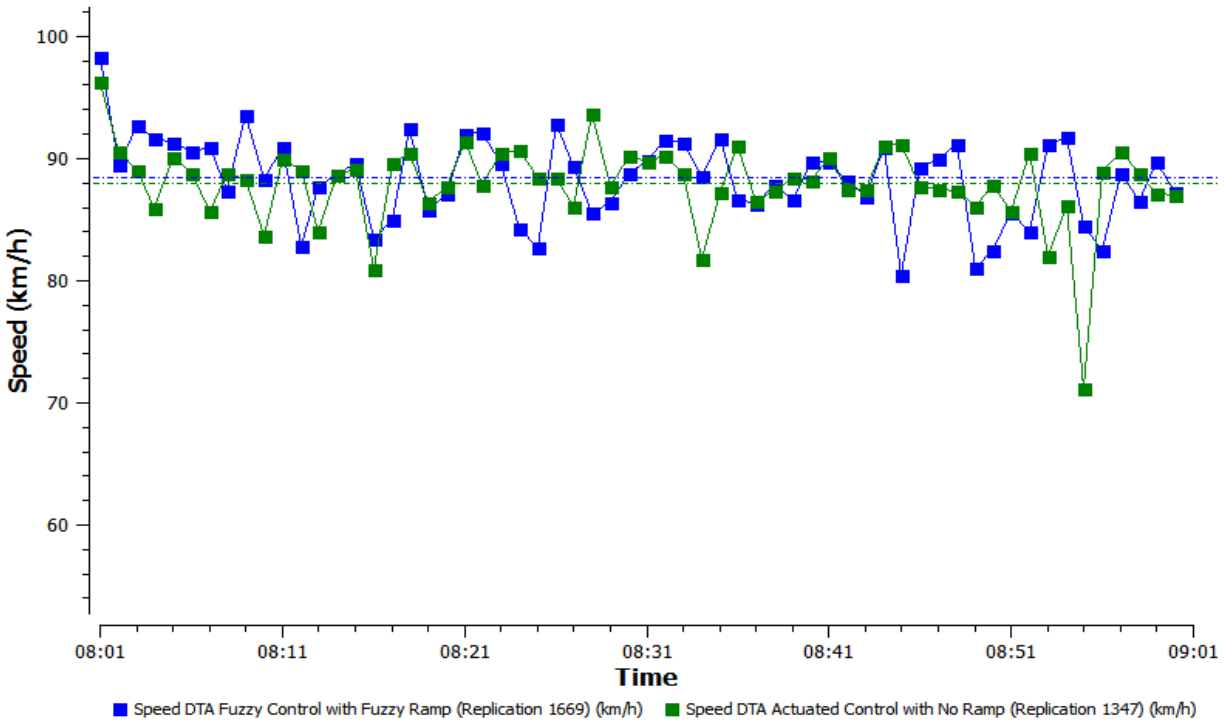


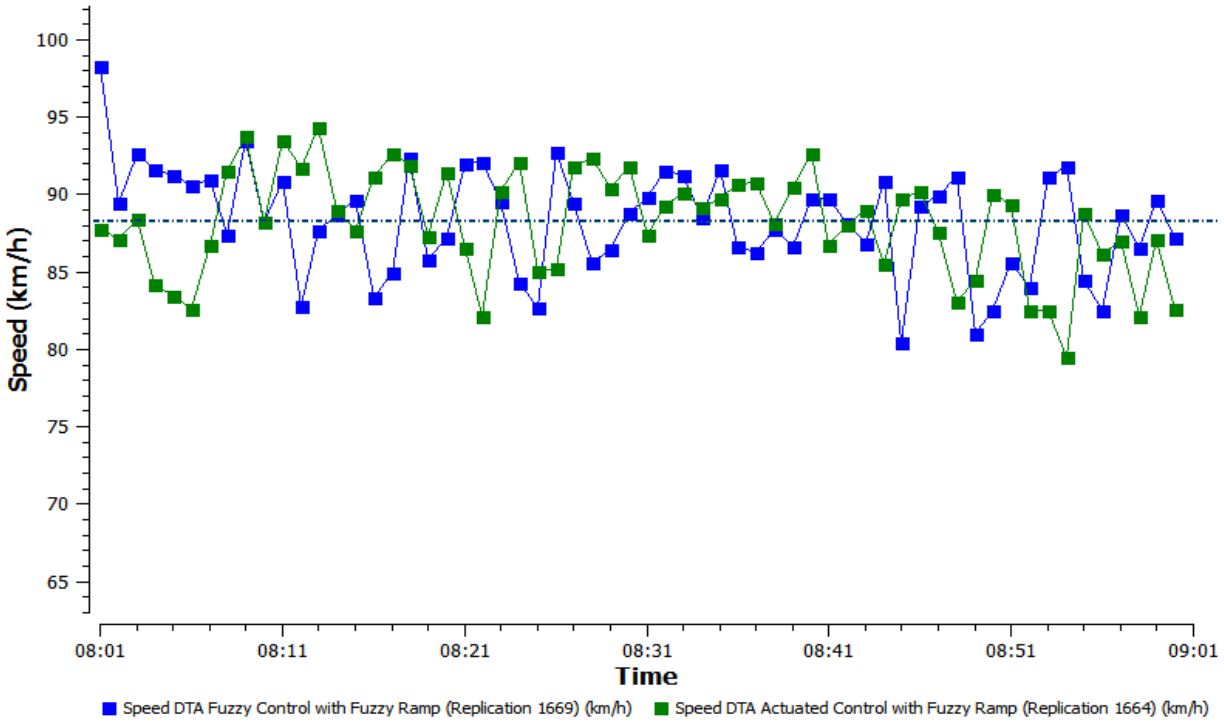
SCENARIO 1
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



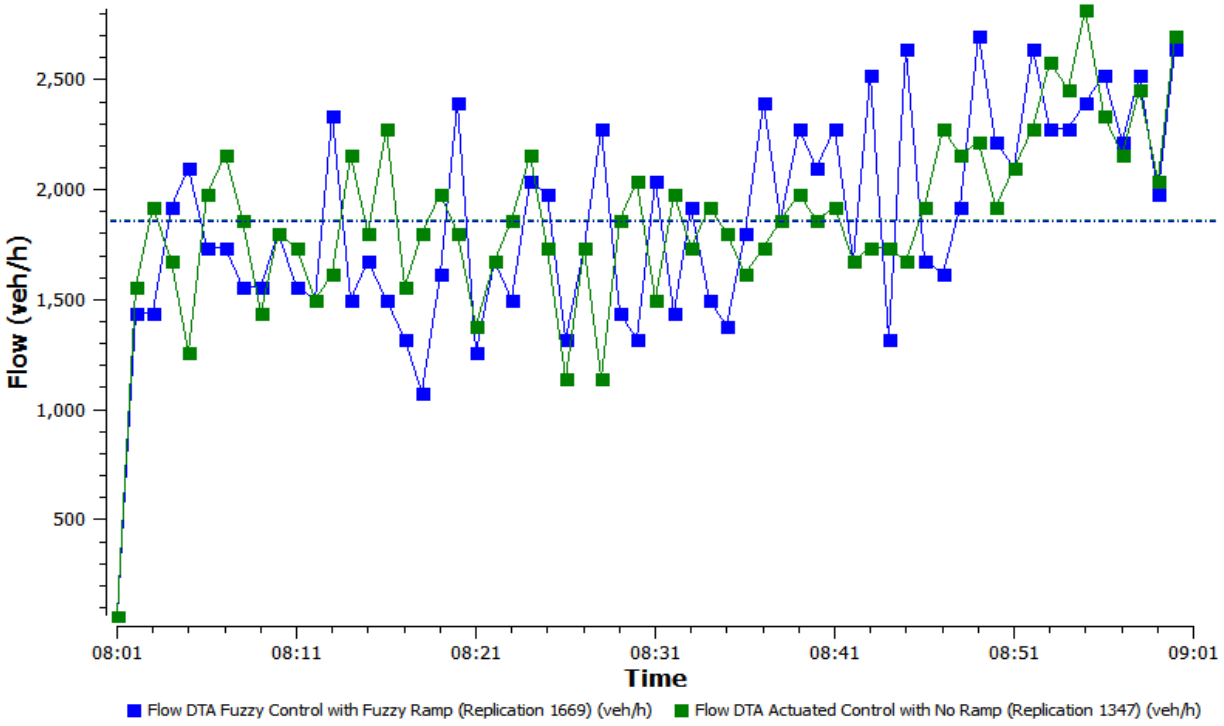


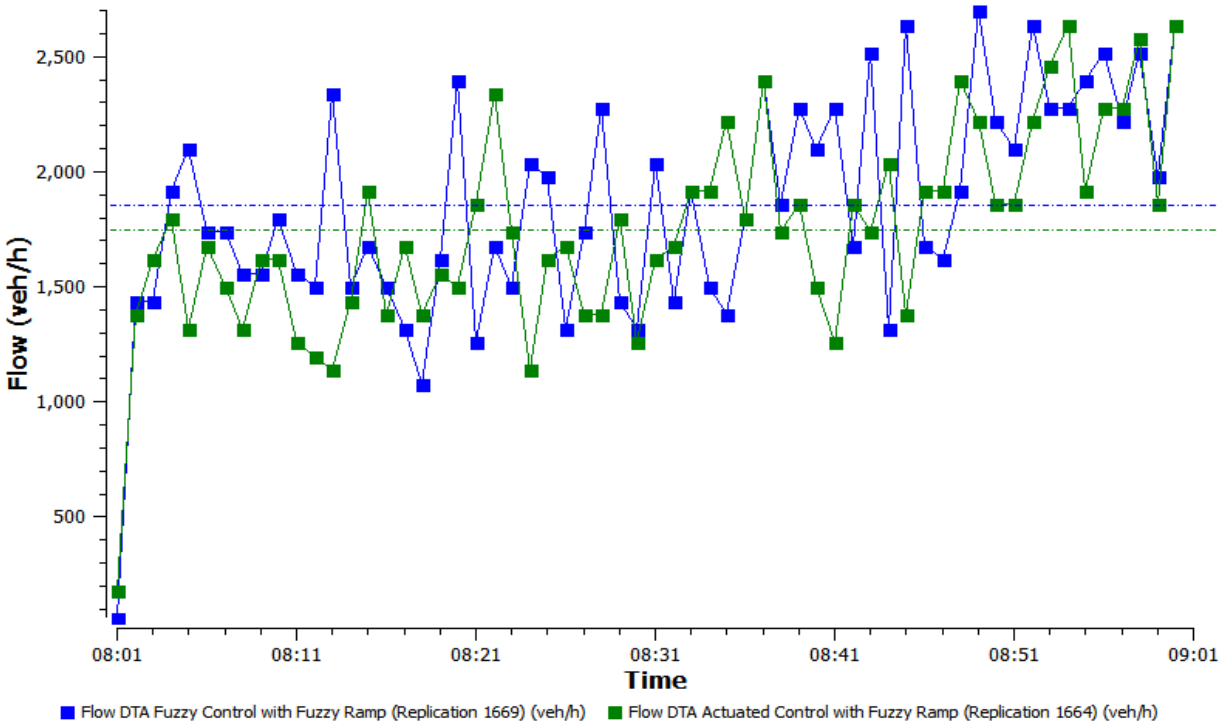
SCENARIO 1
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE SPEED



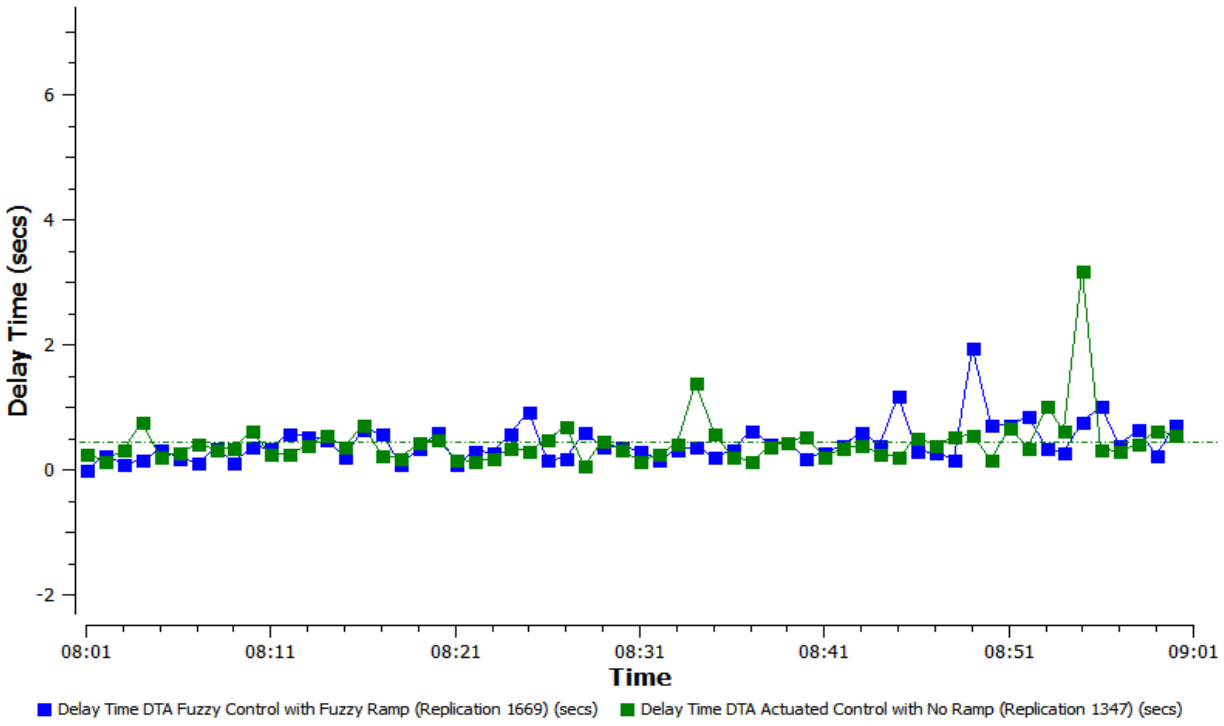


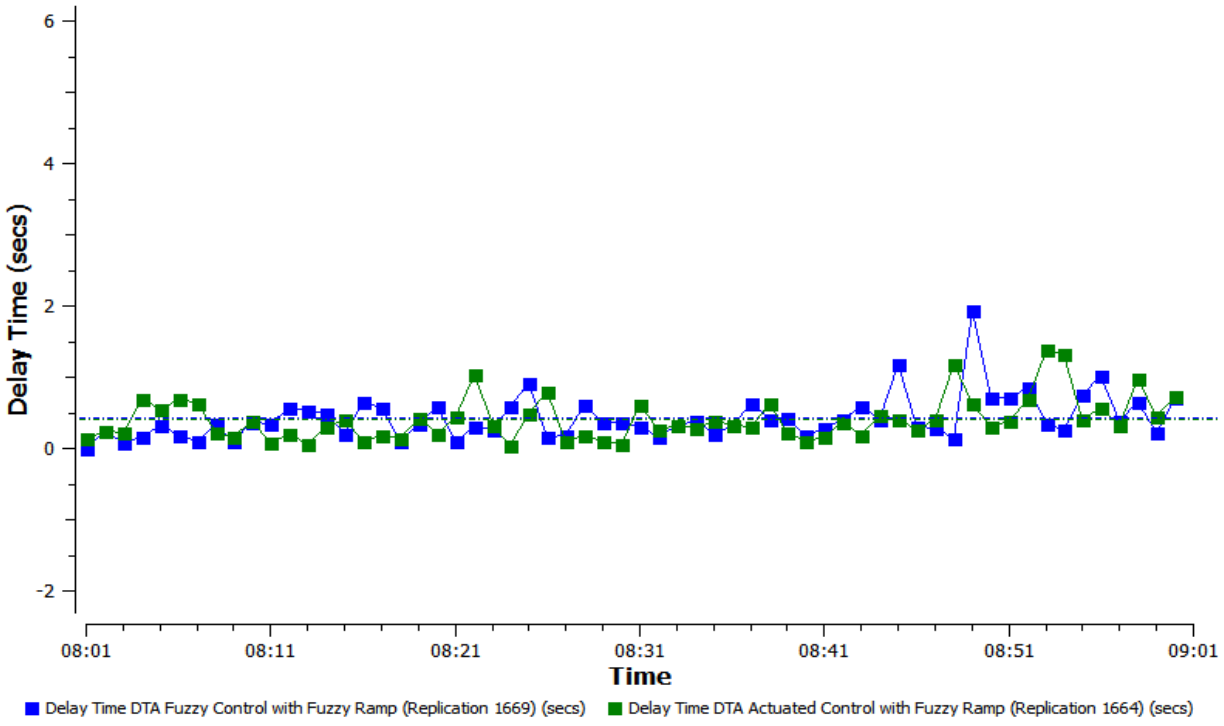
SCENARIO 1
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



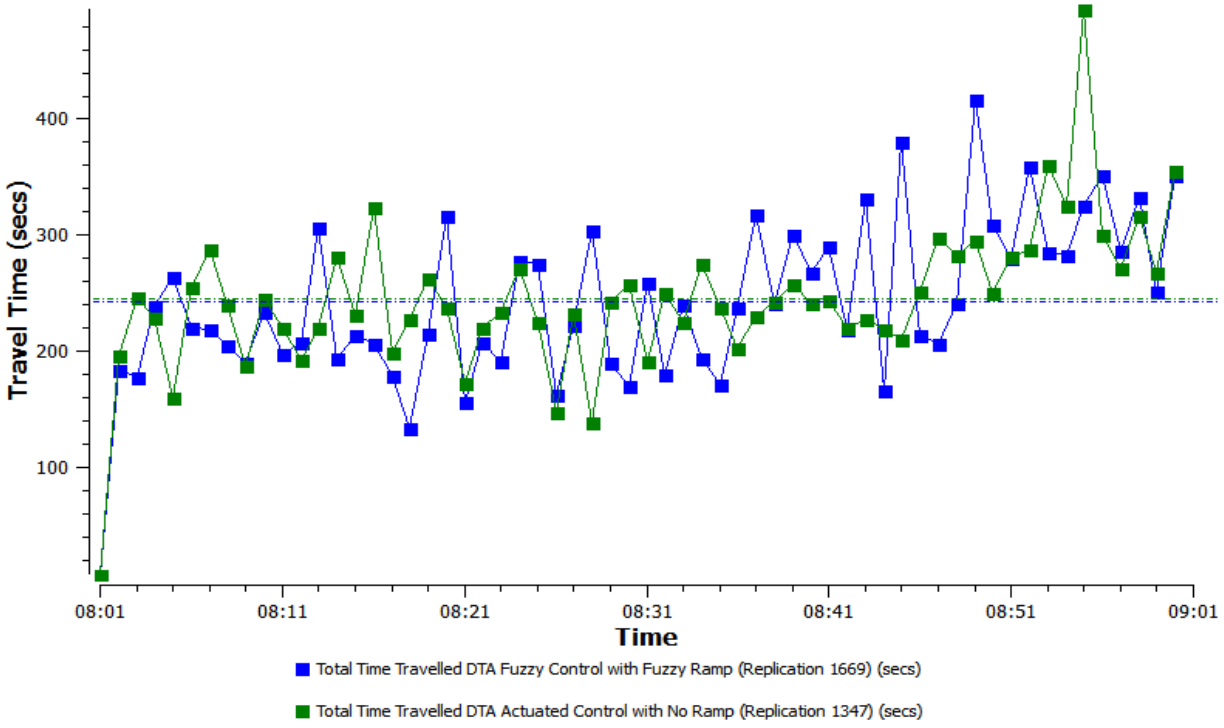


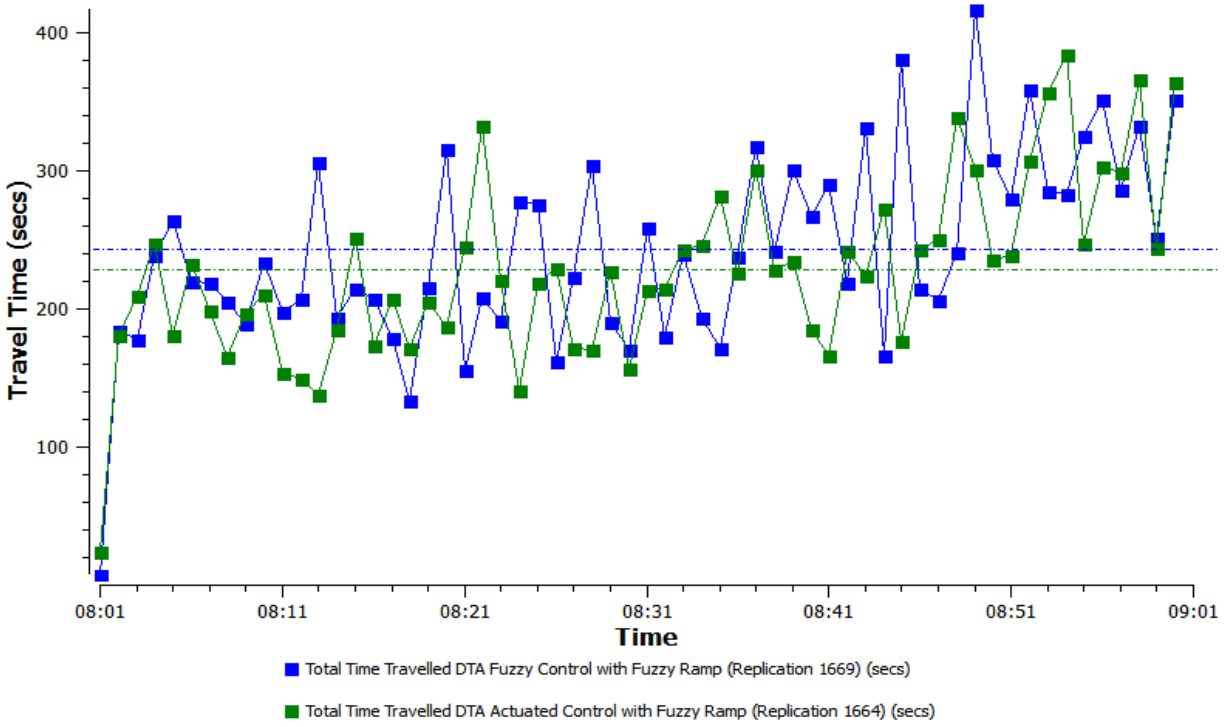
SCENARIO 1
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



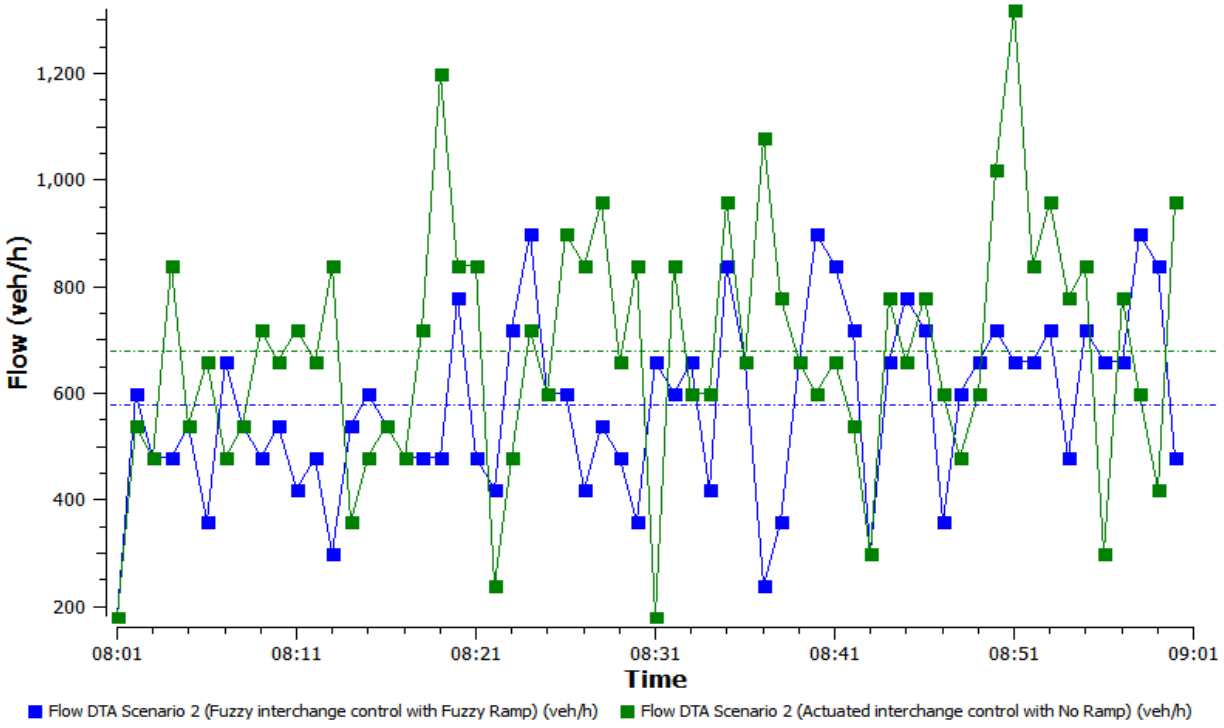


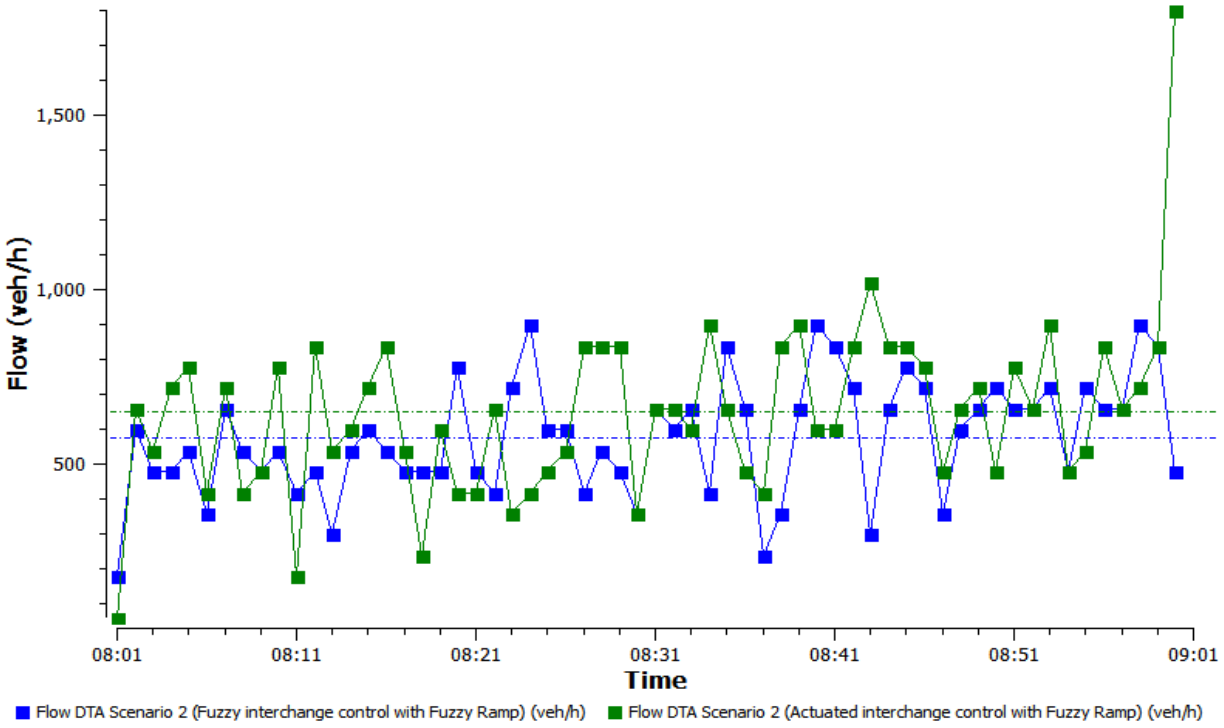
SCENARIO 1
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



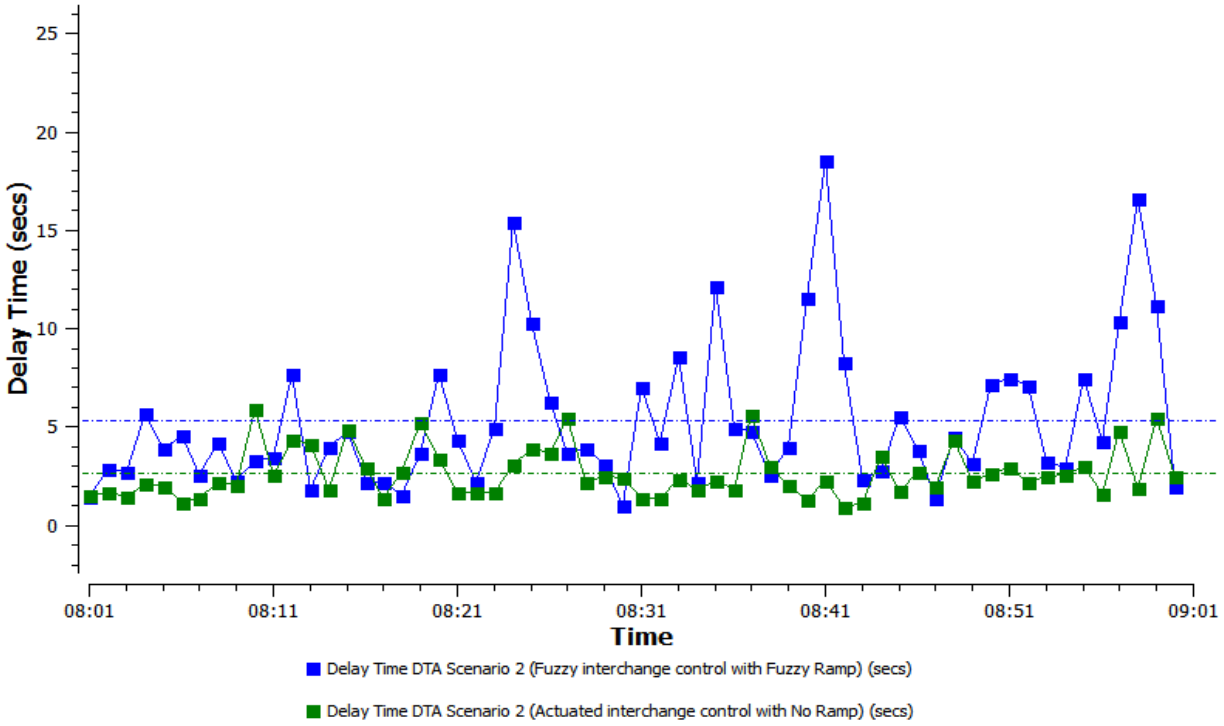


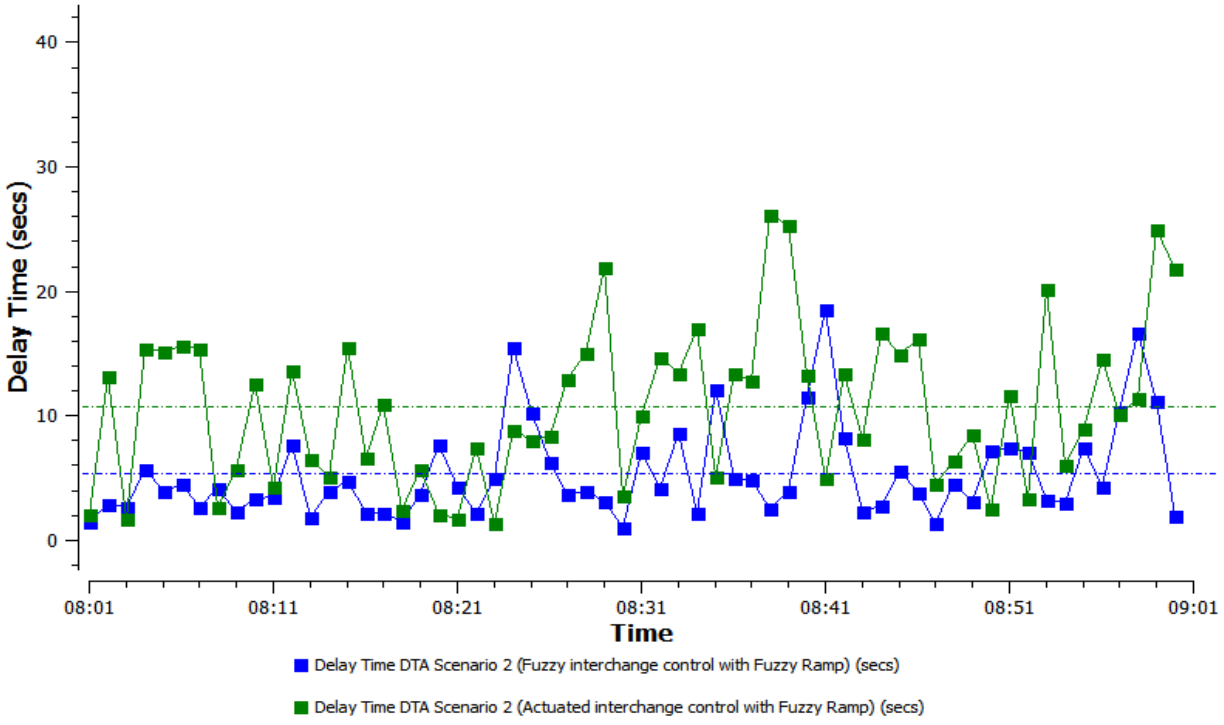
SCENARIO 2
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



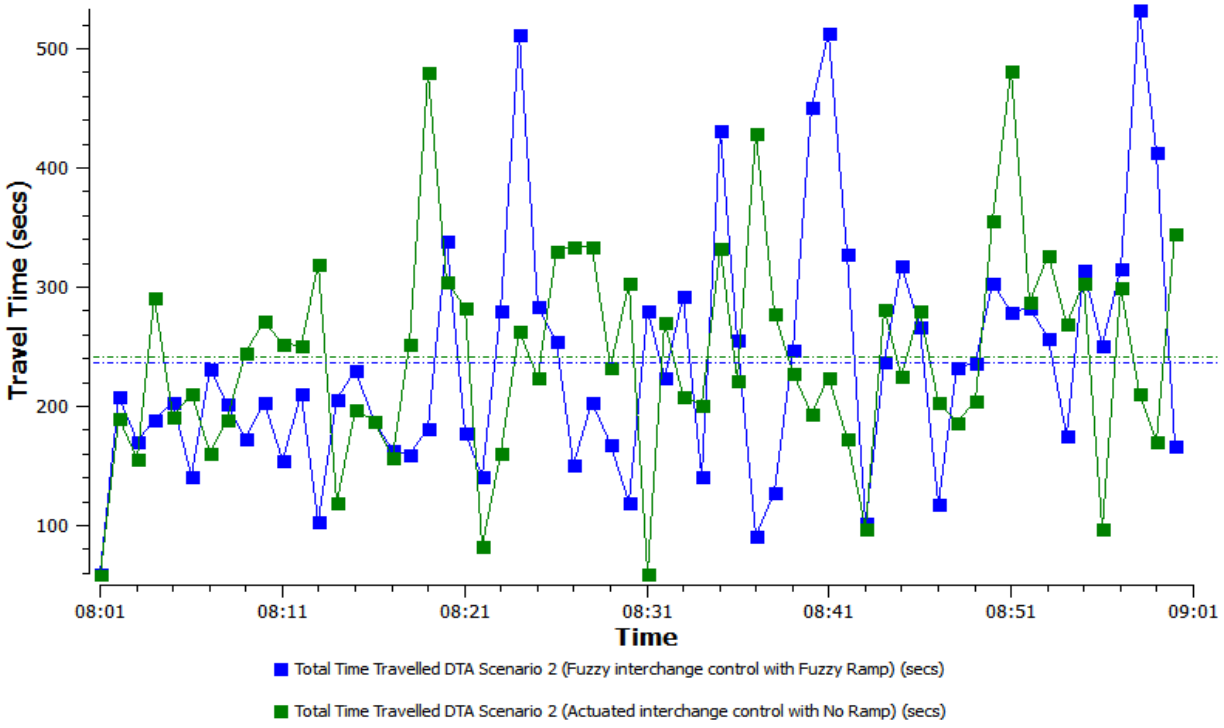


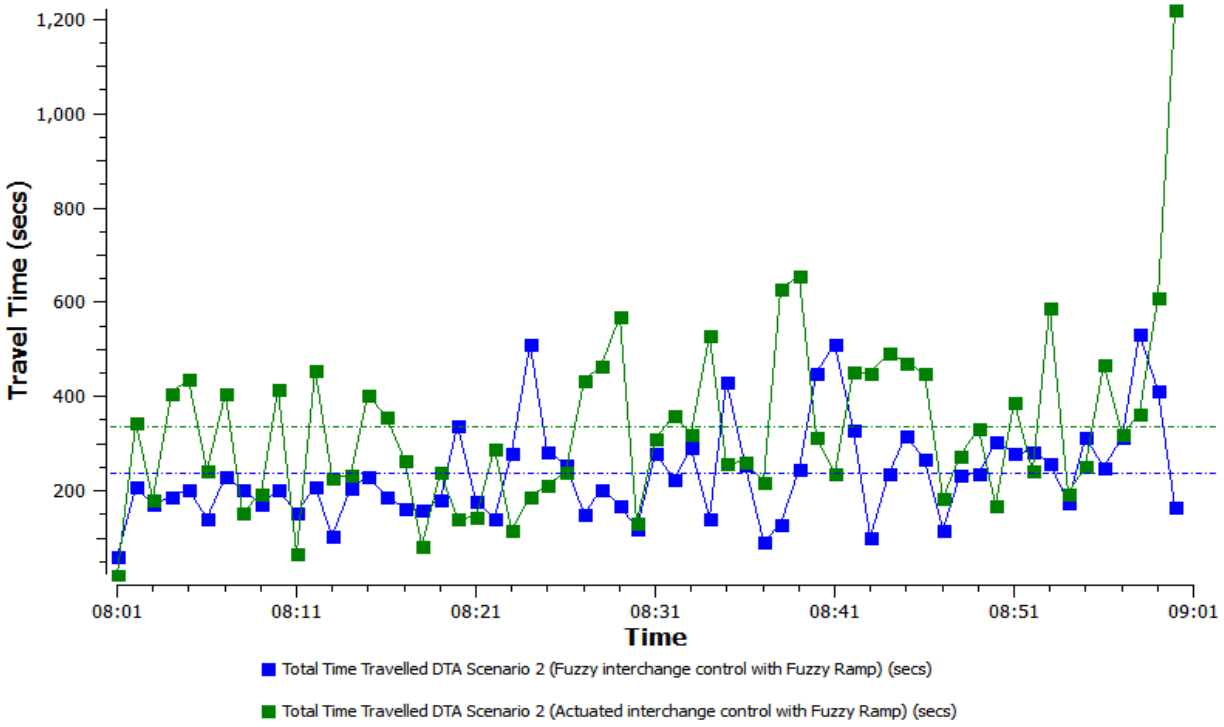
SCENARIO 2
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



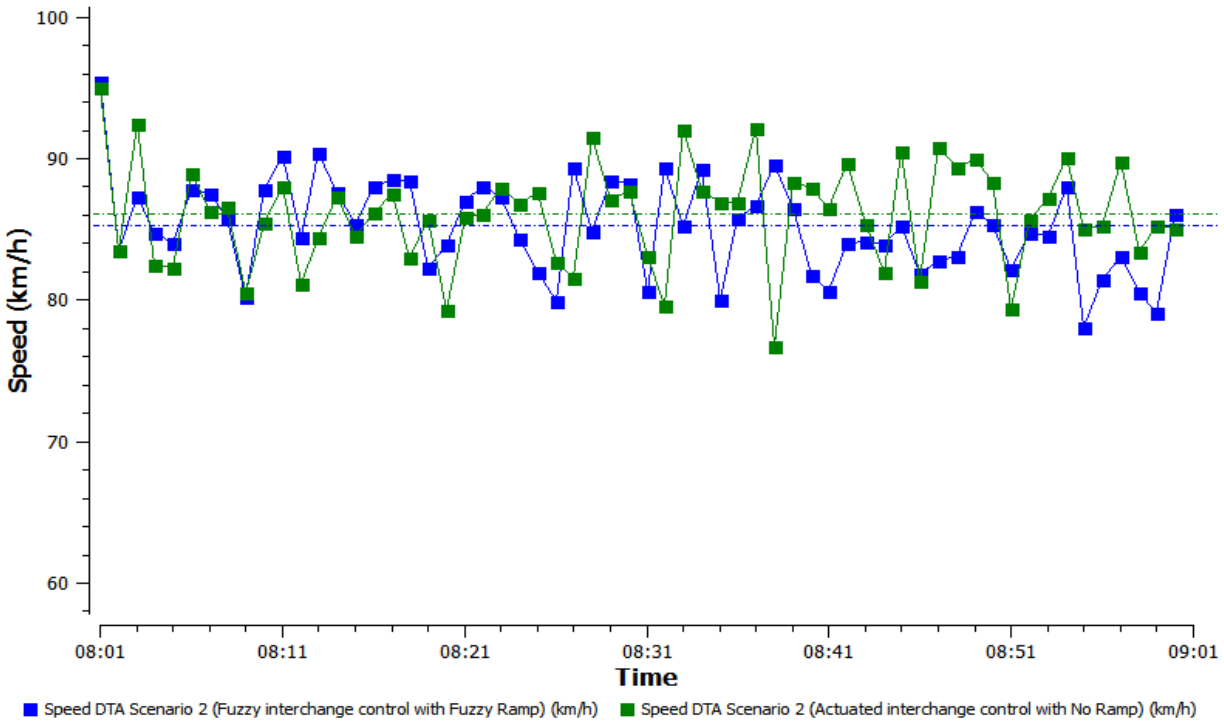


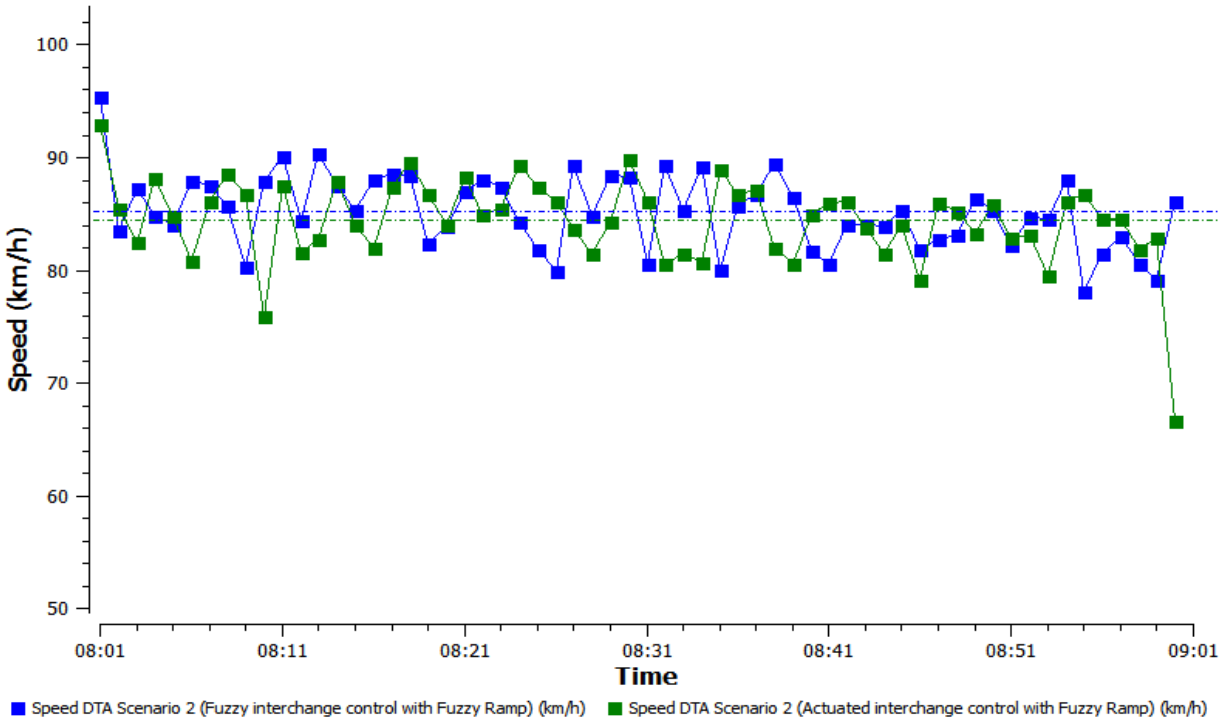
SCENARIO 2
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



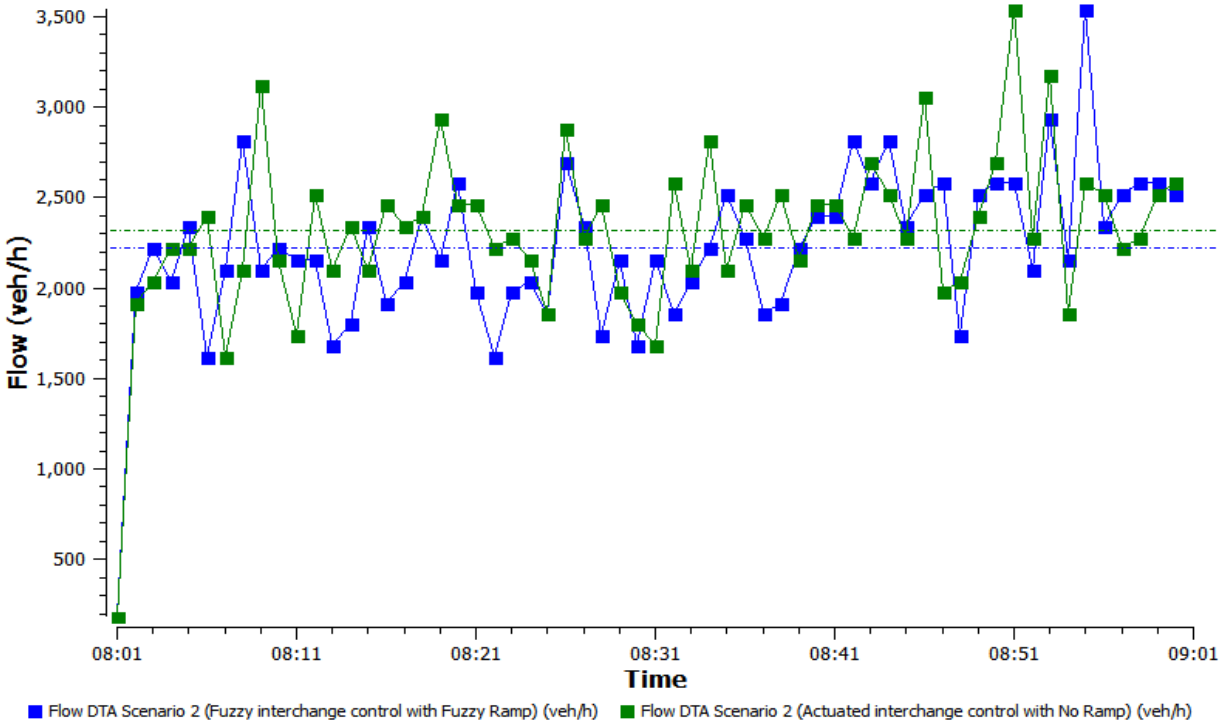


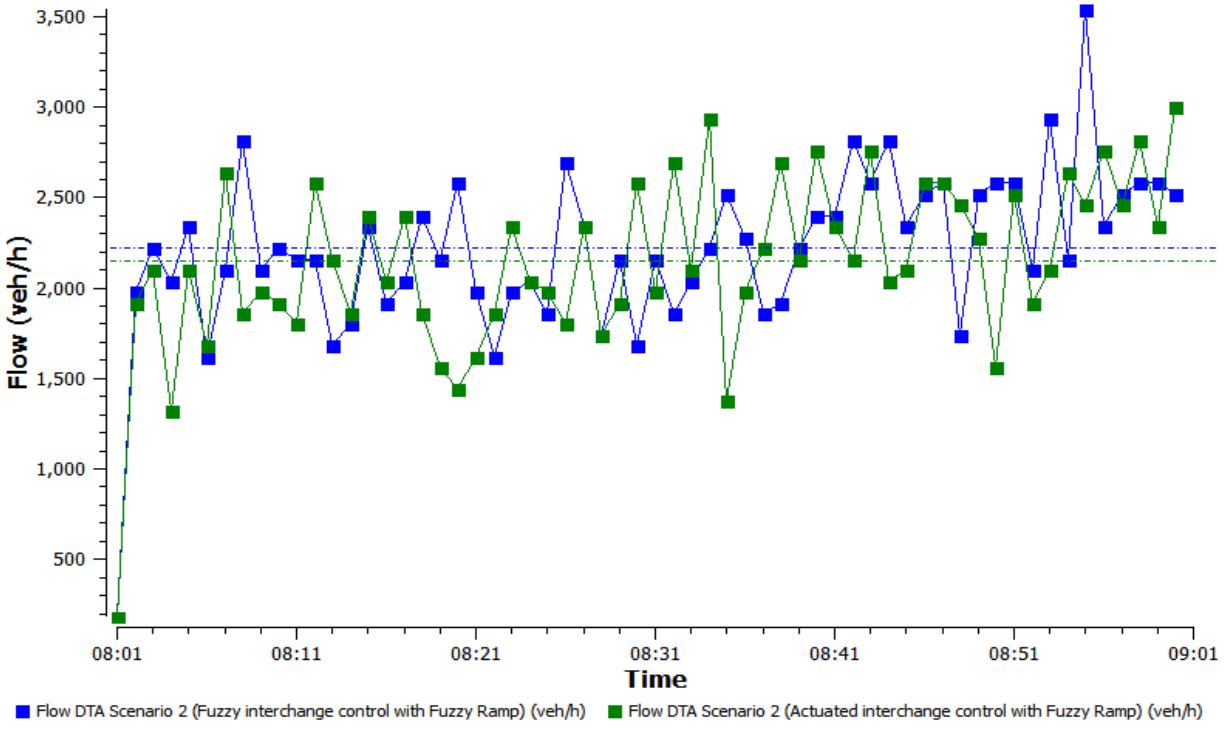
SCENARIO 2
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE SPEED



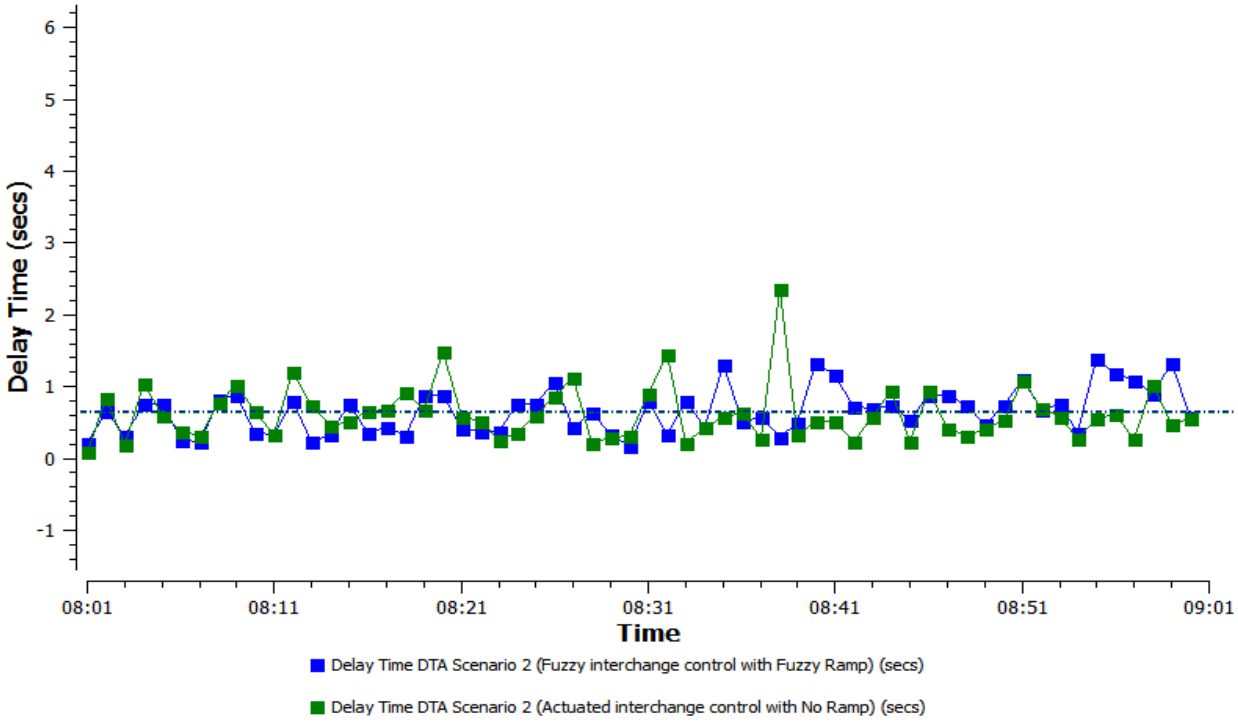


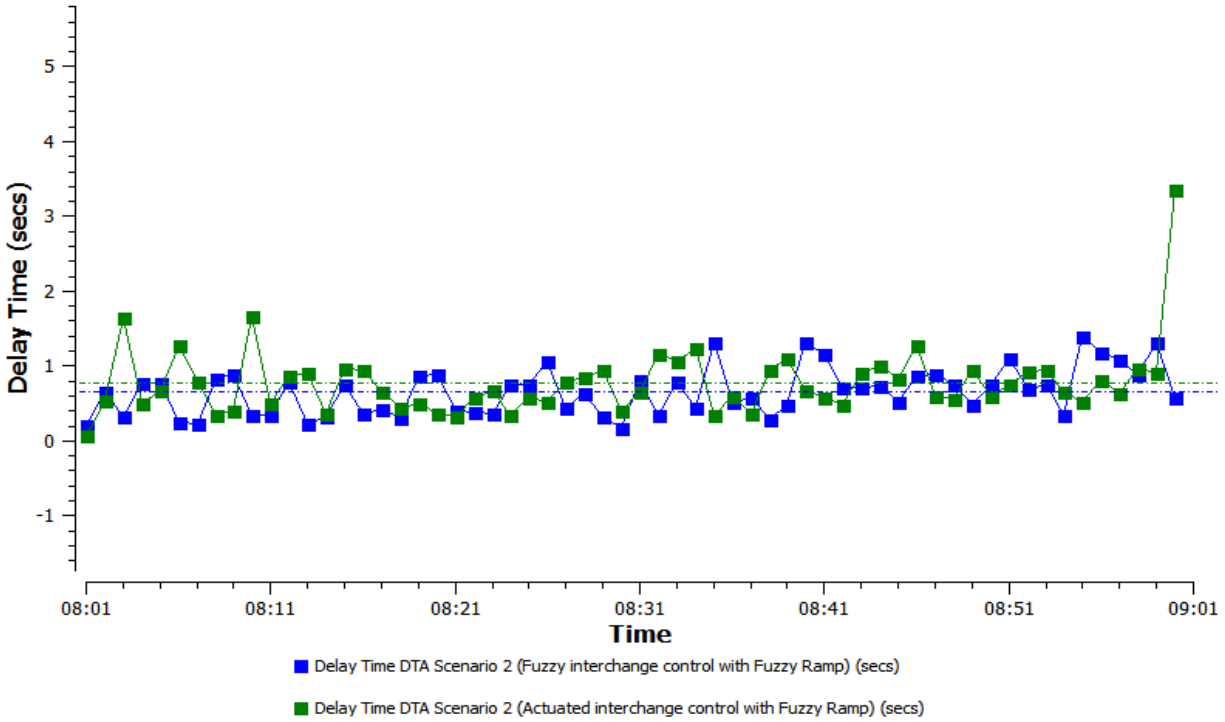
SCENARIO 2
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



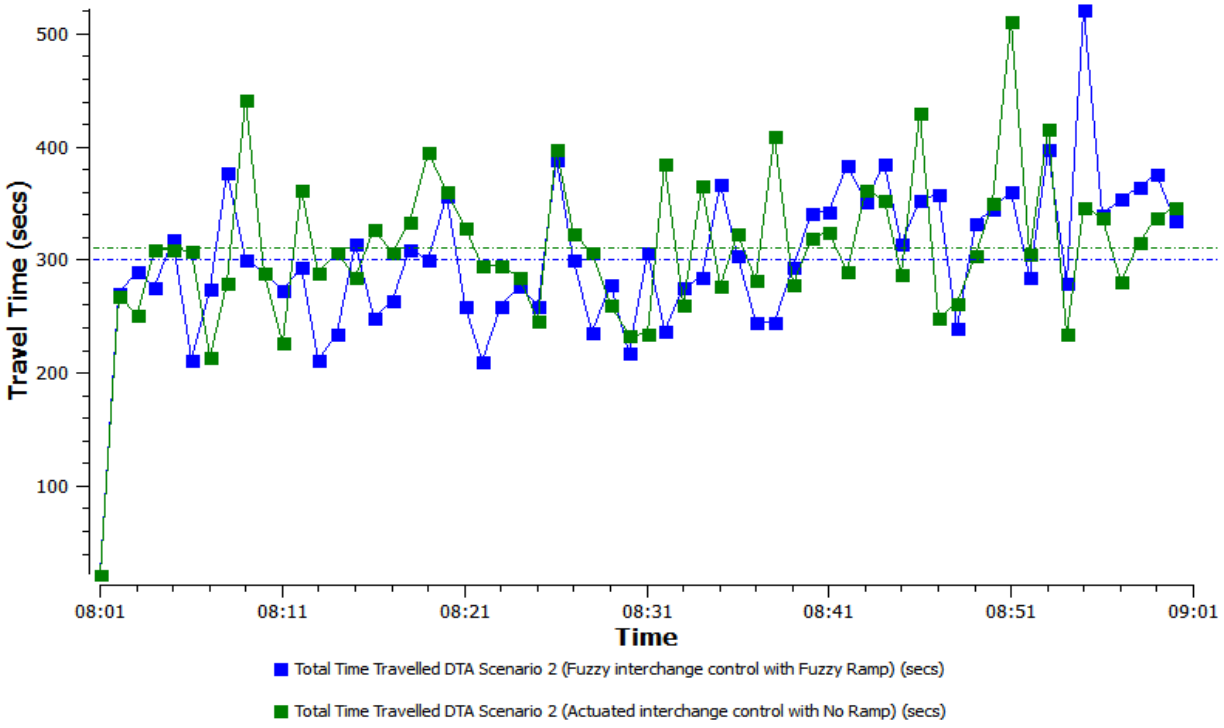


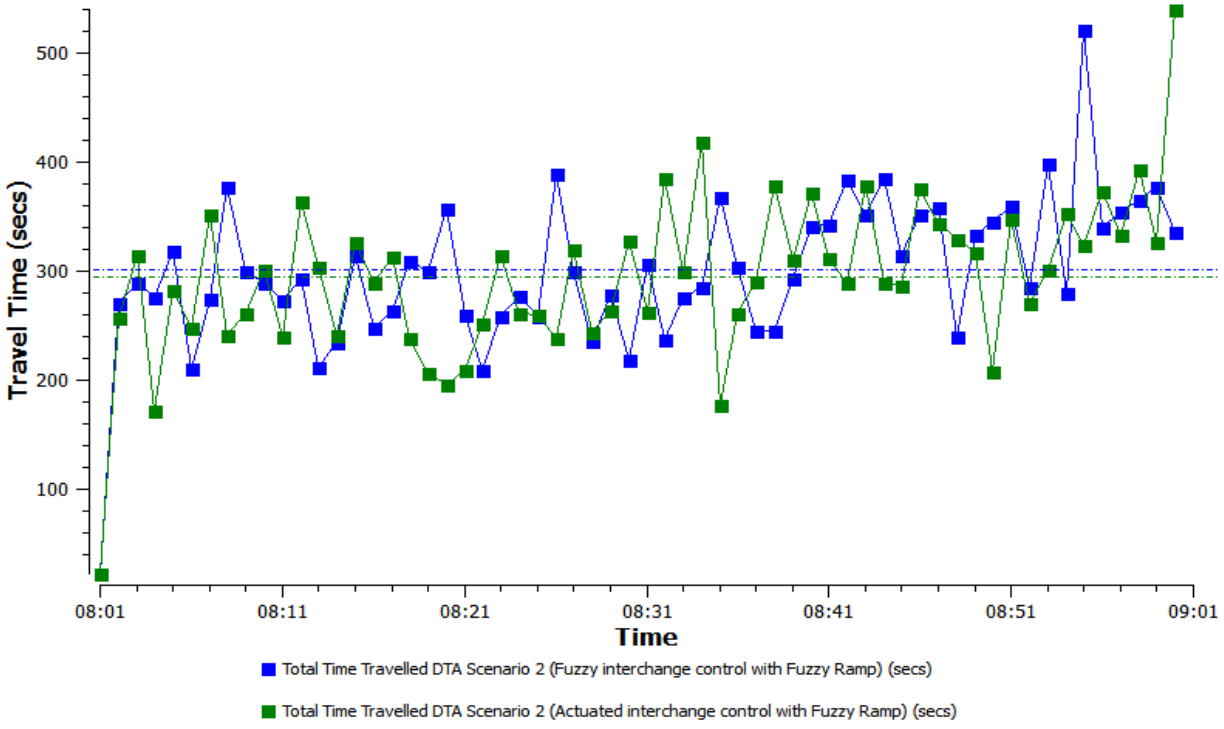
SCENARIO 2
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



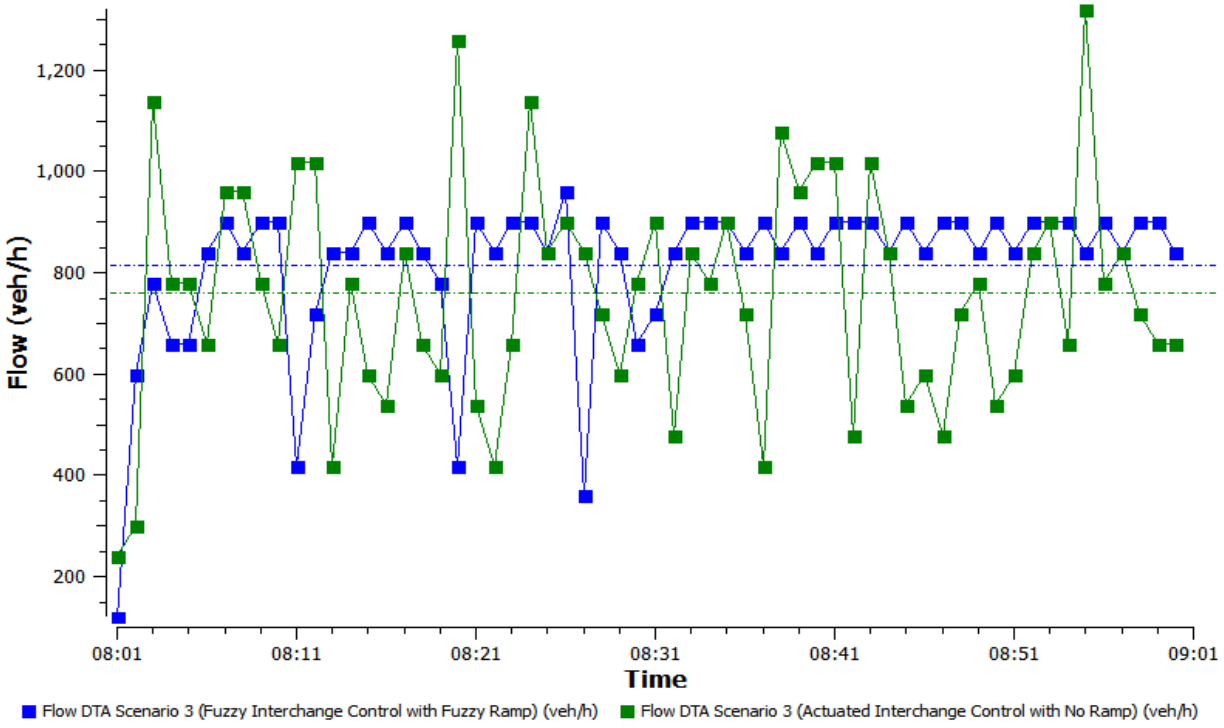


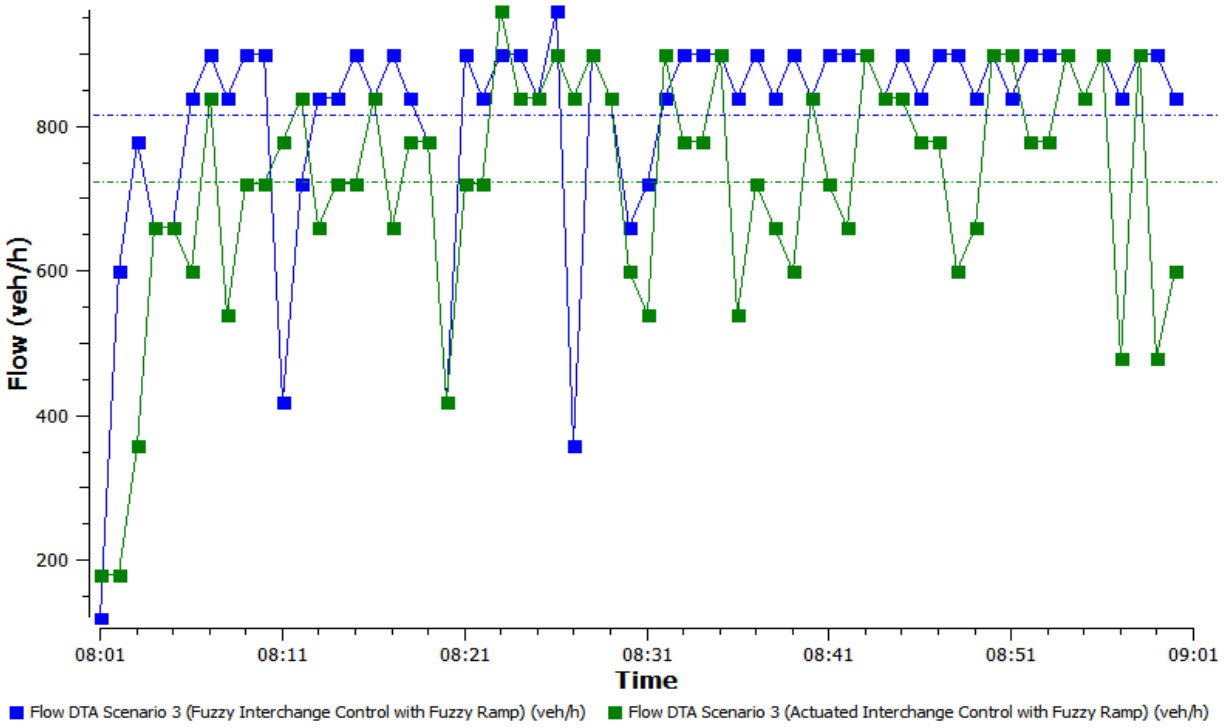
SCENARIO 2
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



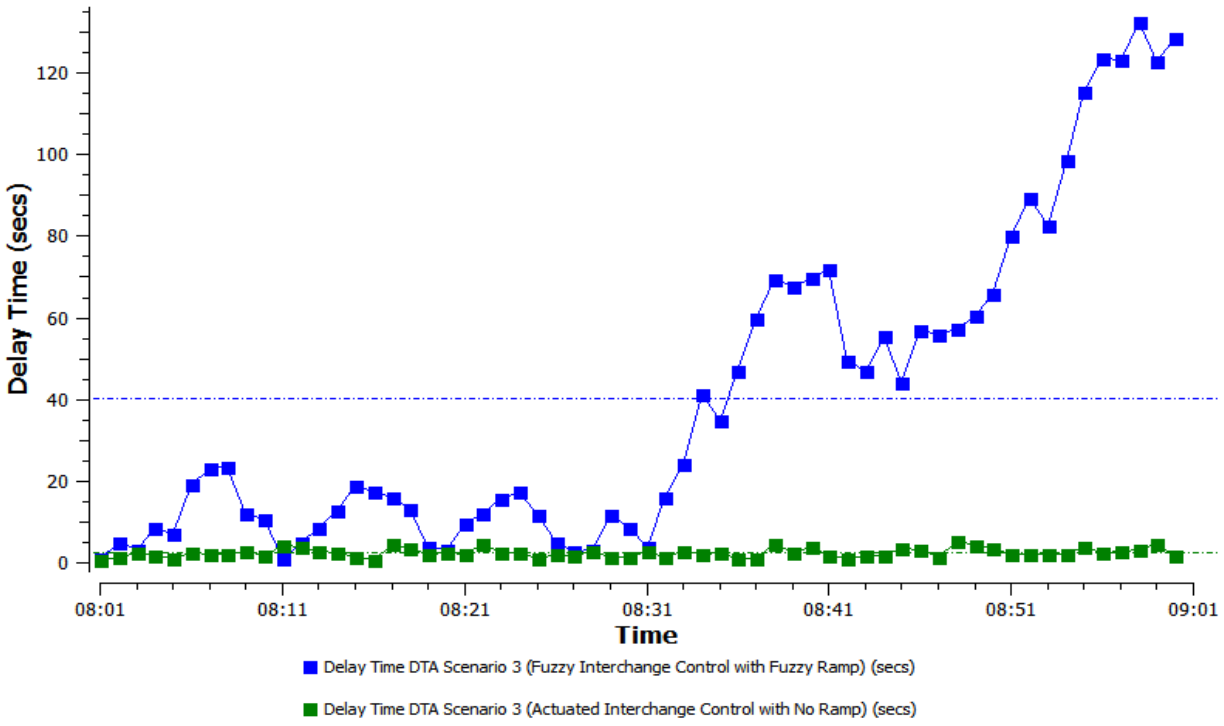


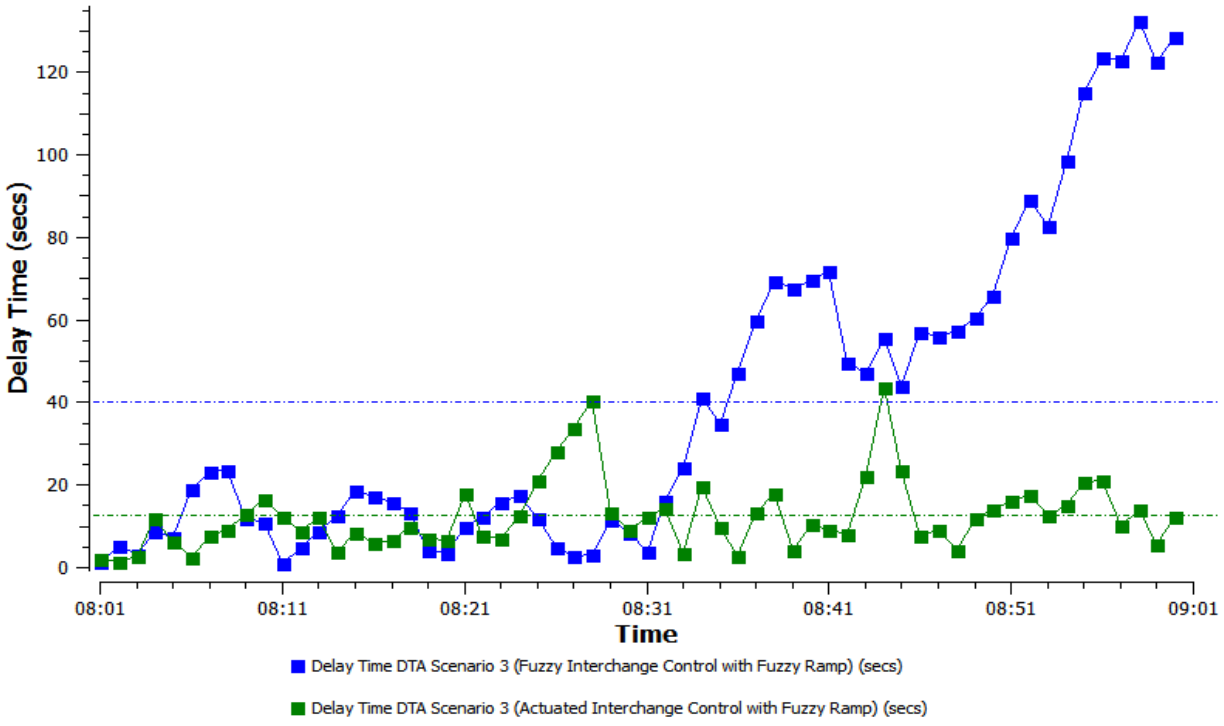
SCENARIO 3
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



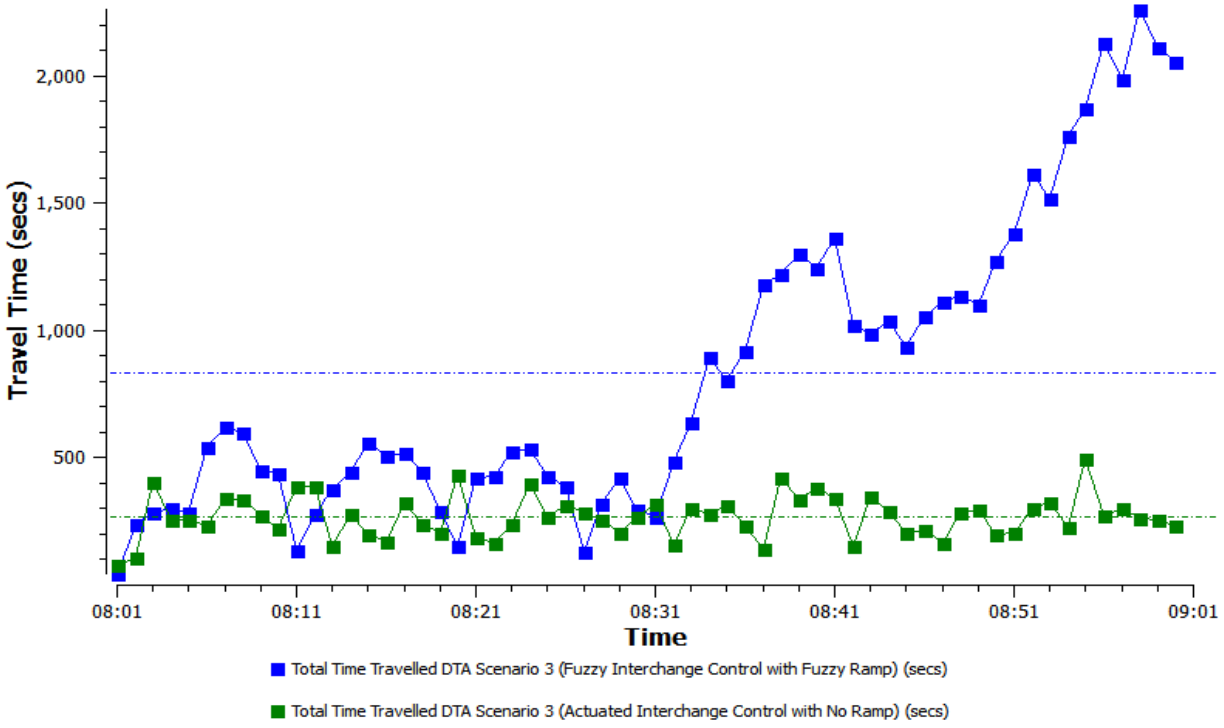


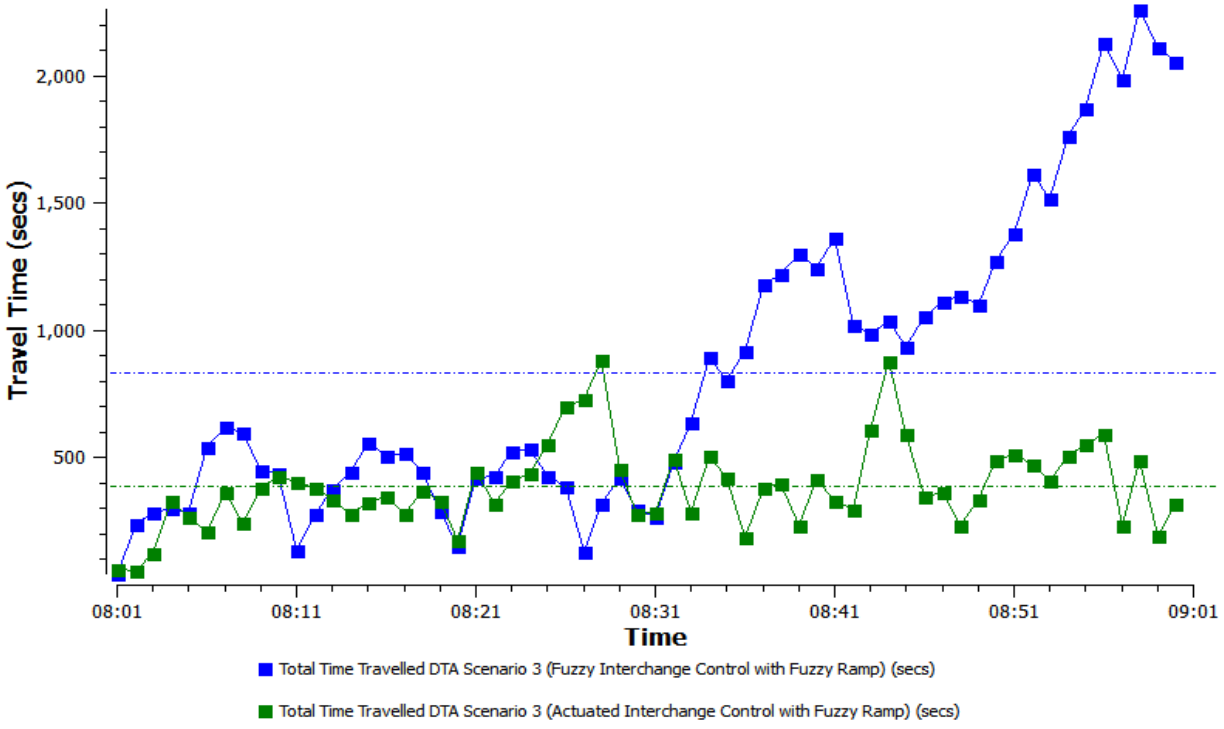
SCENARIO 3
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



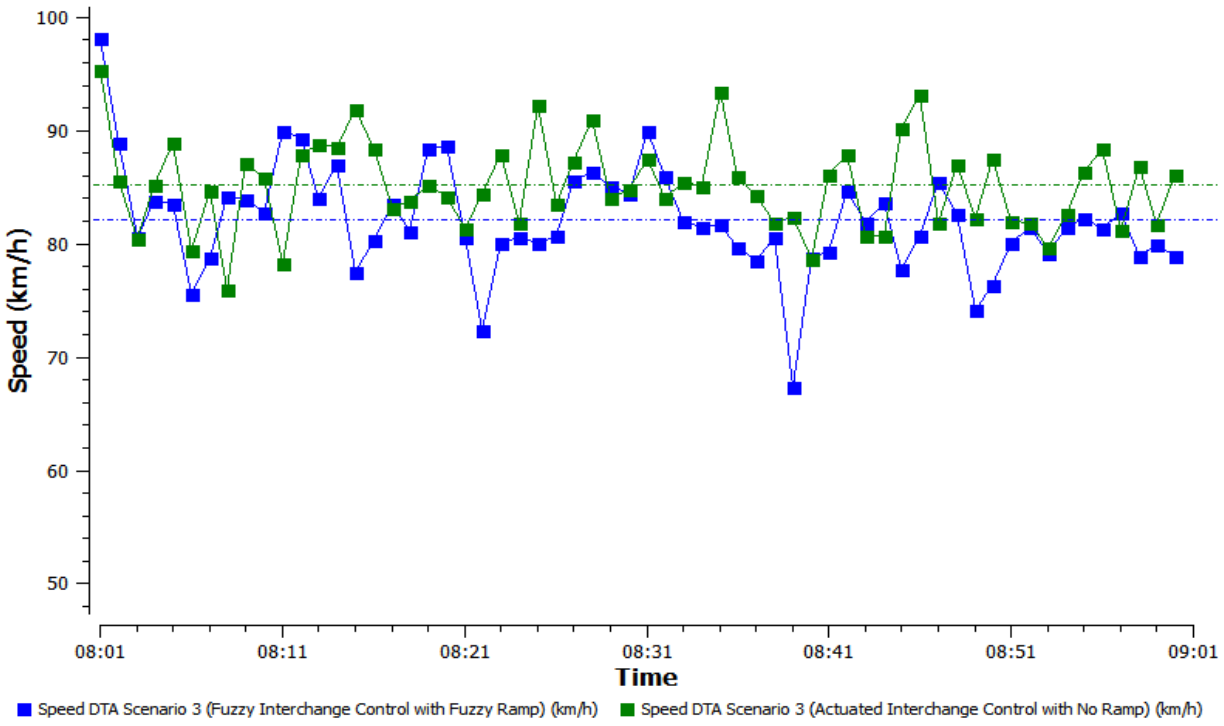


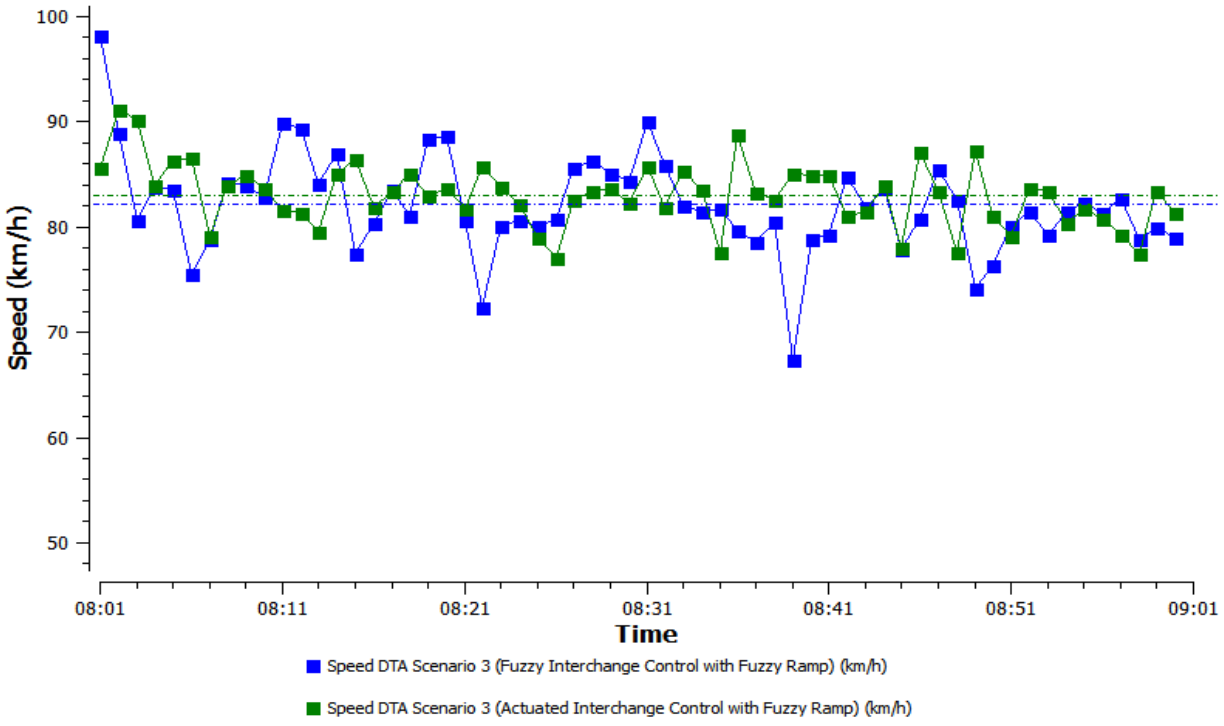
SCENARIO 3
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



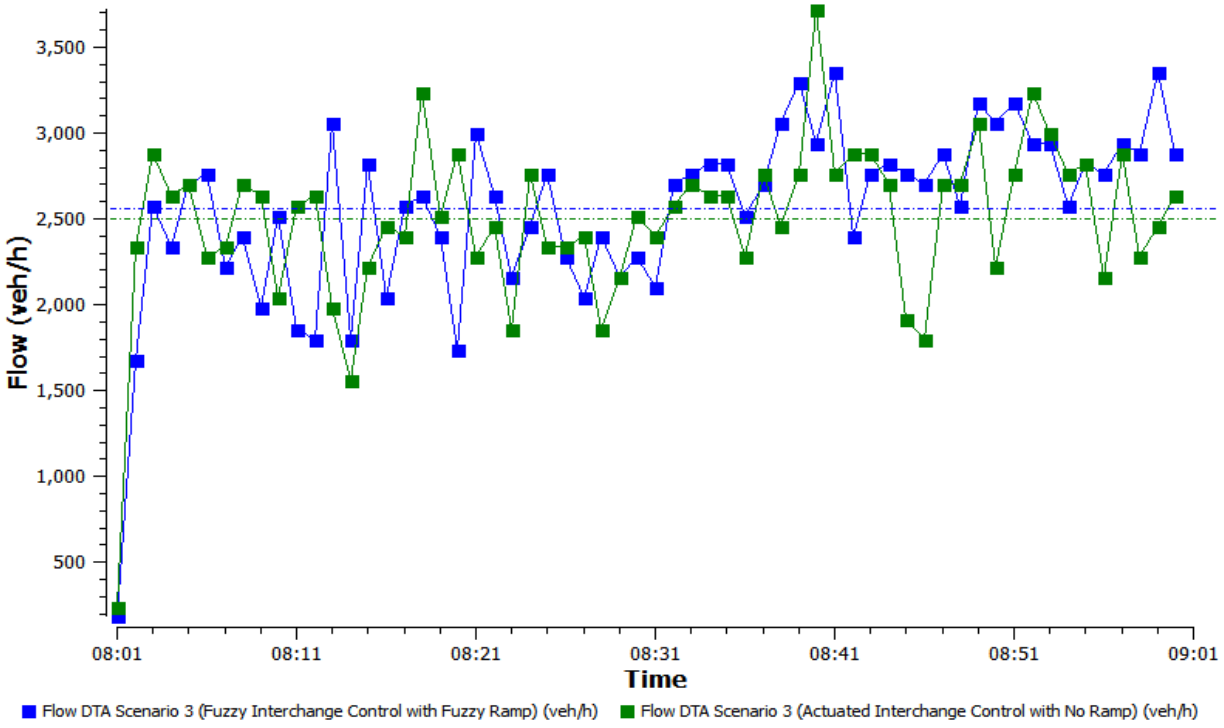


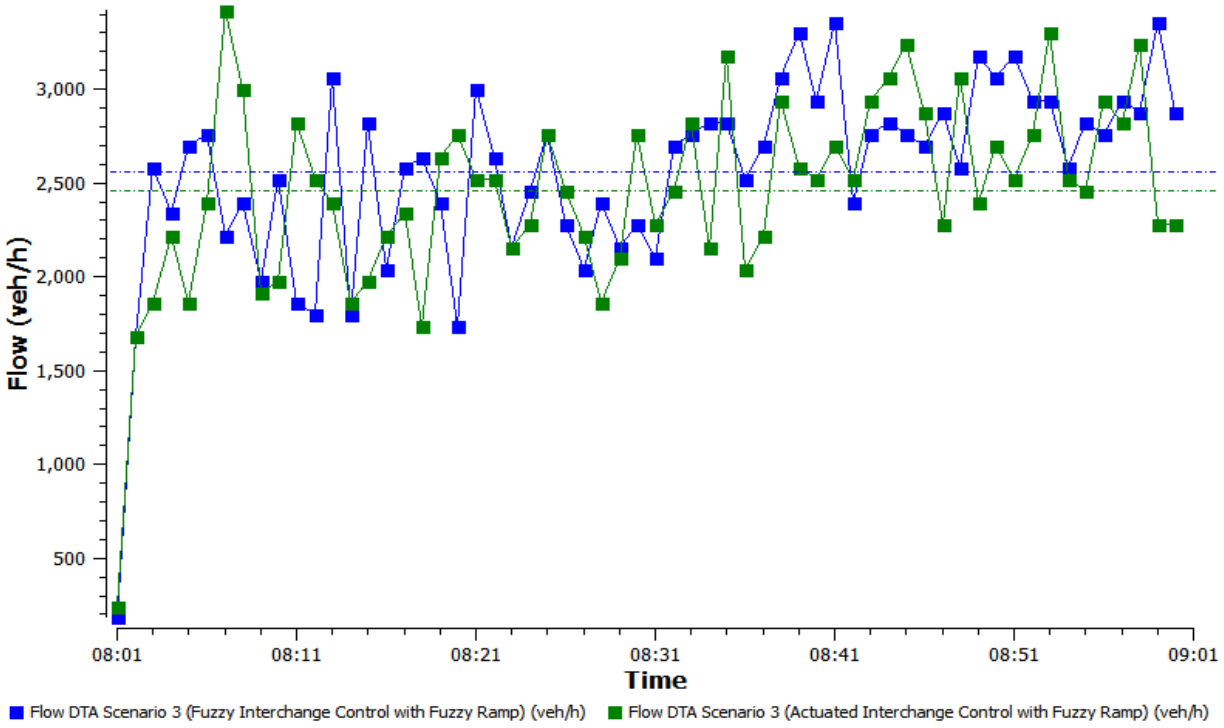
SCENARIO 3
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE SPEED



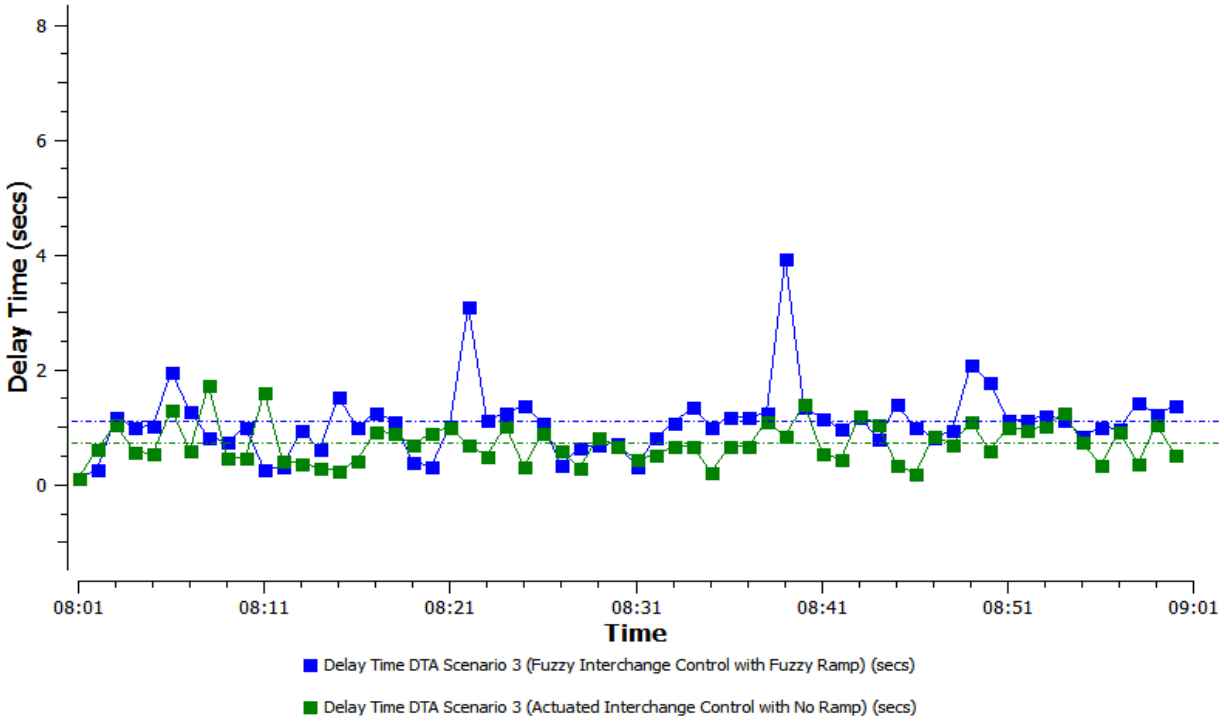


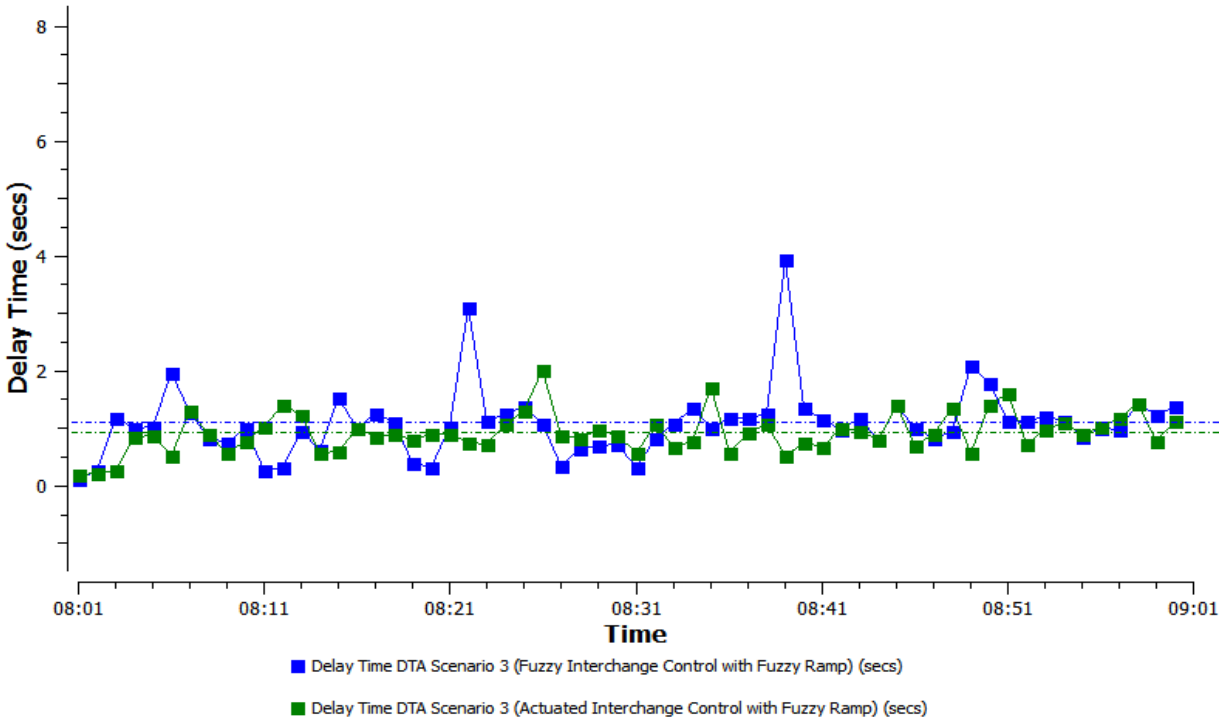
SCENARIO 3
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



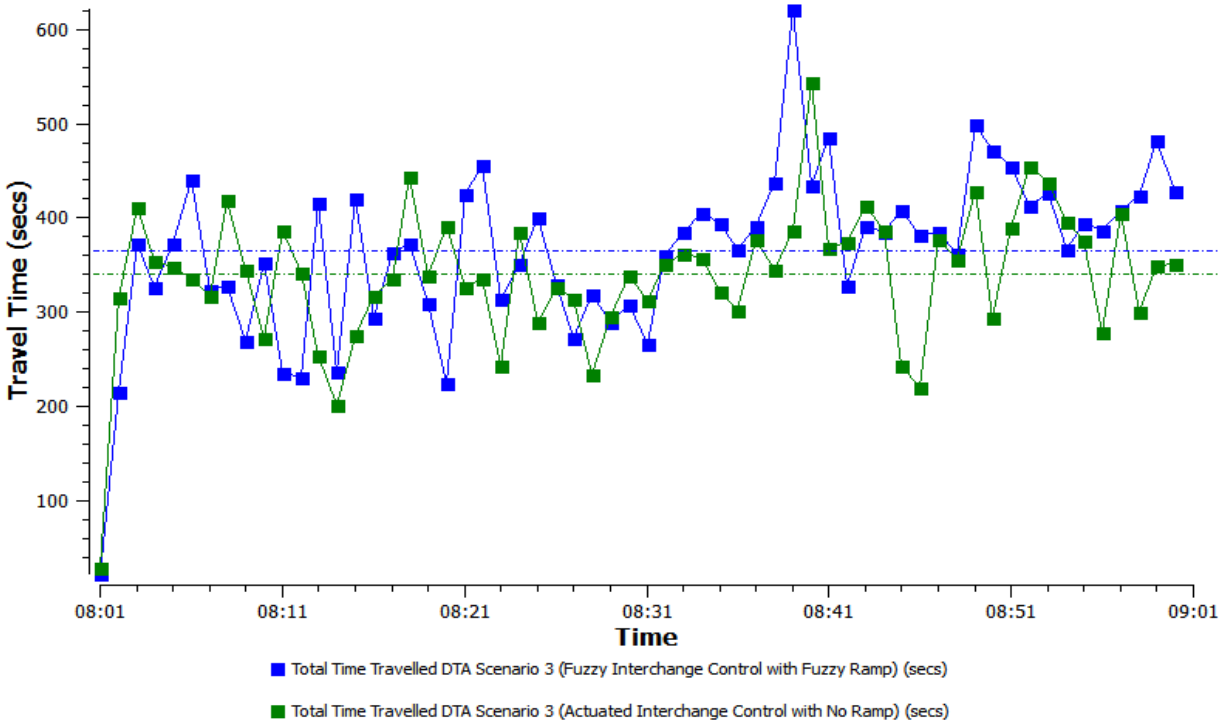


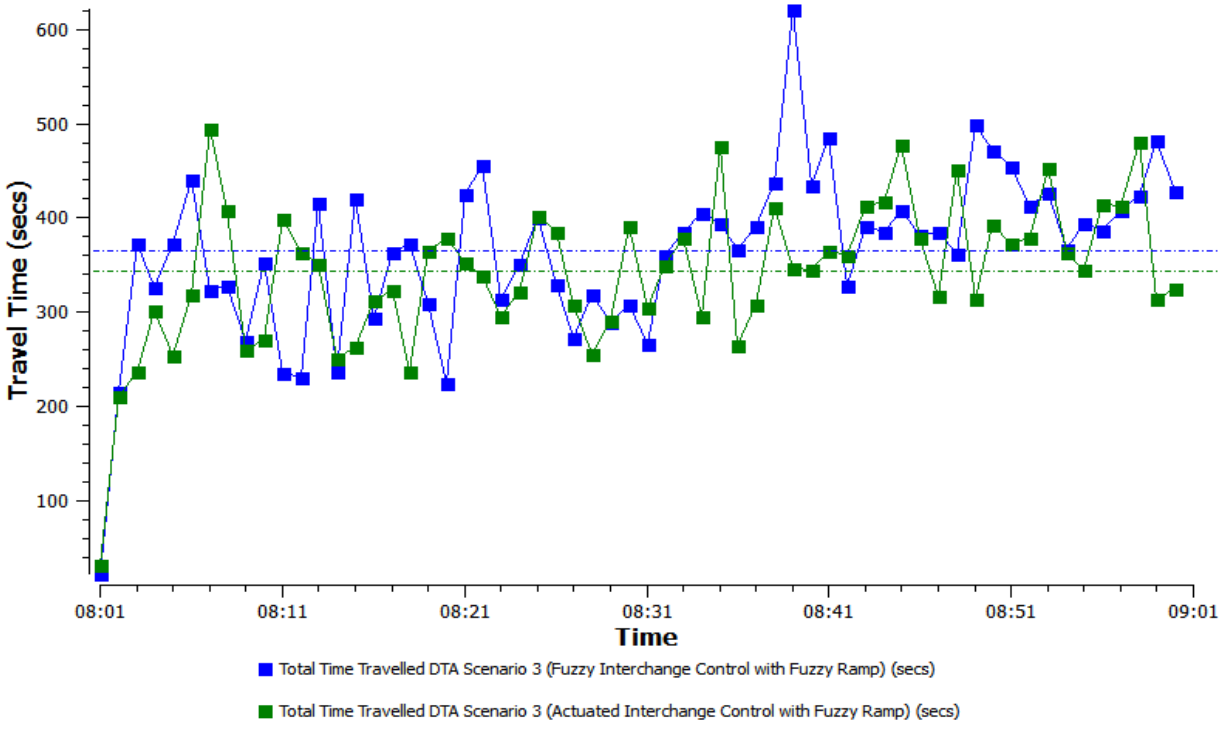
SCENARIO 3
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



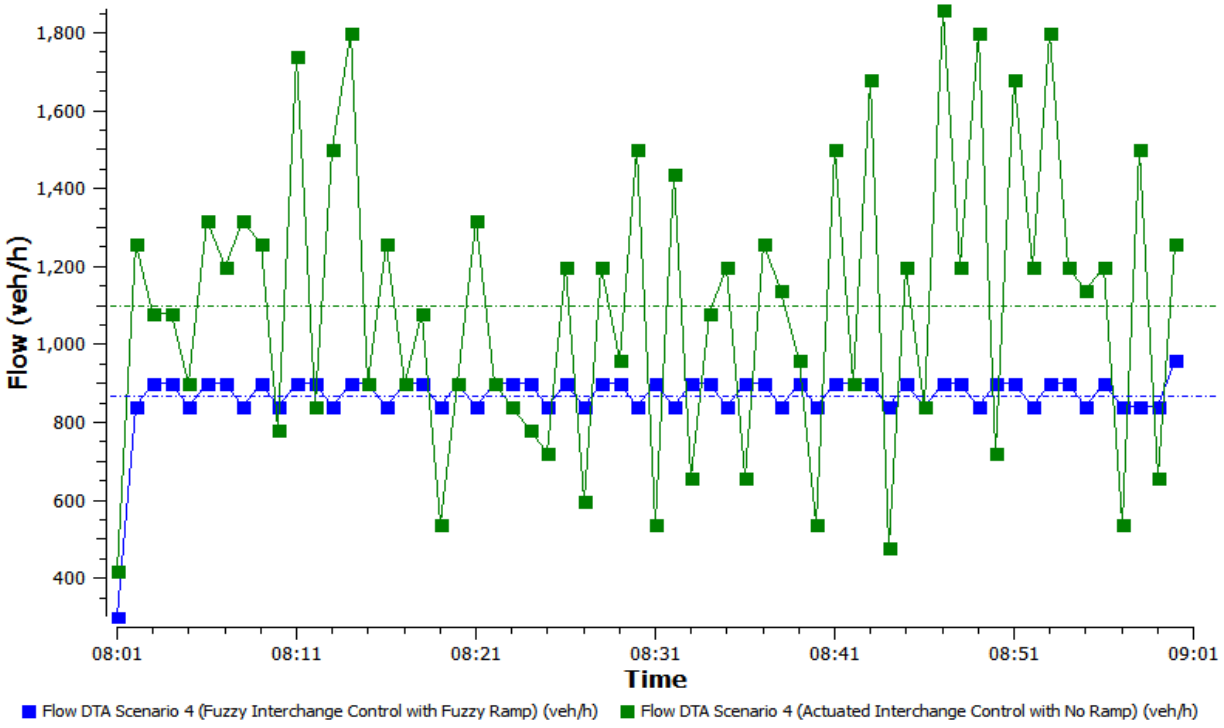


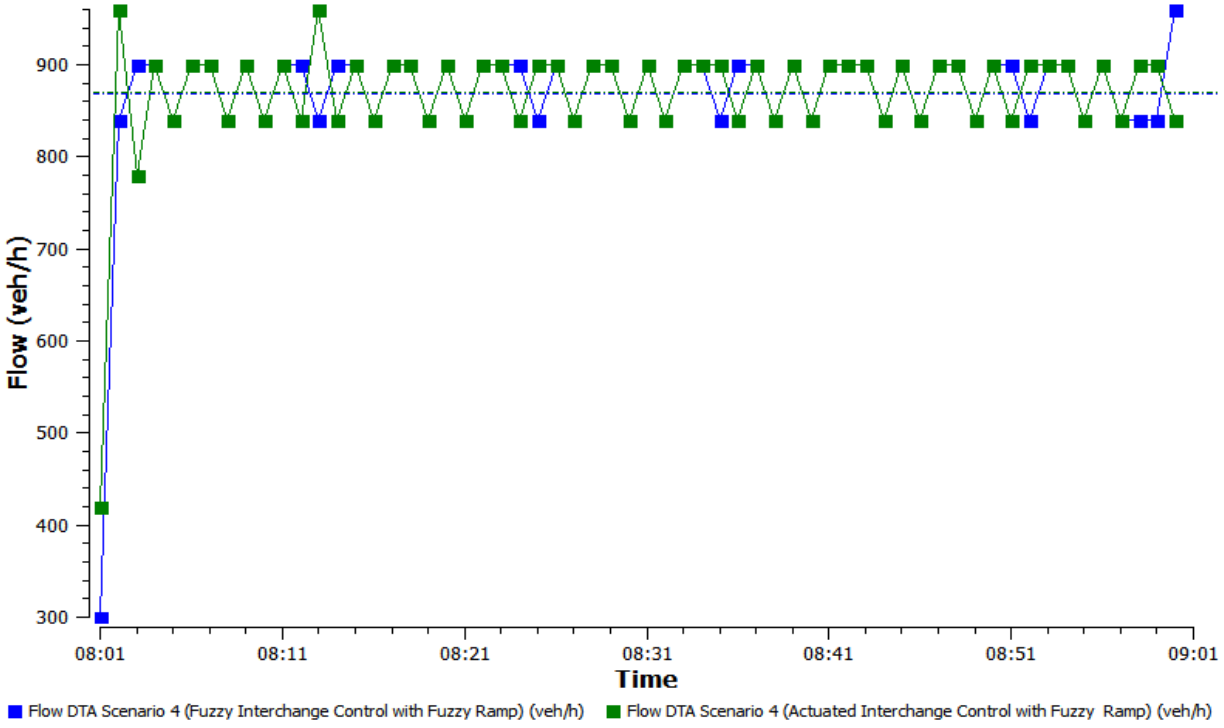
SCENARIO 3
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



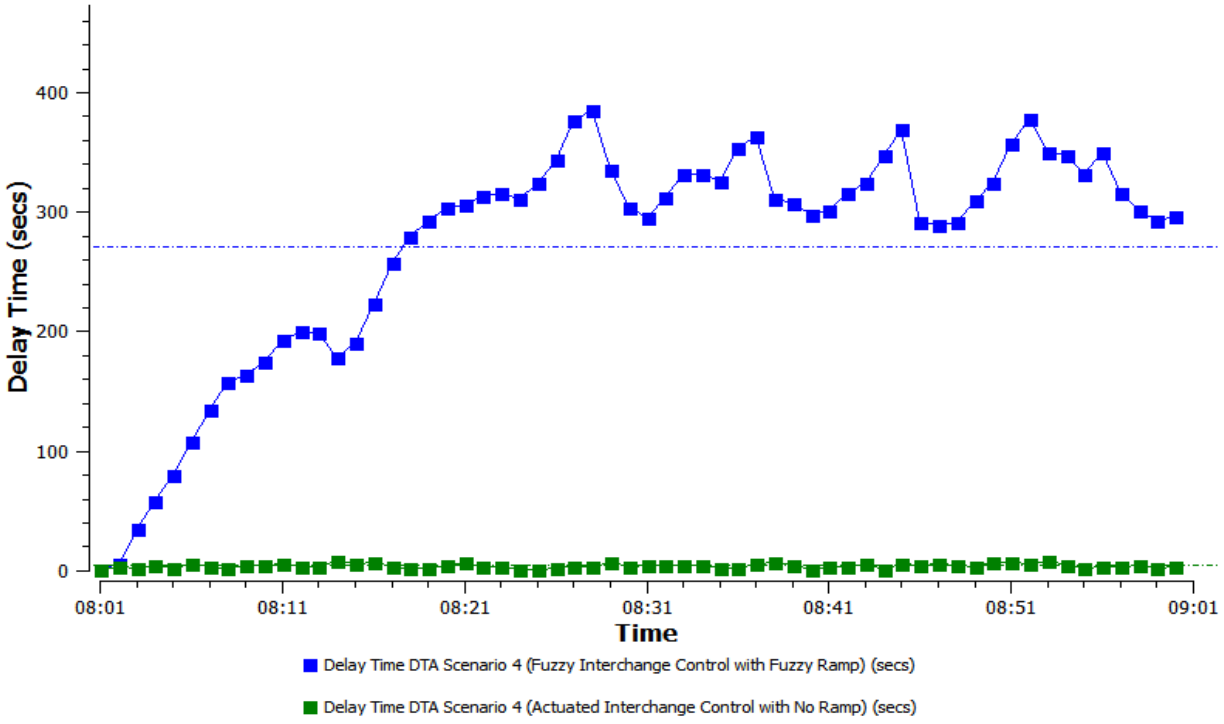


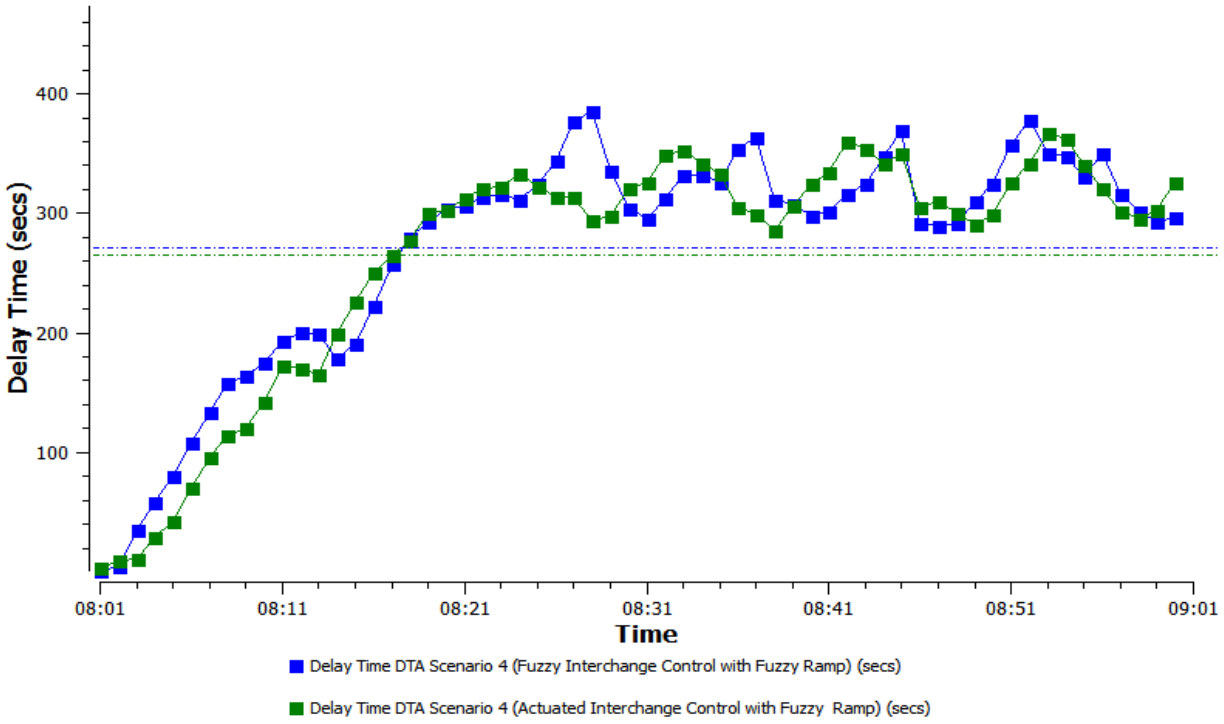
SCENARIO 4
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



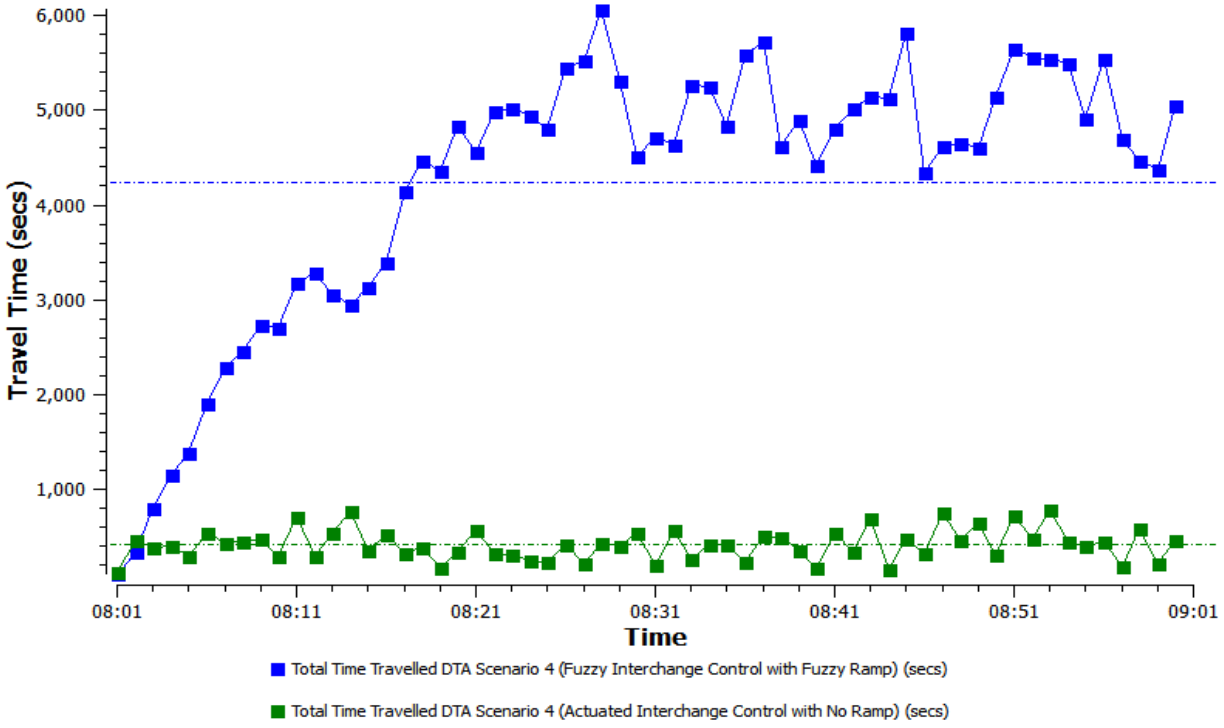


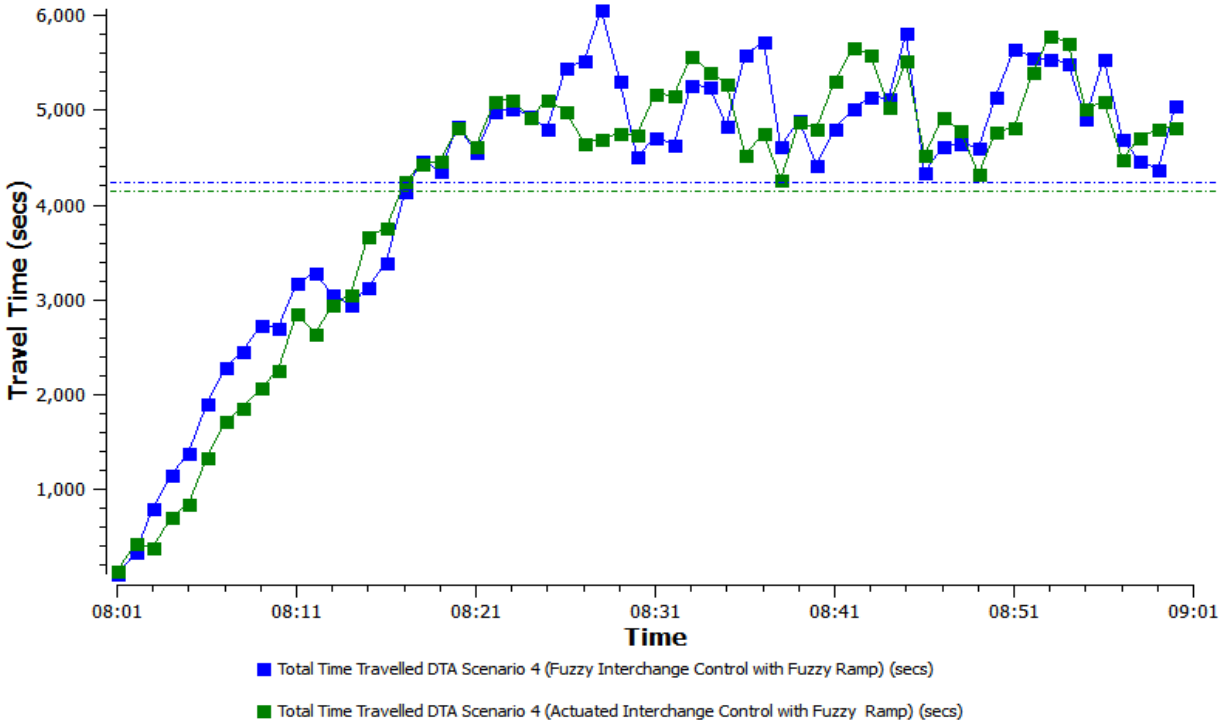
SCENARIO 4
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



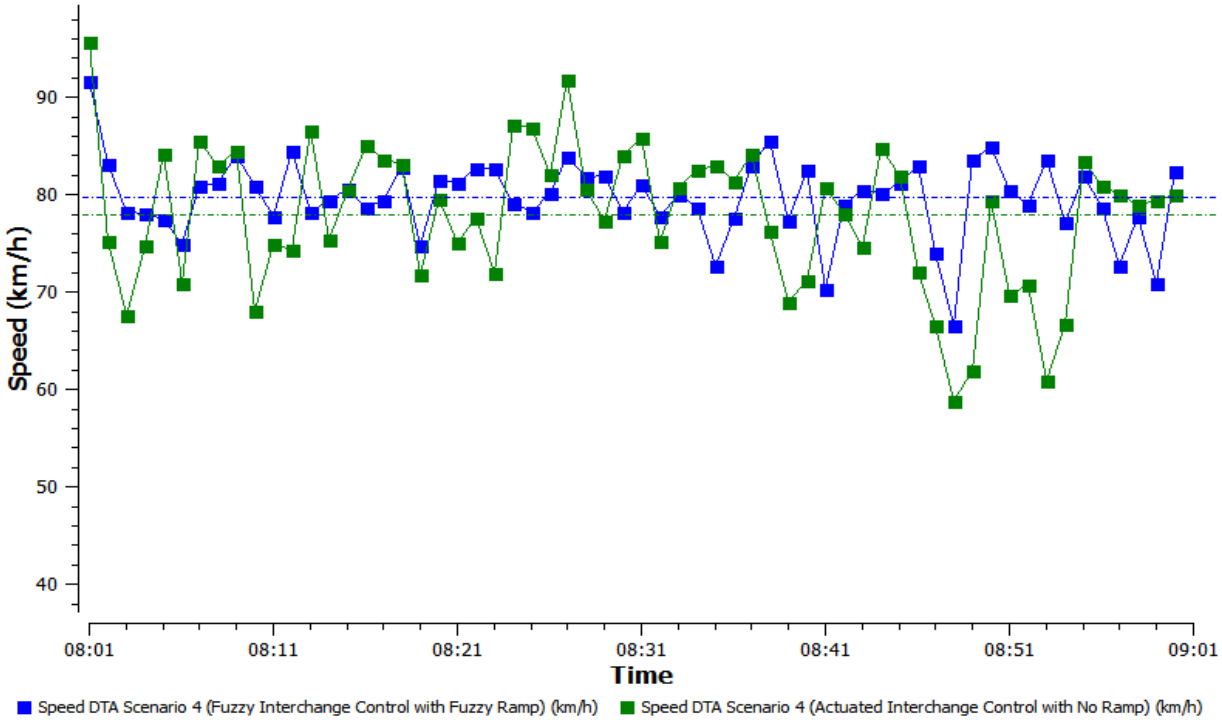


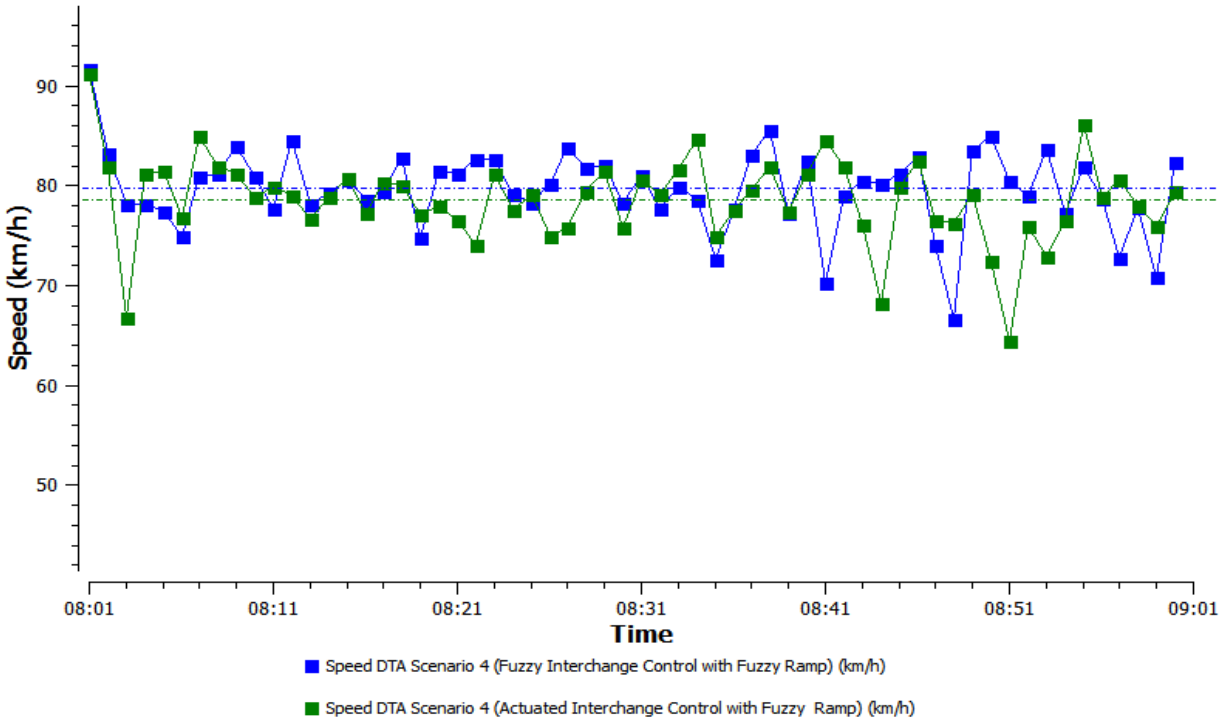
SCENARIO 4
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



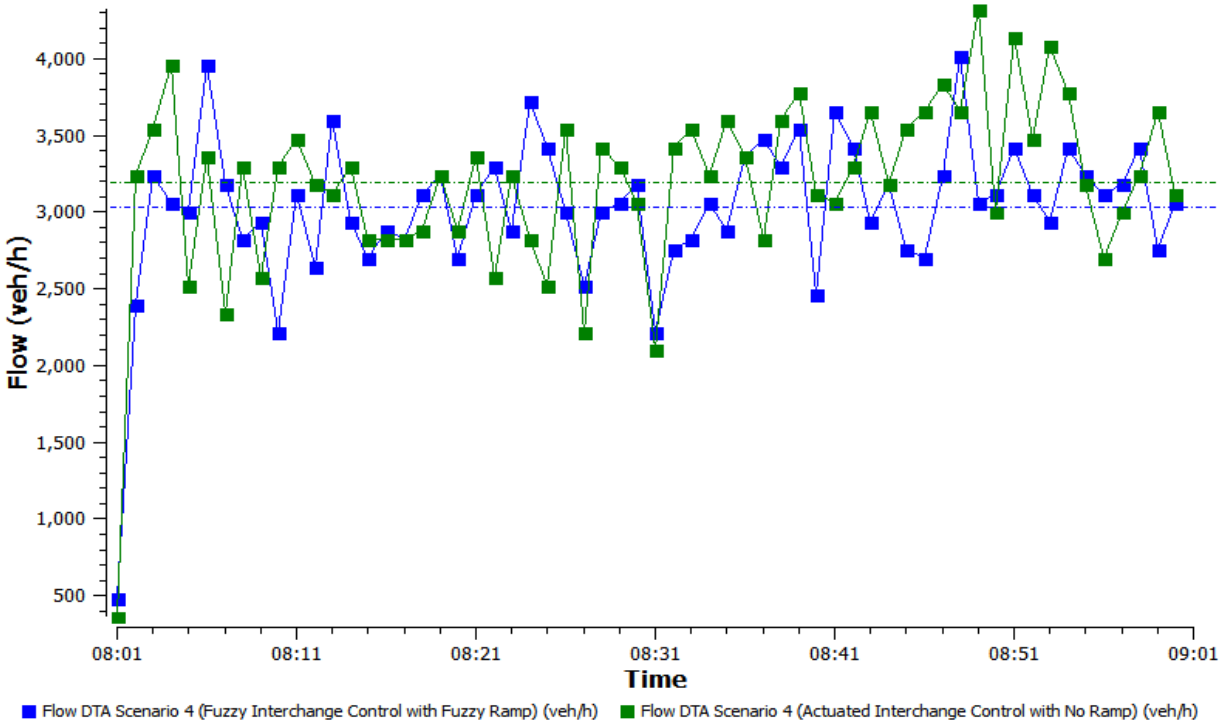


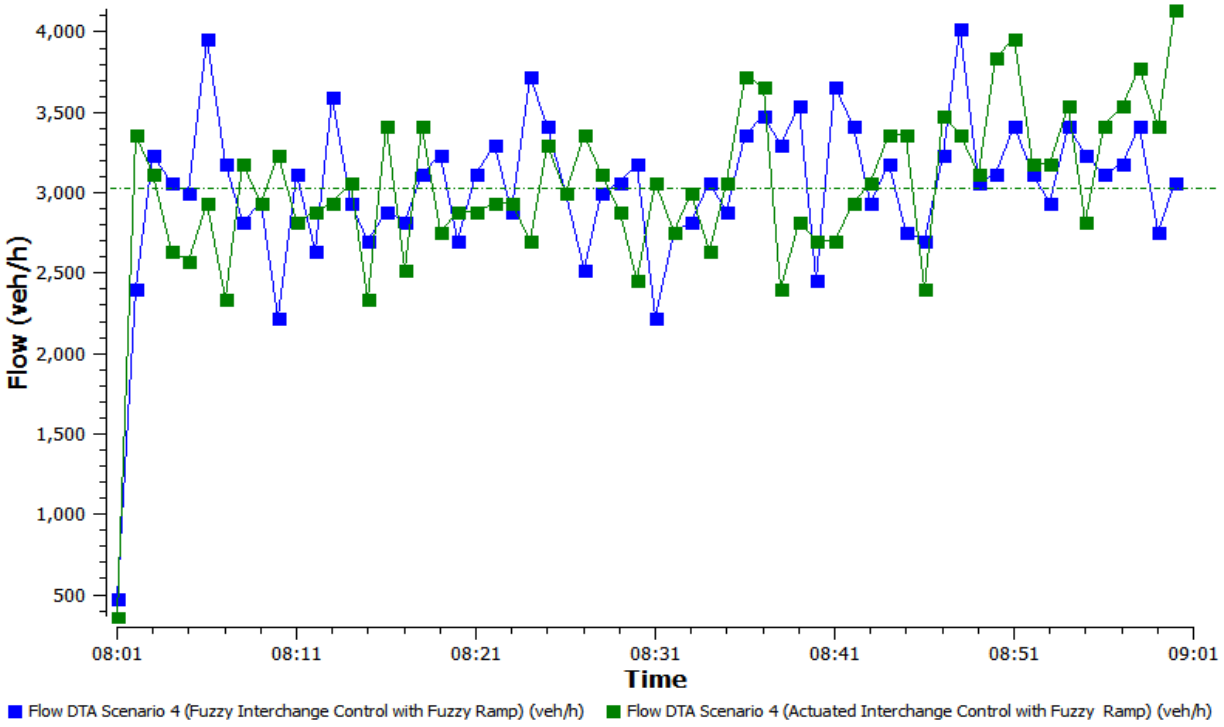
SCENARIO 4
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE SPEED



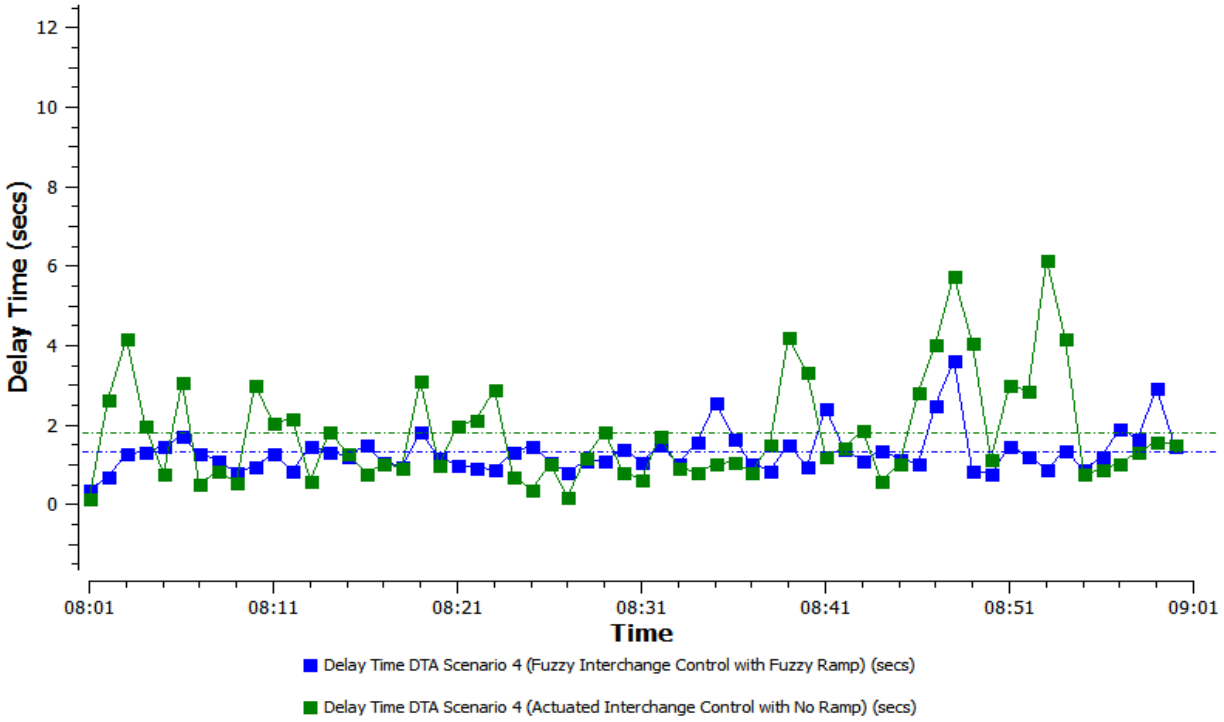


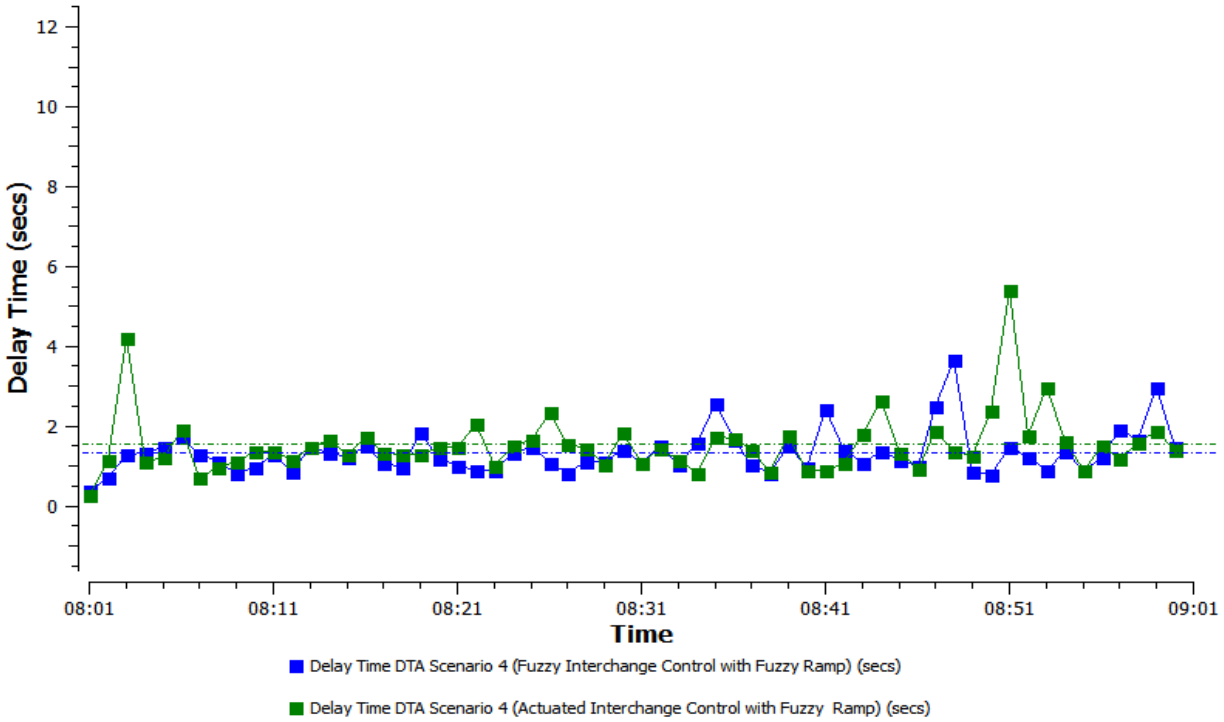
SCENARIO 4
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



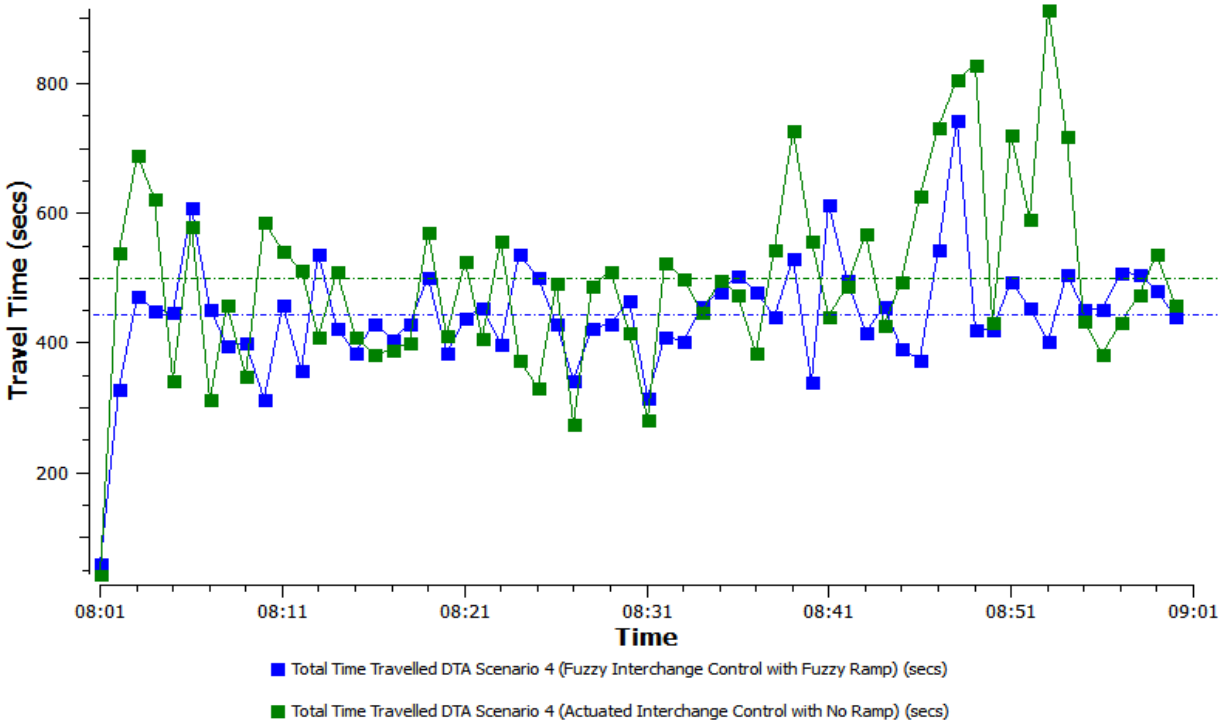


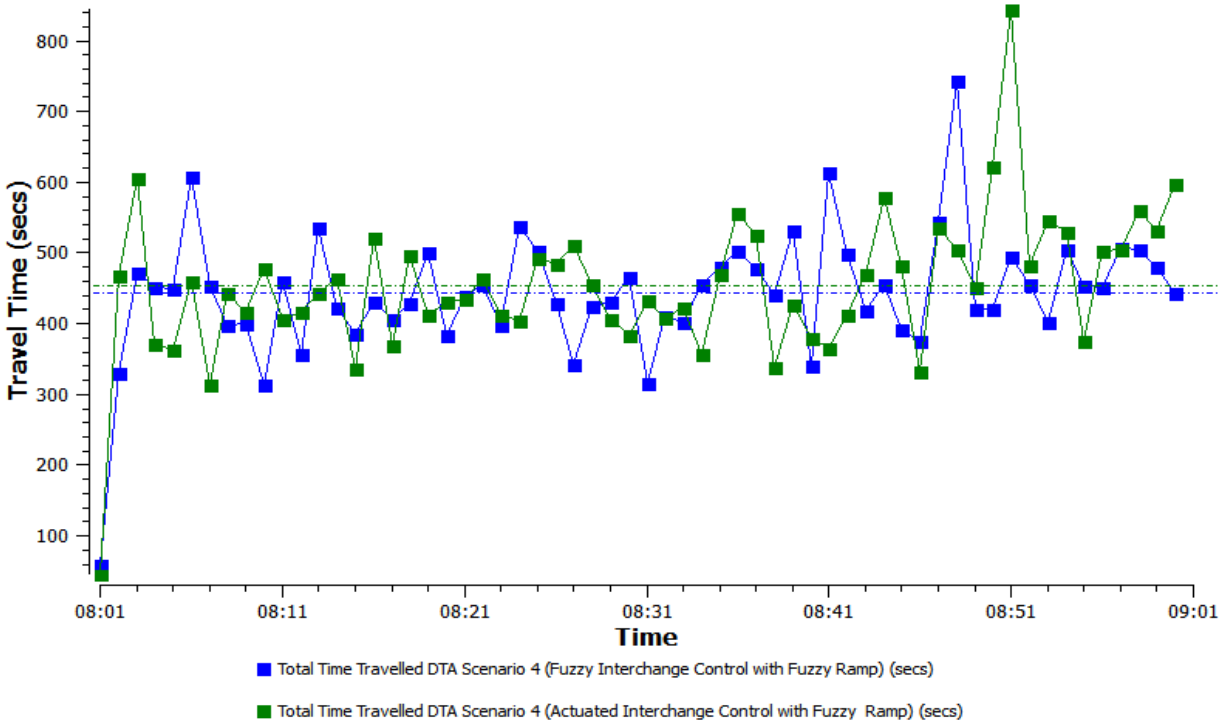
SCENARIO 4
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



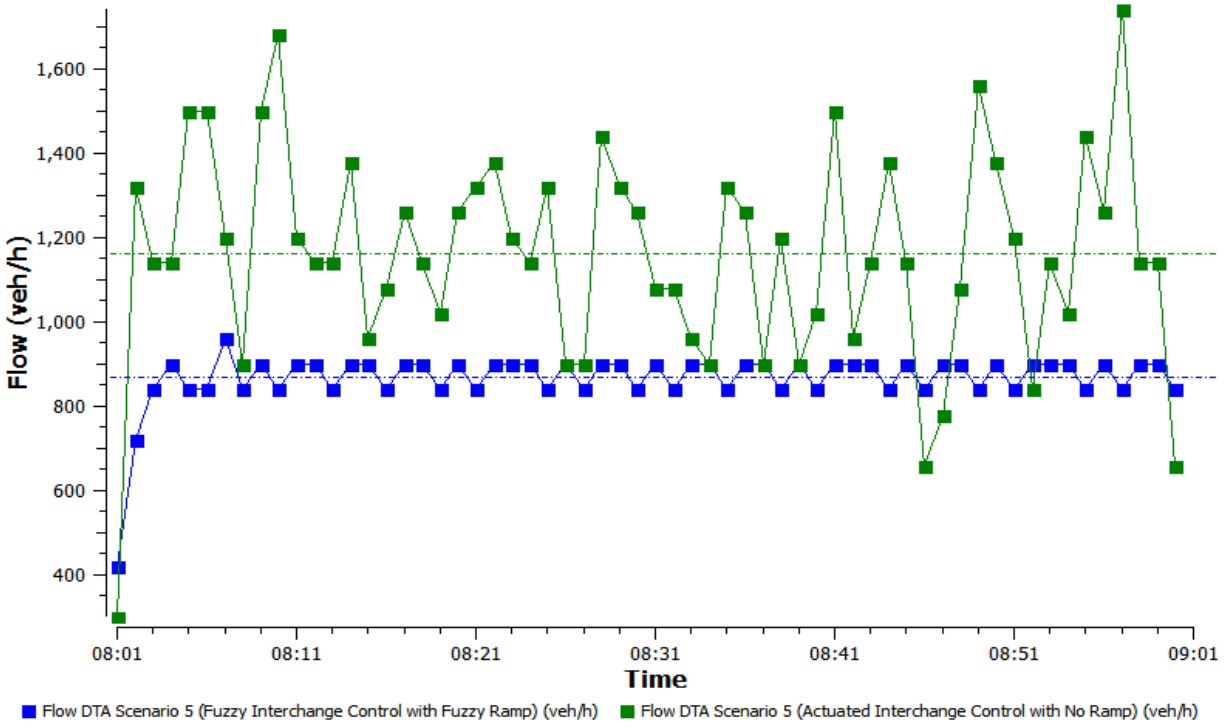


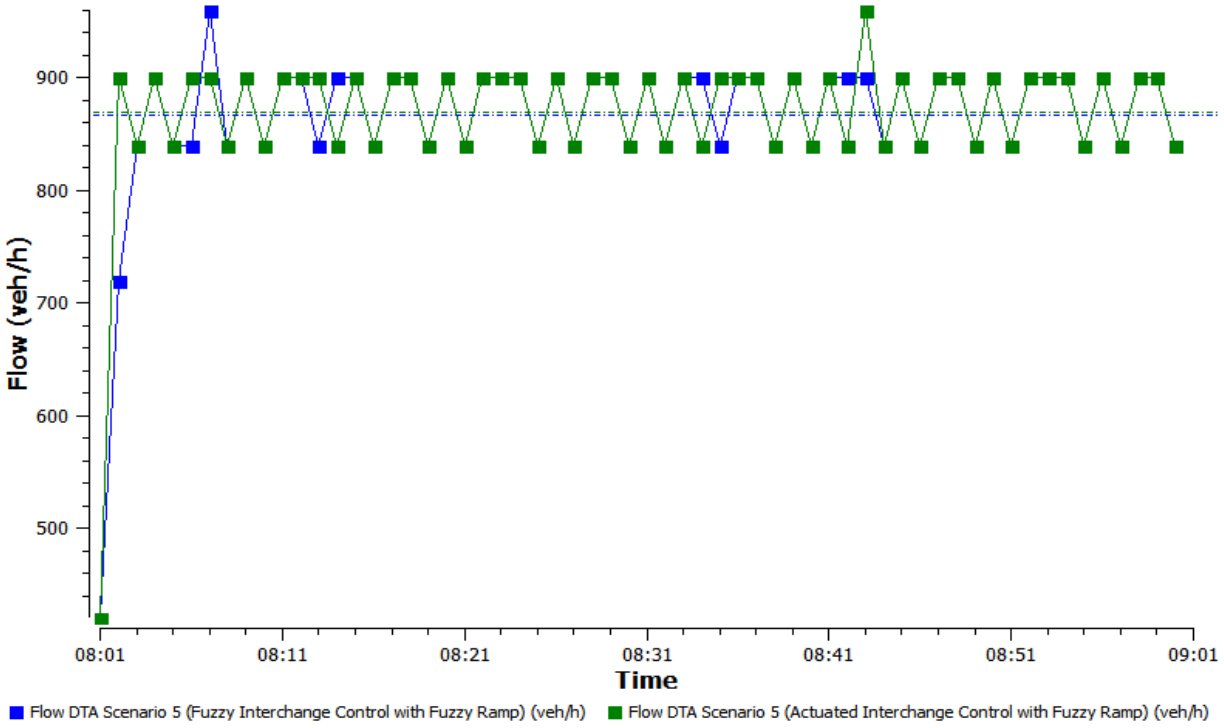
SCENARIO 4
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



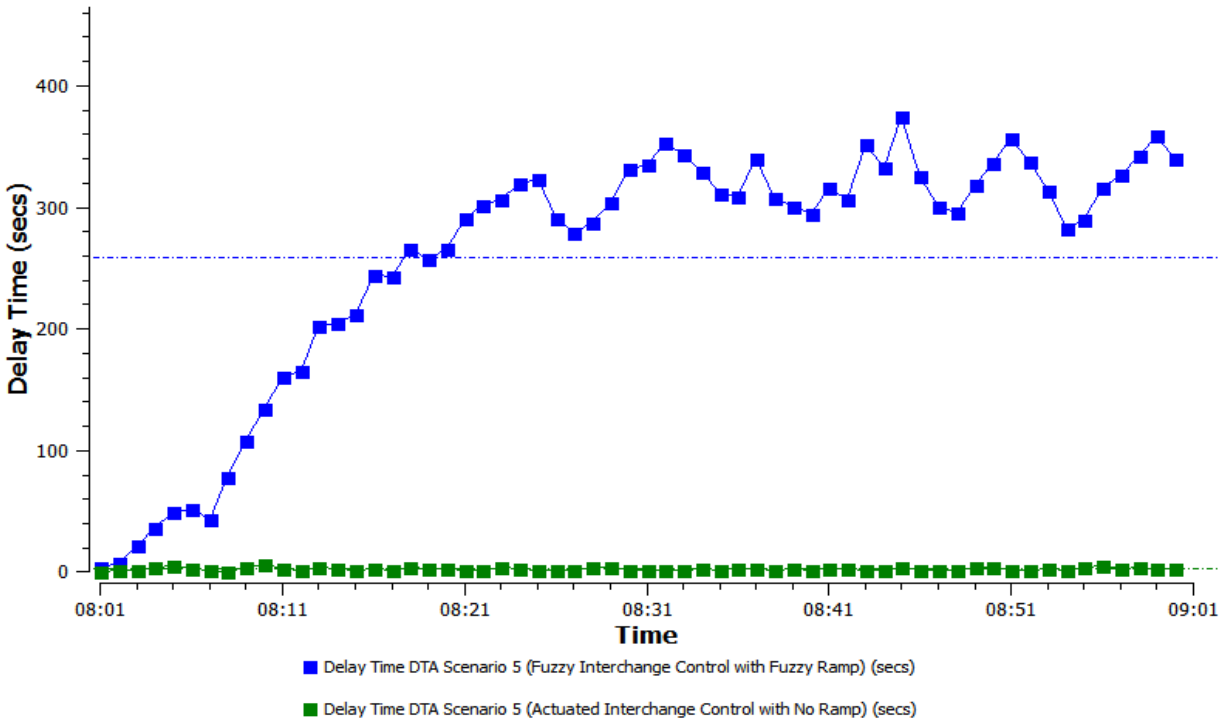


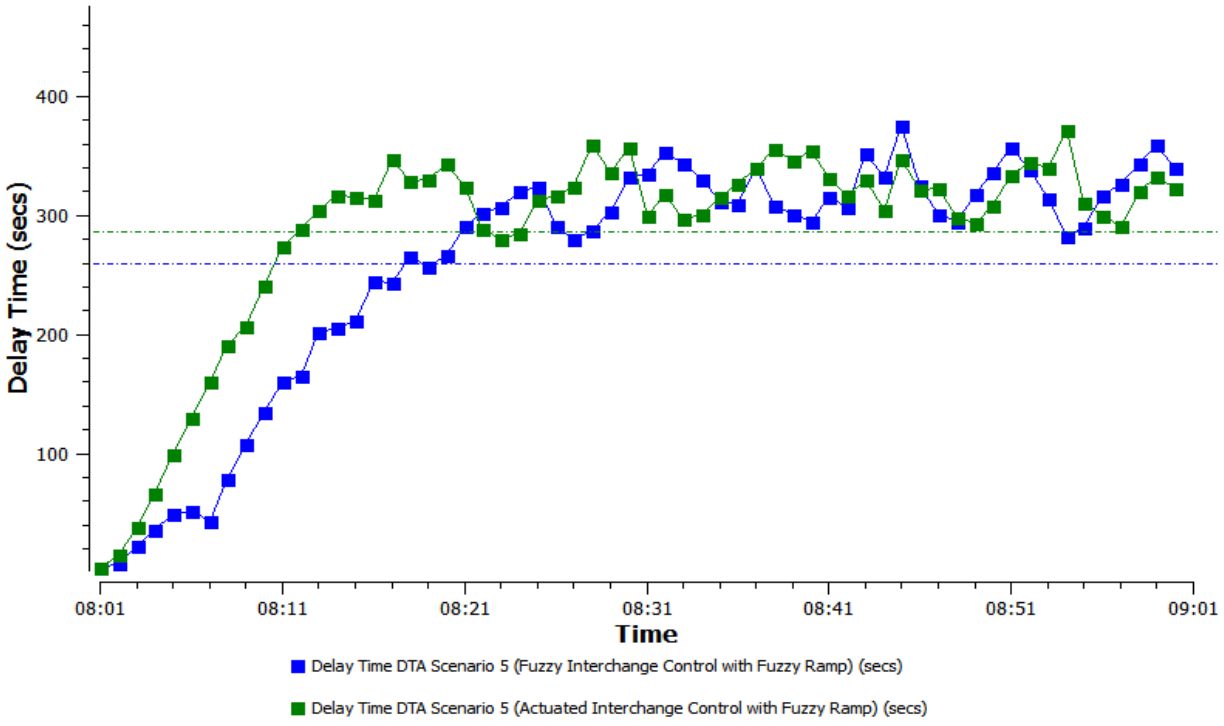
SCENARIO 5
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



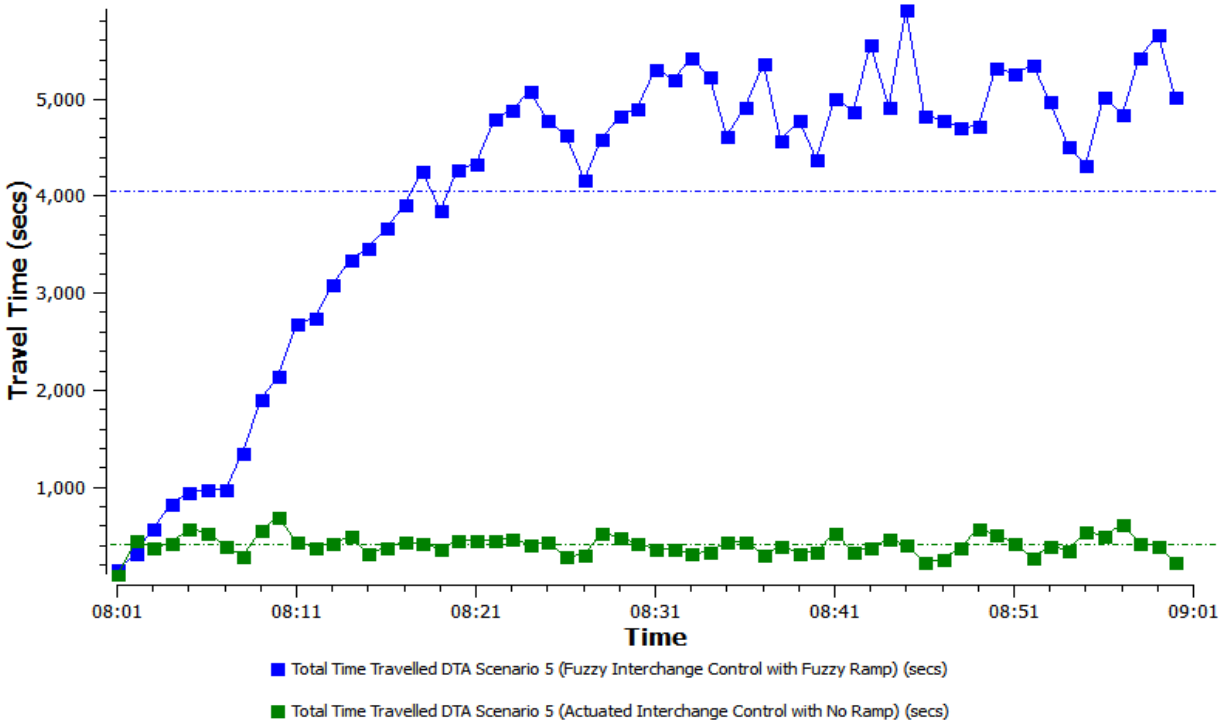


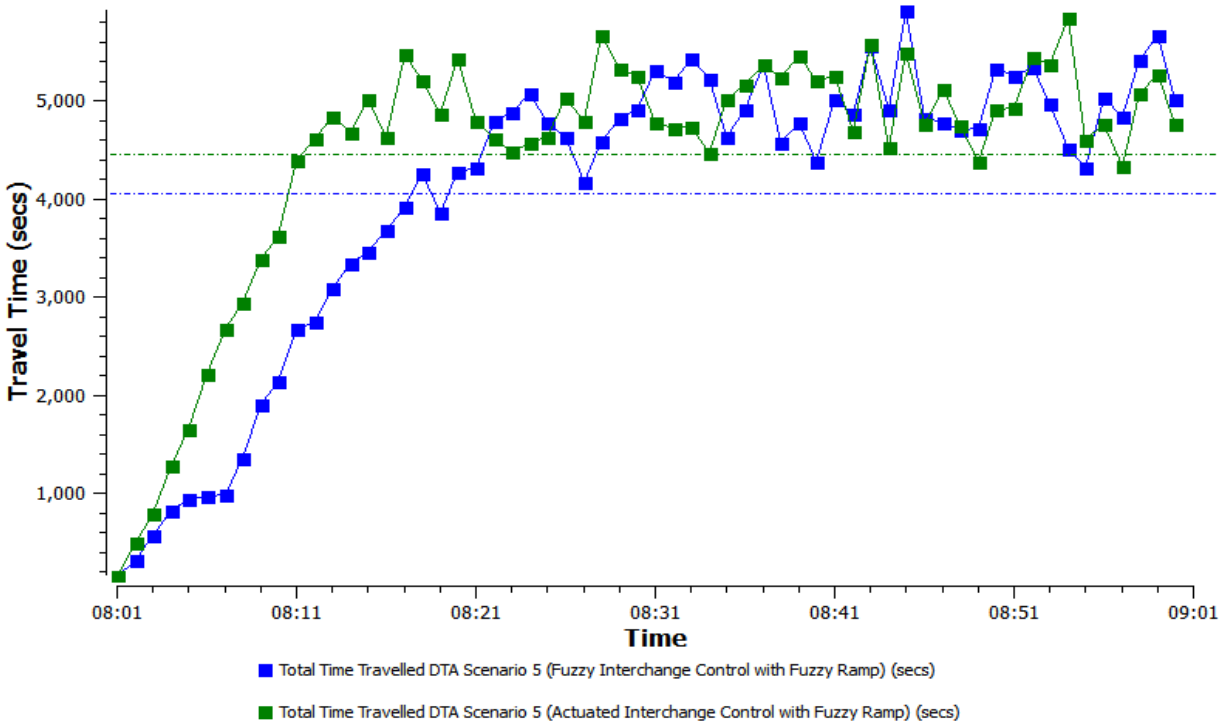
SCENARIO 5
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



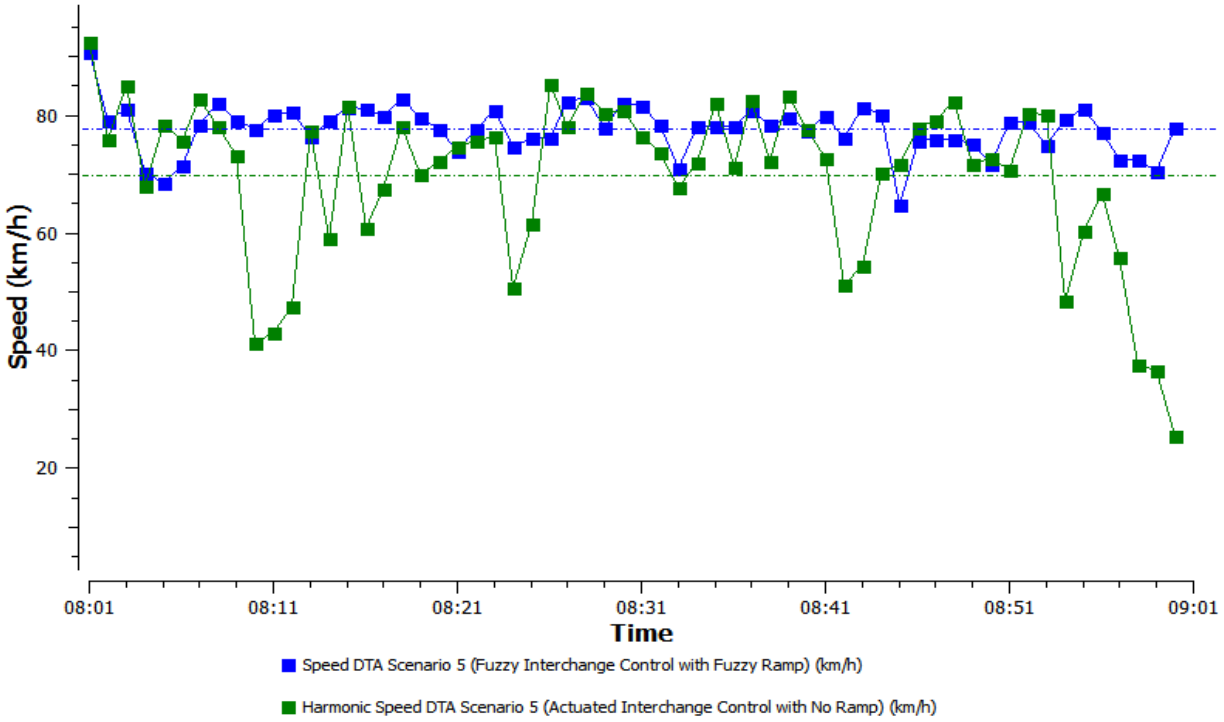


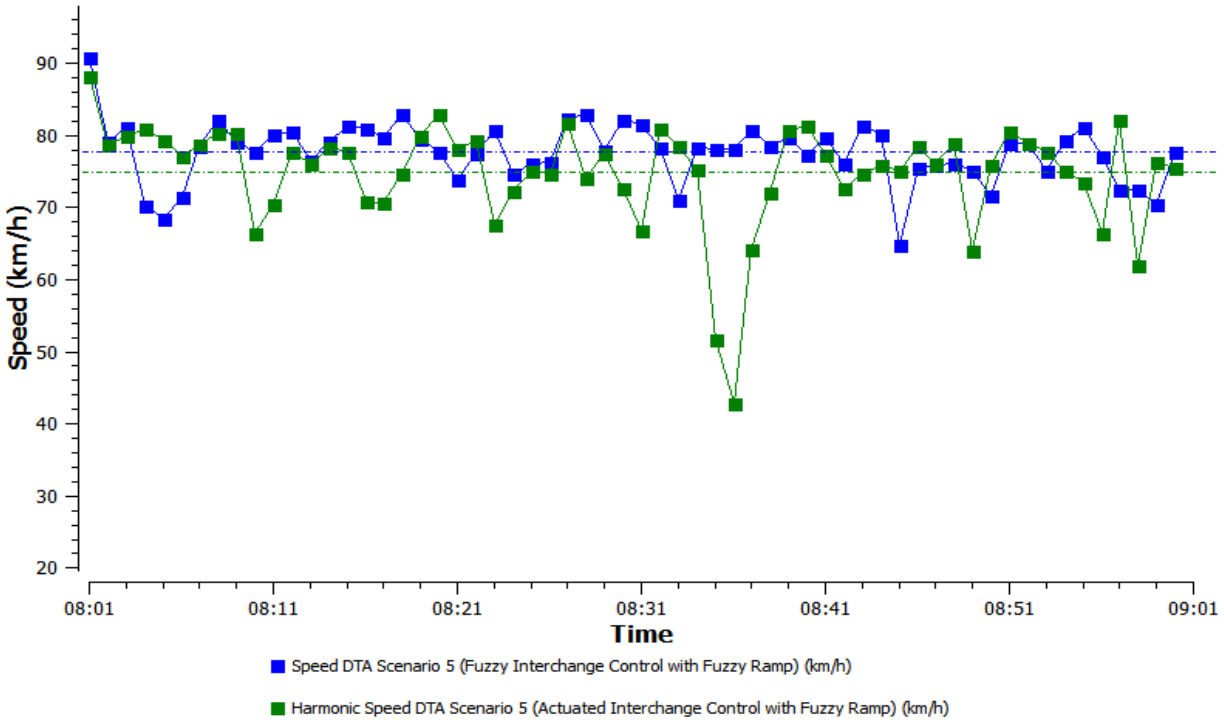
SCENARIO 5
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



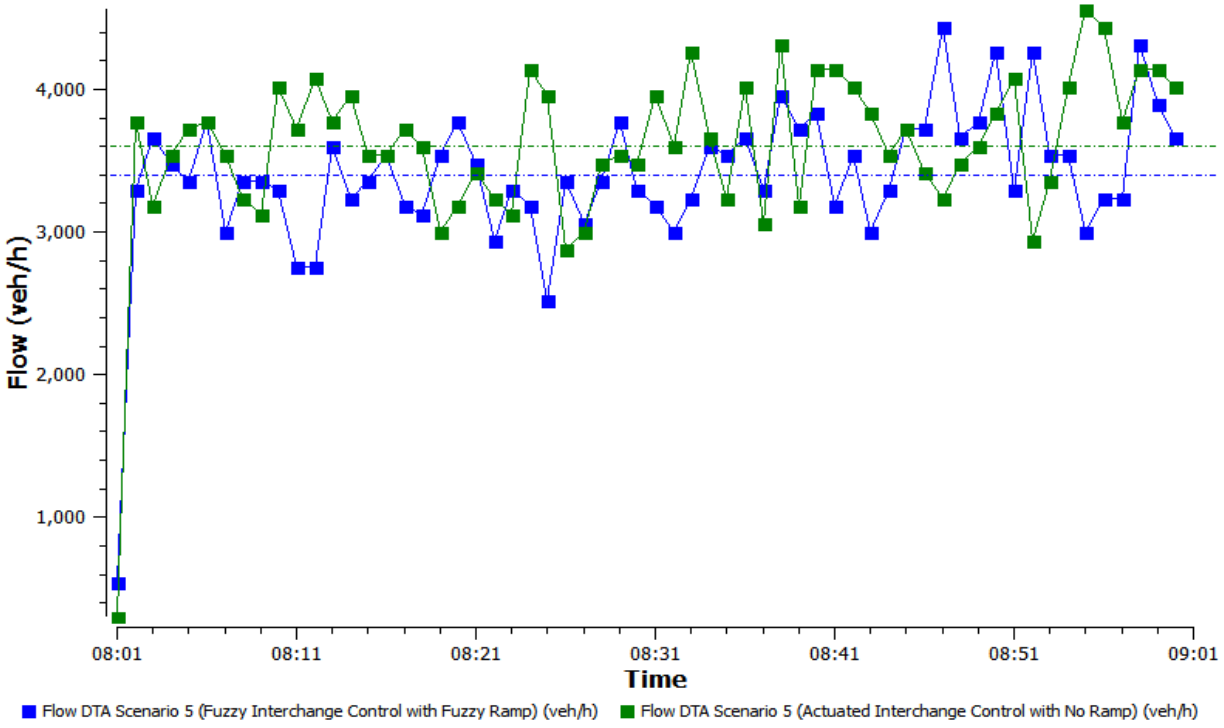


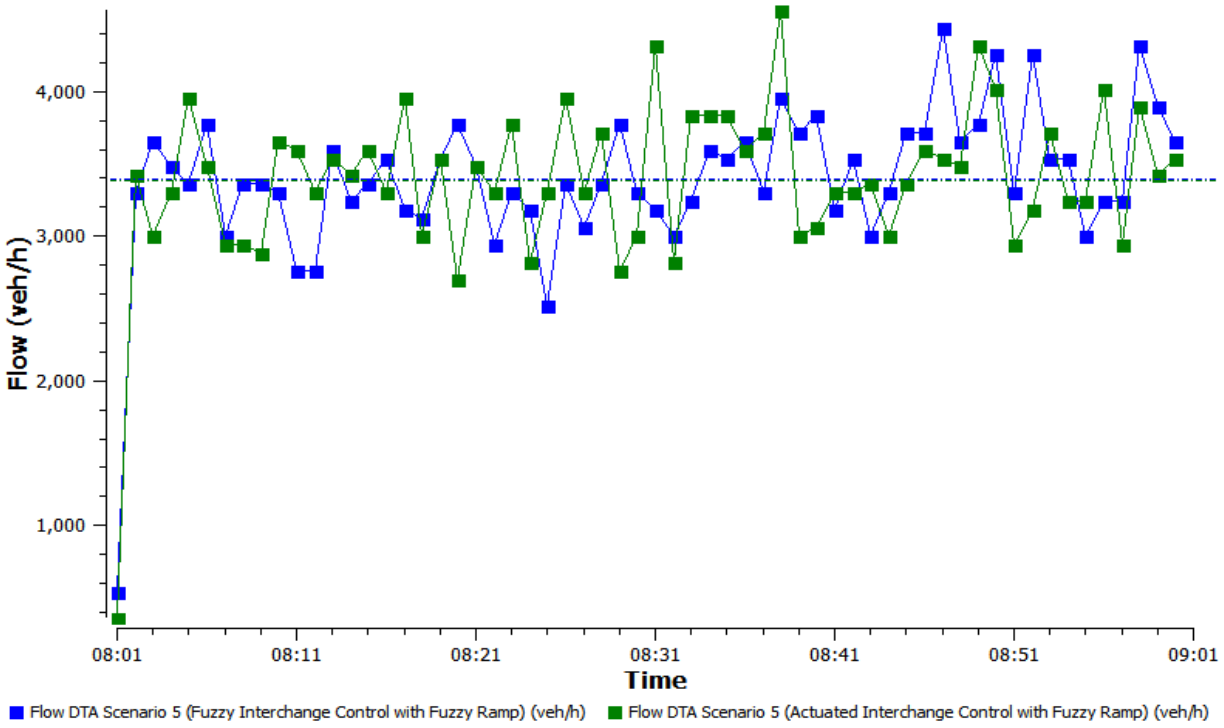
SCENARIO 5
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE SPEED



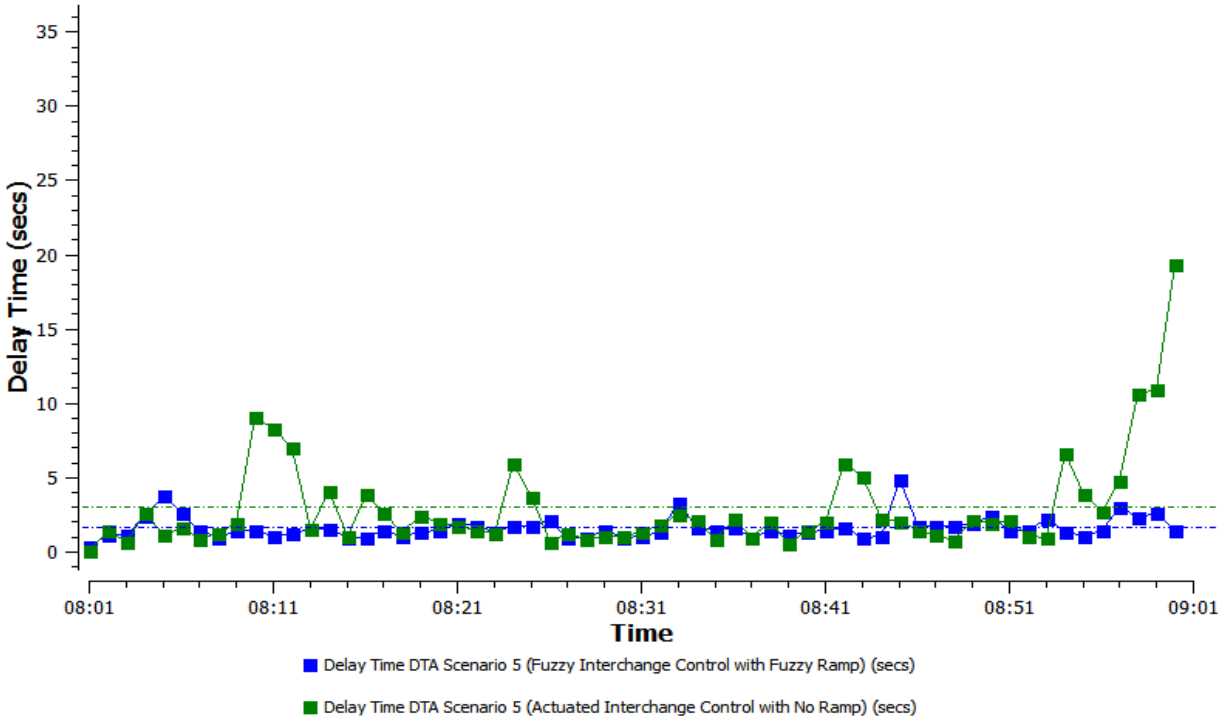


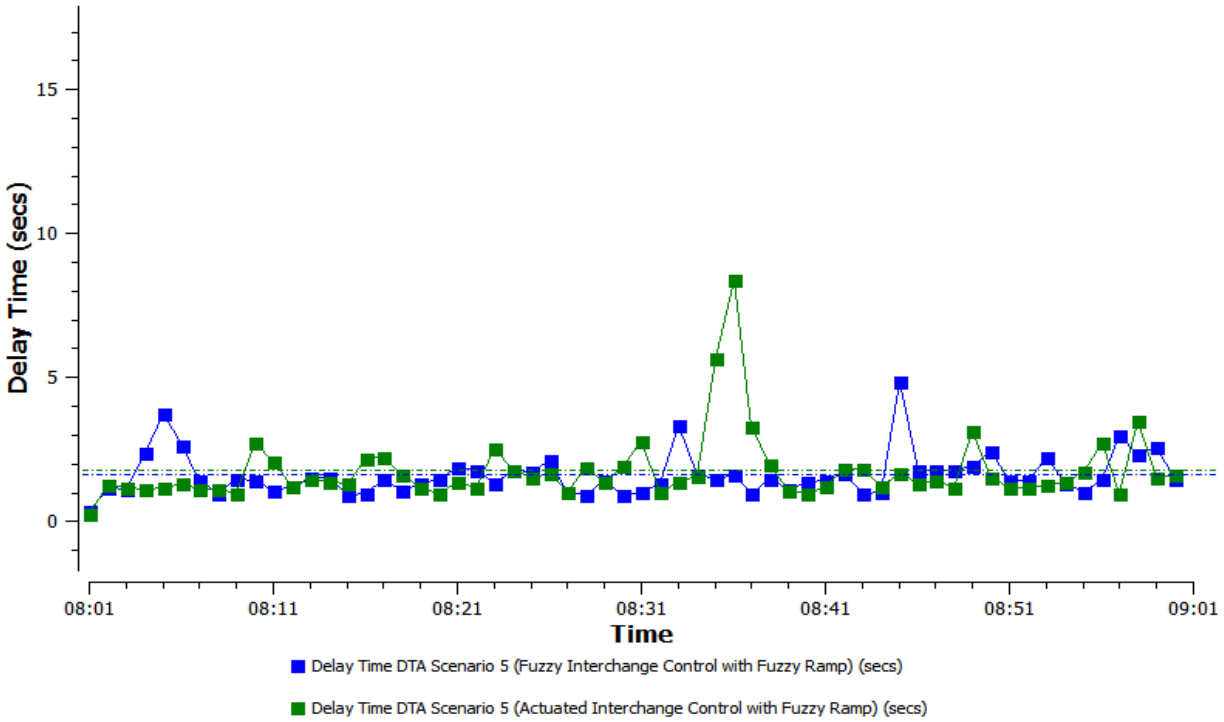
SCENARIO 5
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



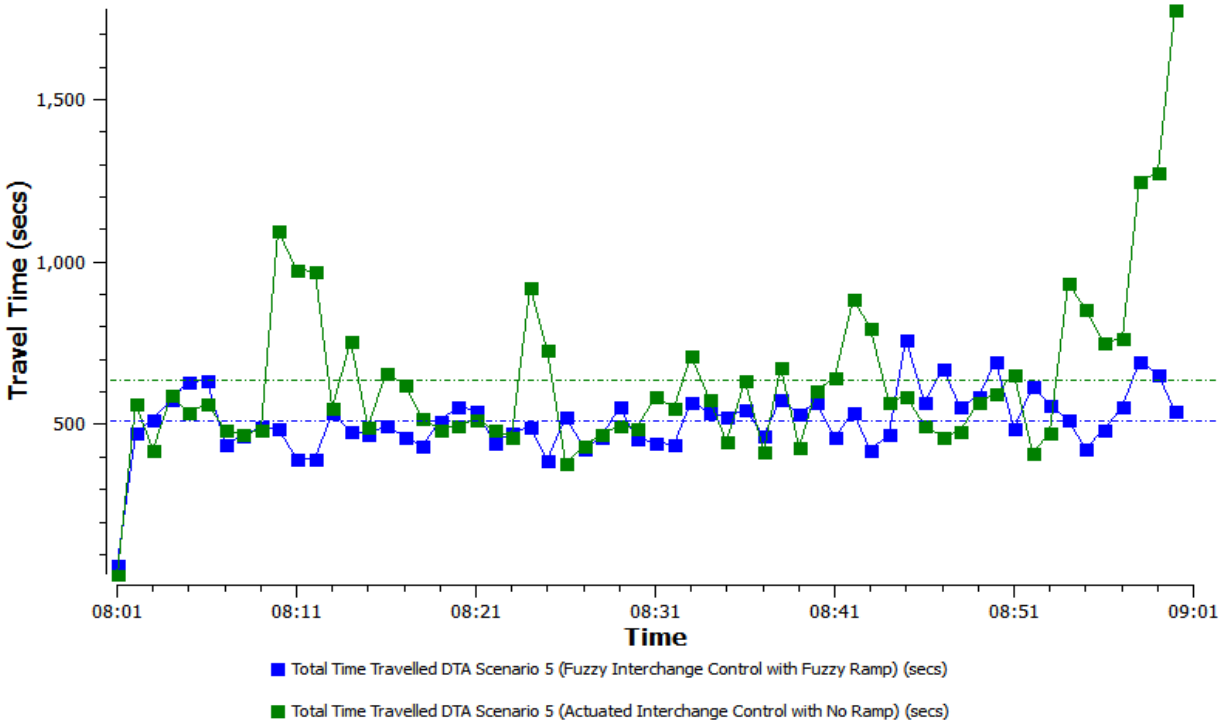


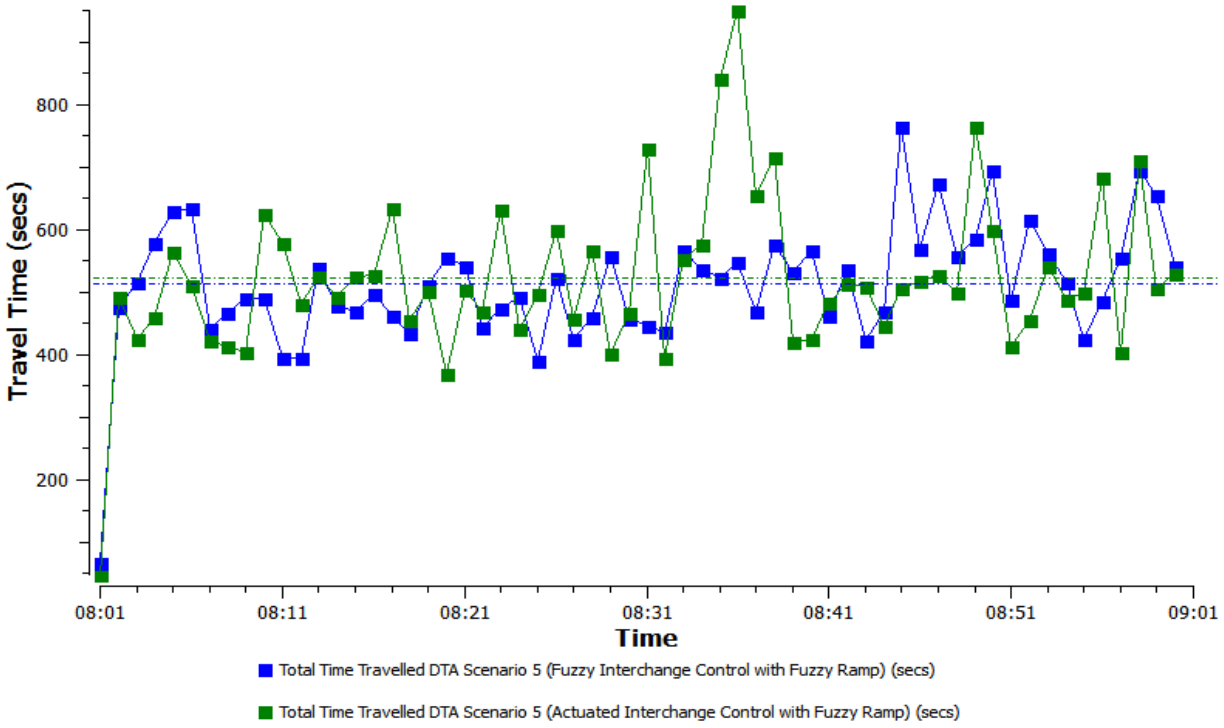
SCENARIO 5
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



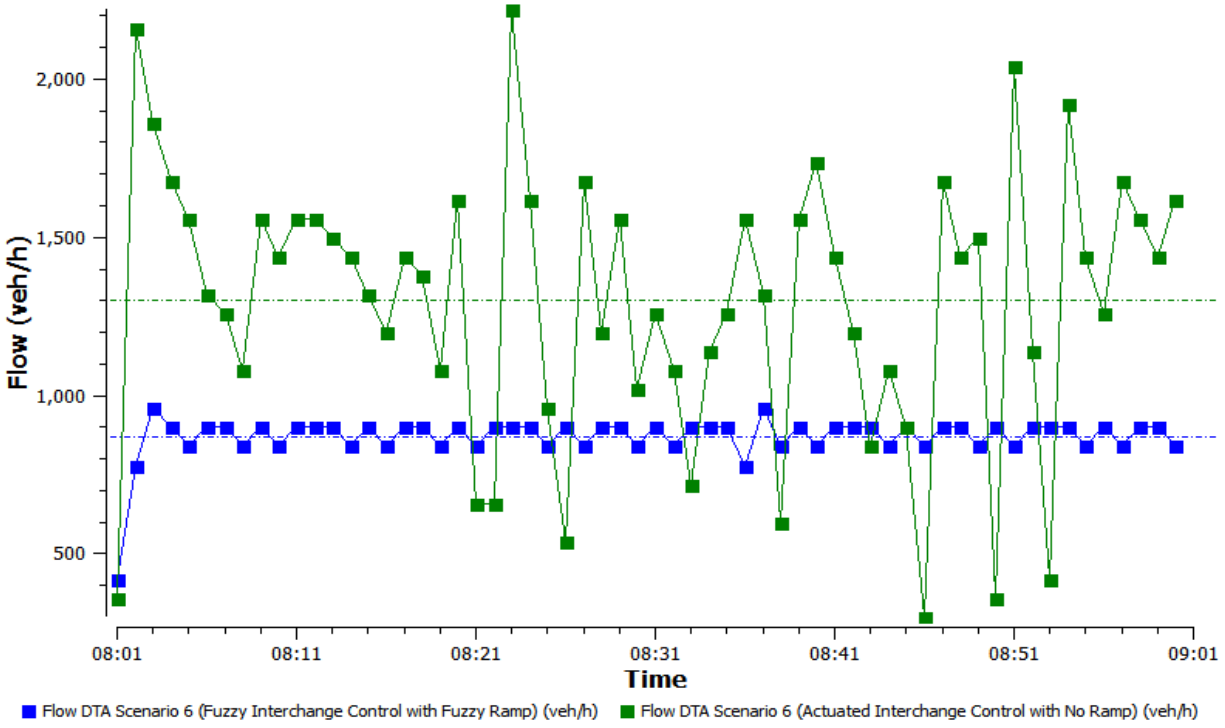


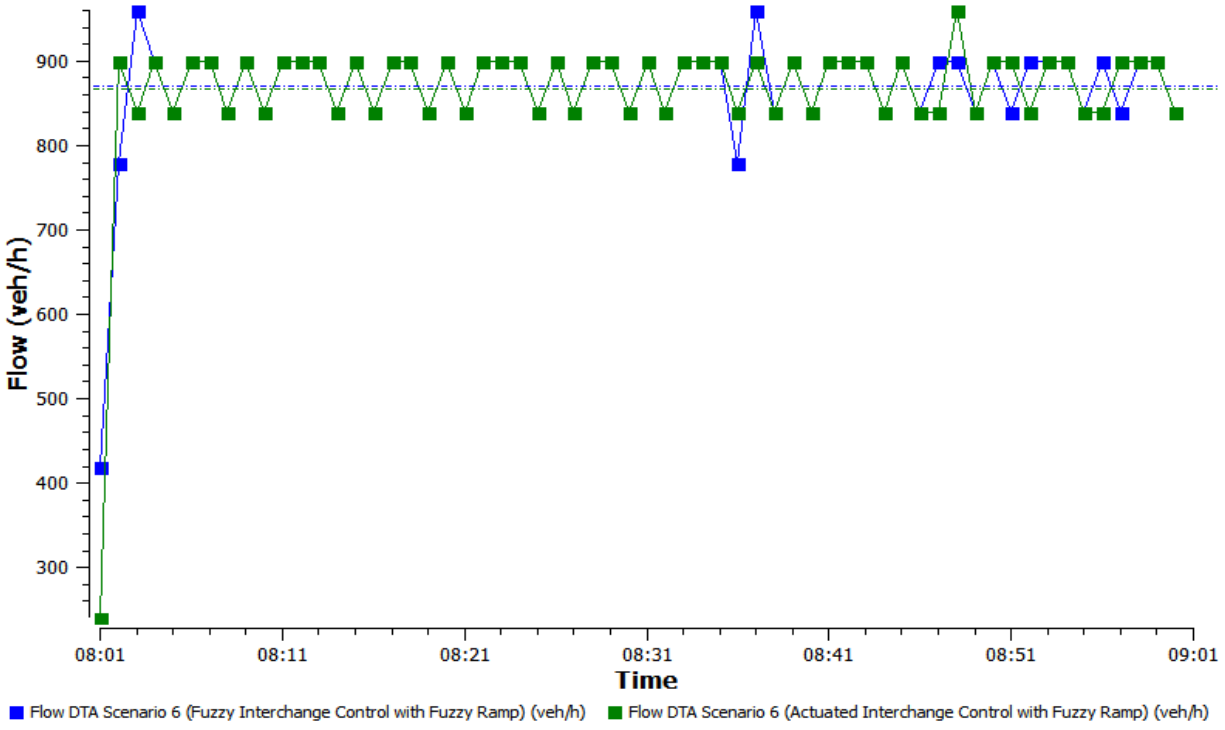
SCENARIO 5
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



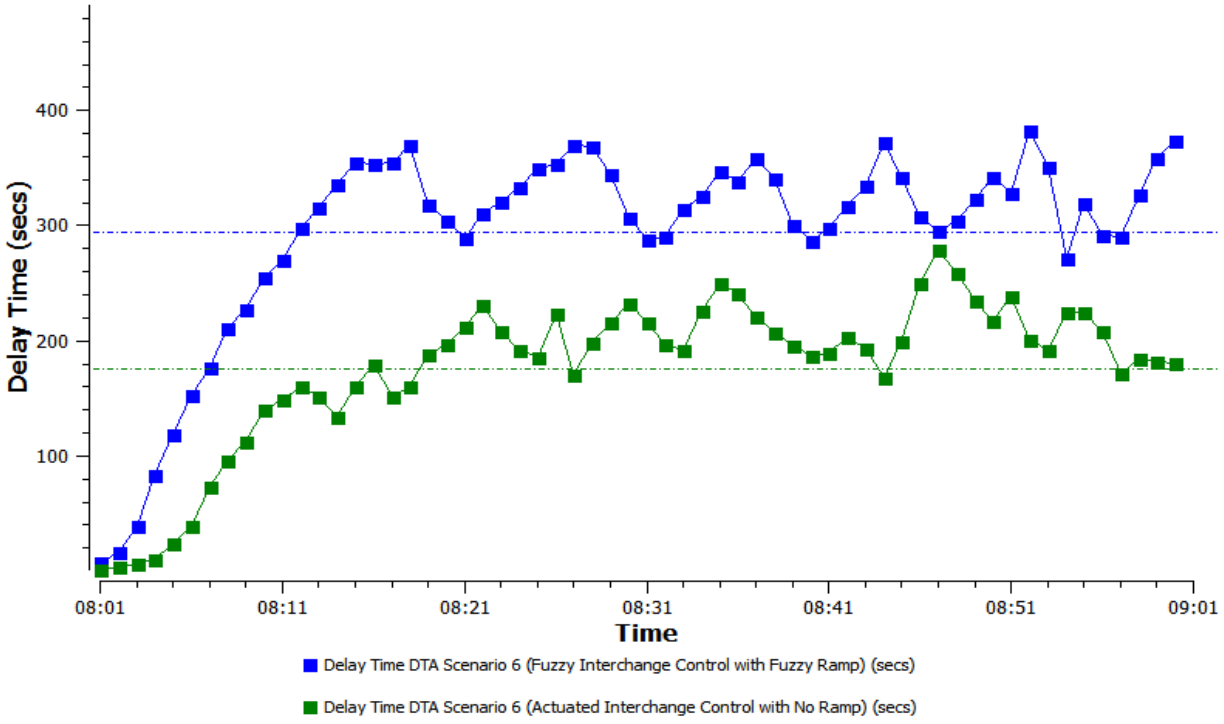


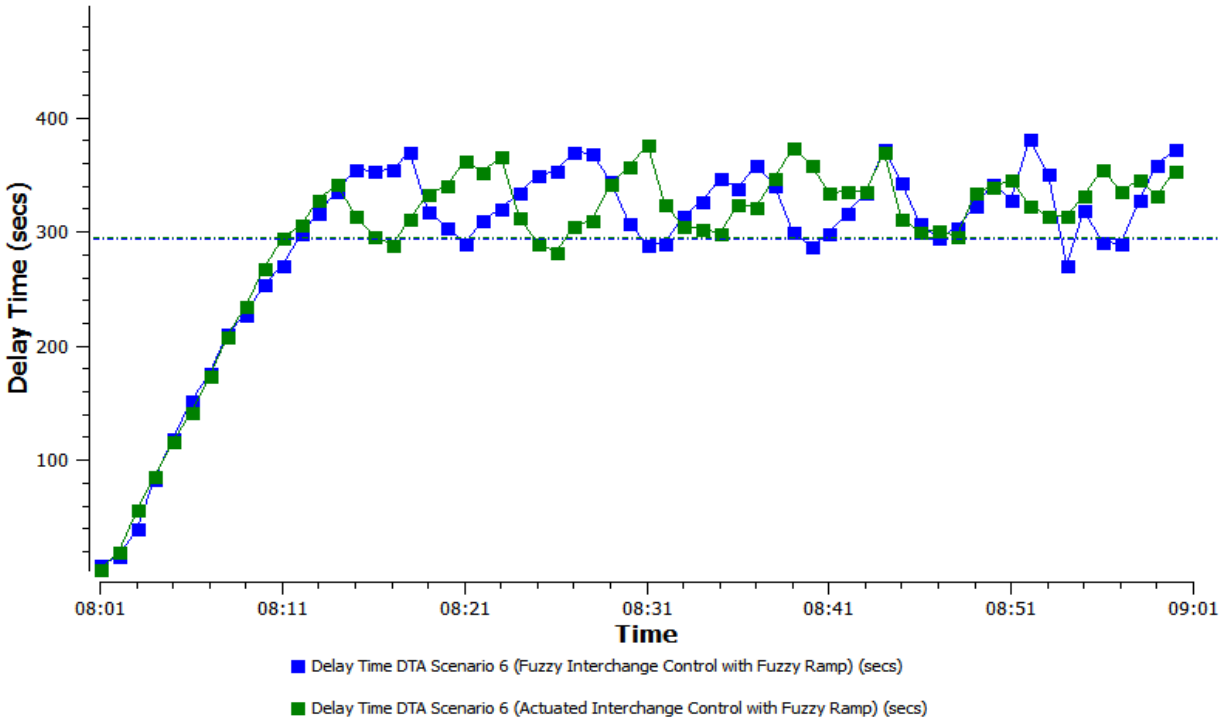
SCENARIO 6
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



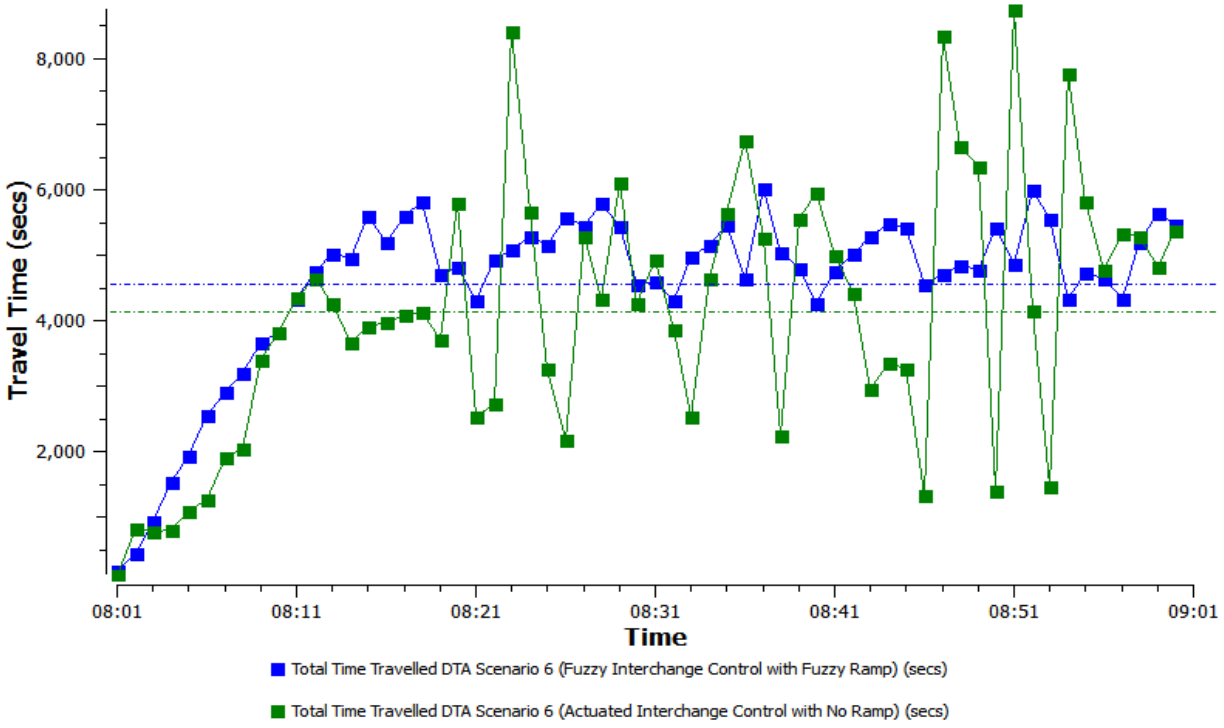


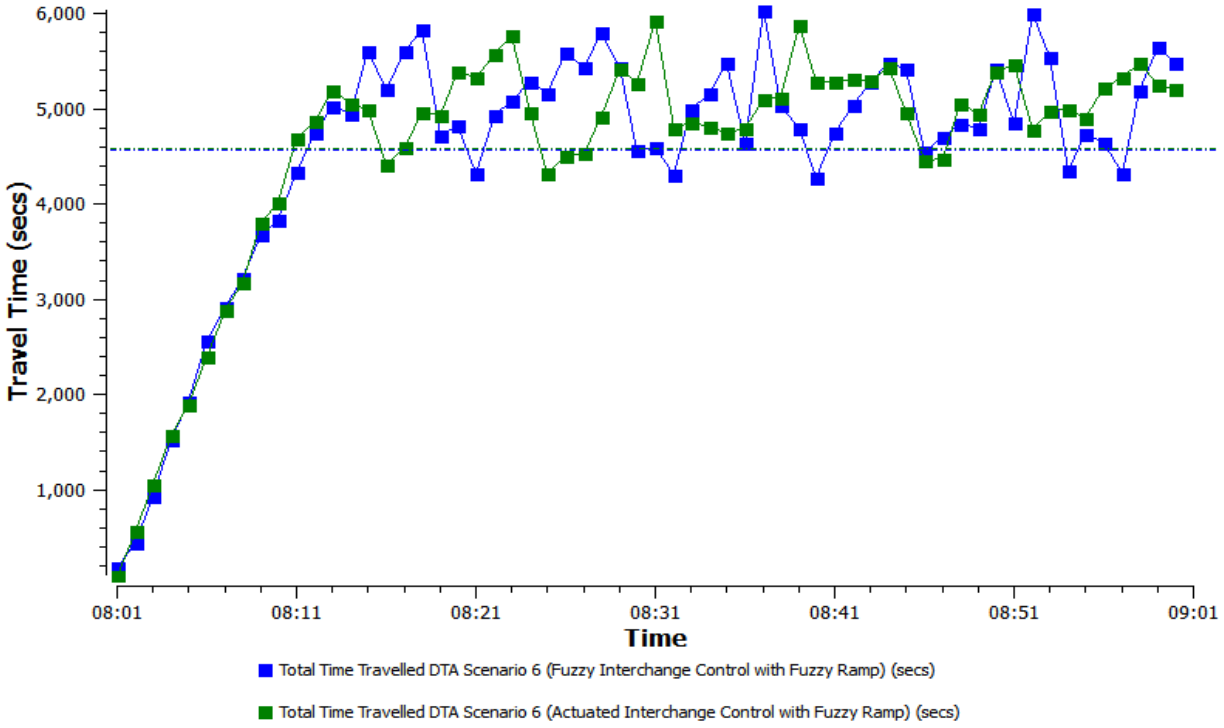
SCENARIO 6
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY



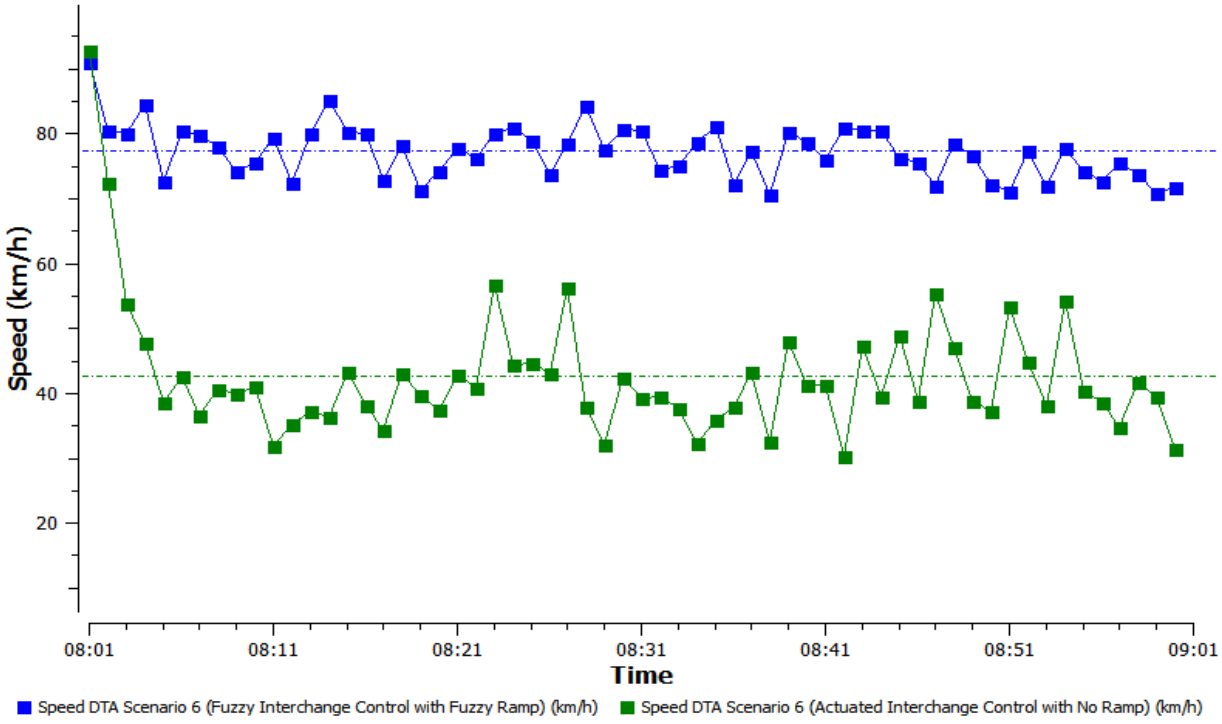


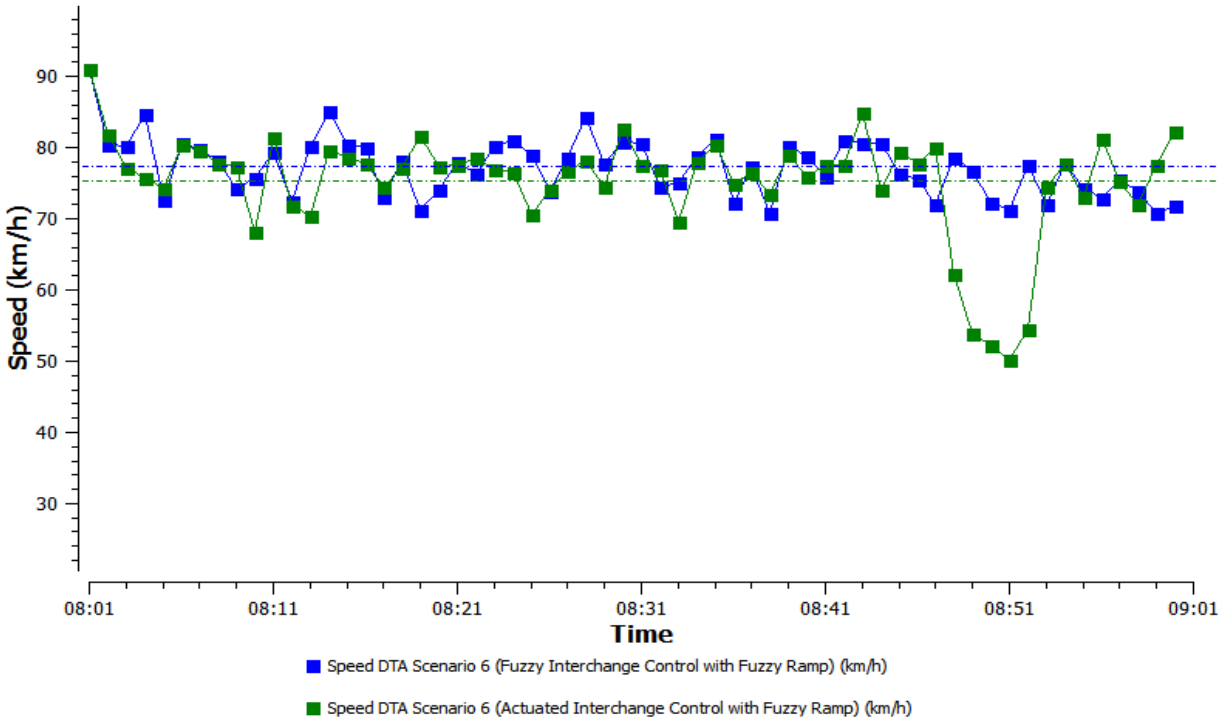
SCENARIO 6
RAMP MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME



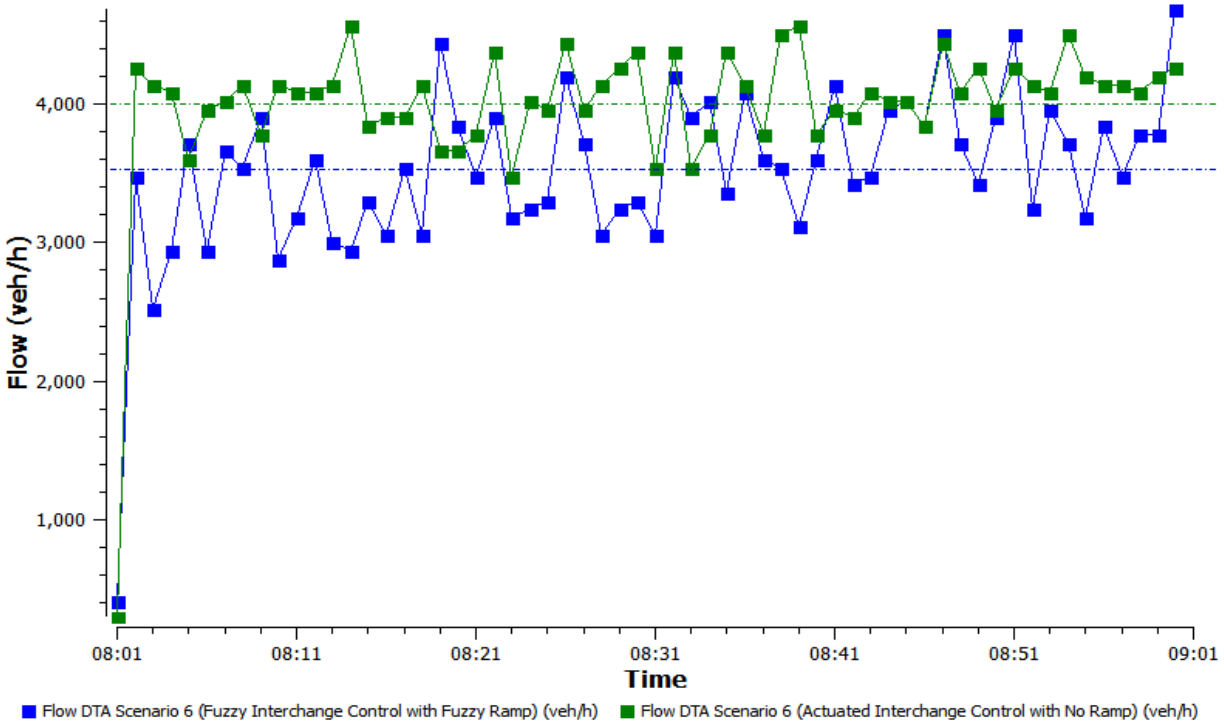


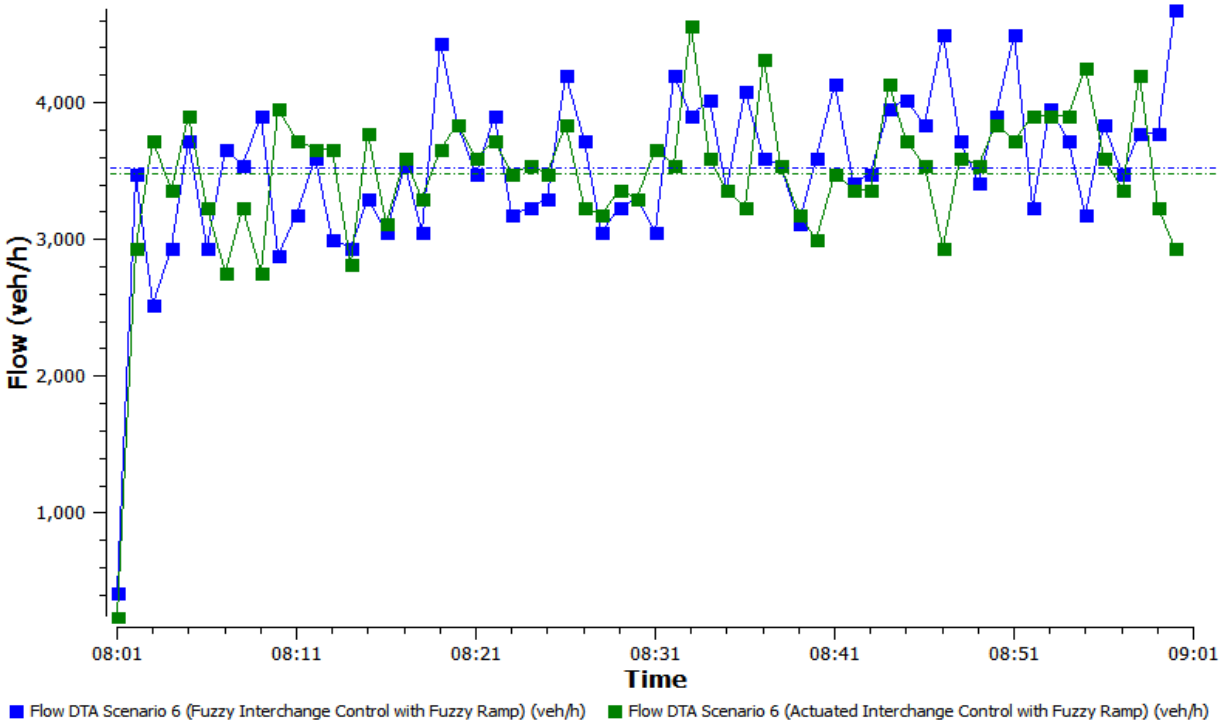
SCENARIO 6
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE SPEED



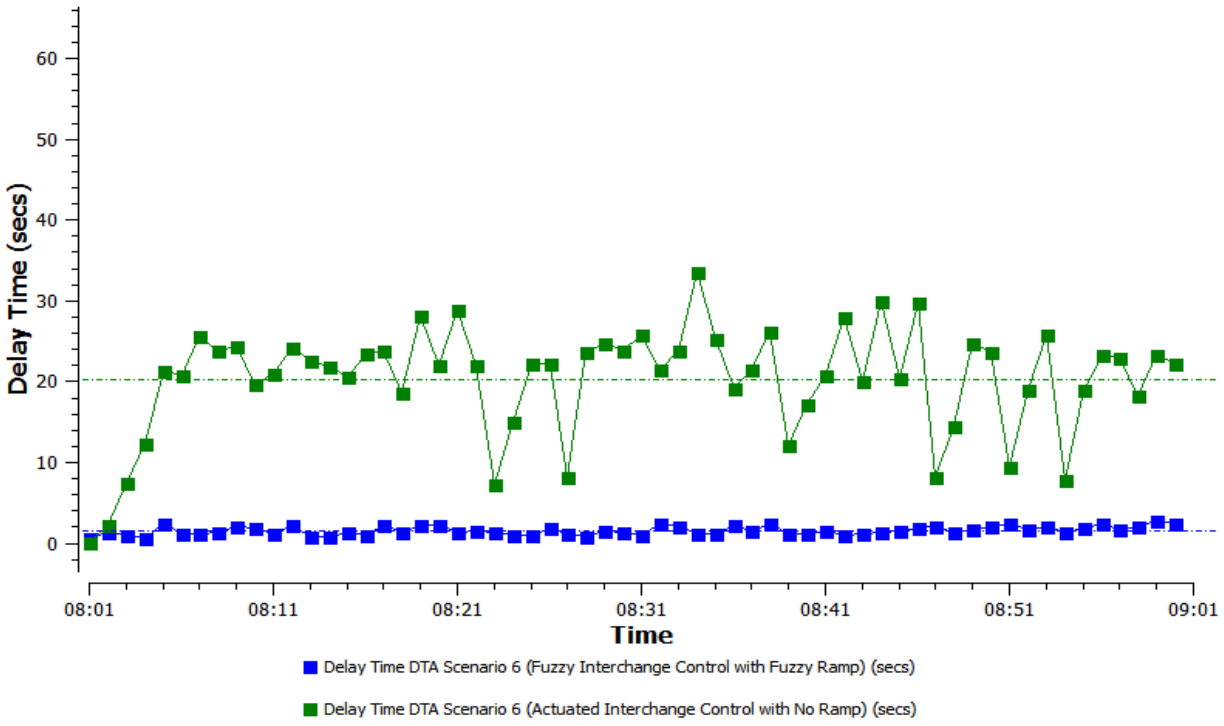


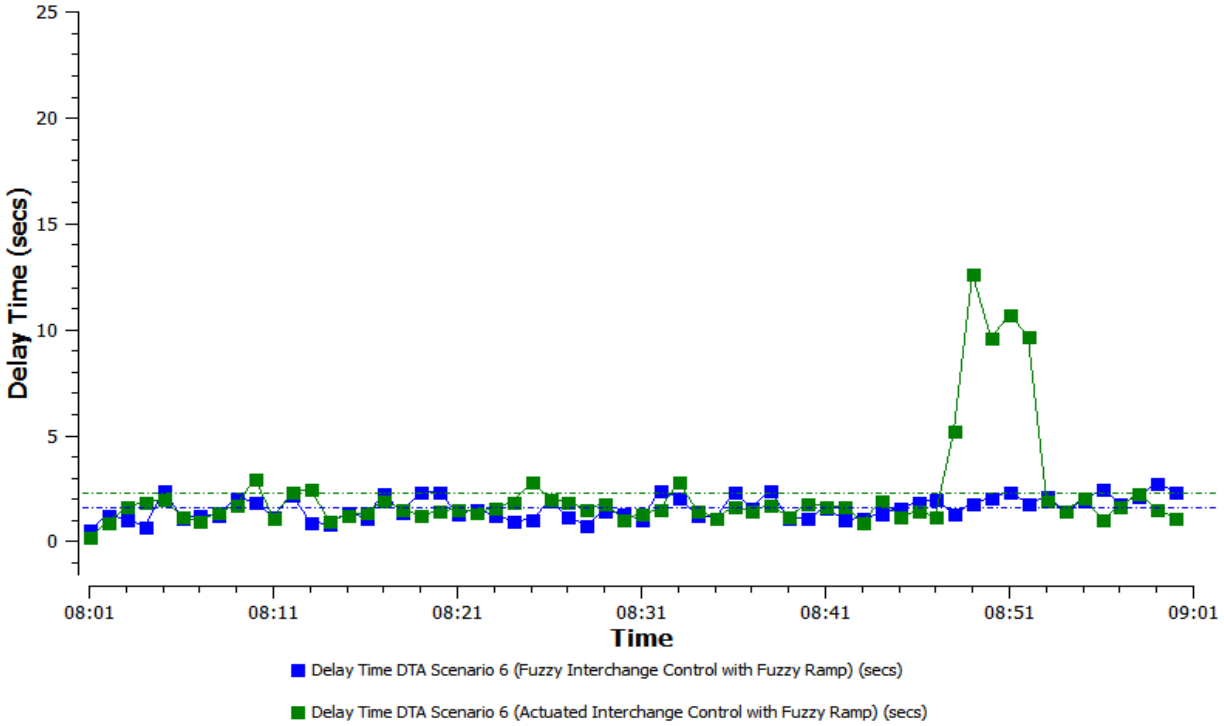
SCENARIO 6
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE FLOW



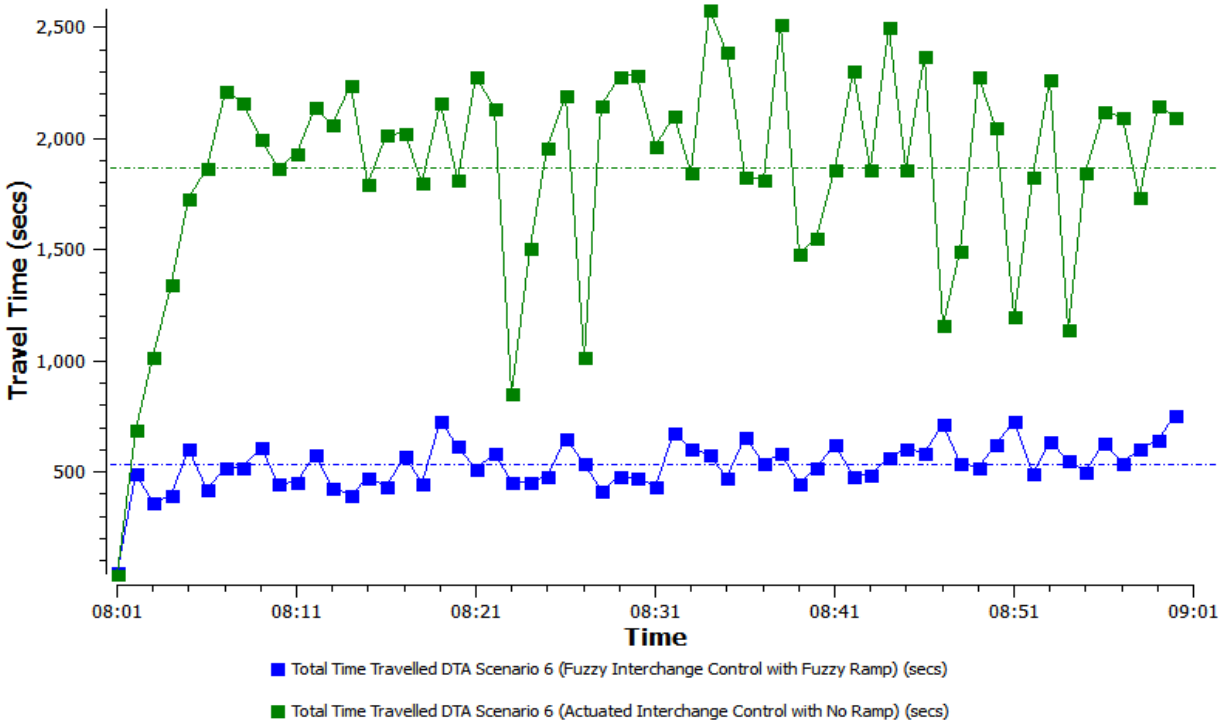


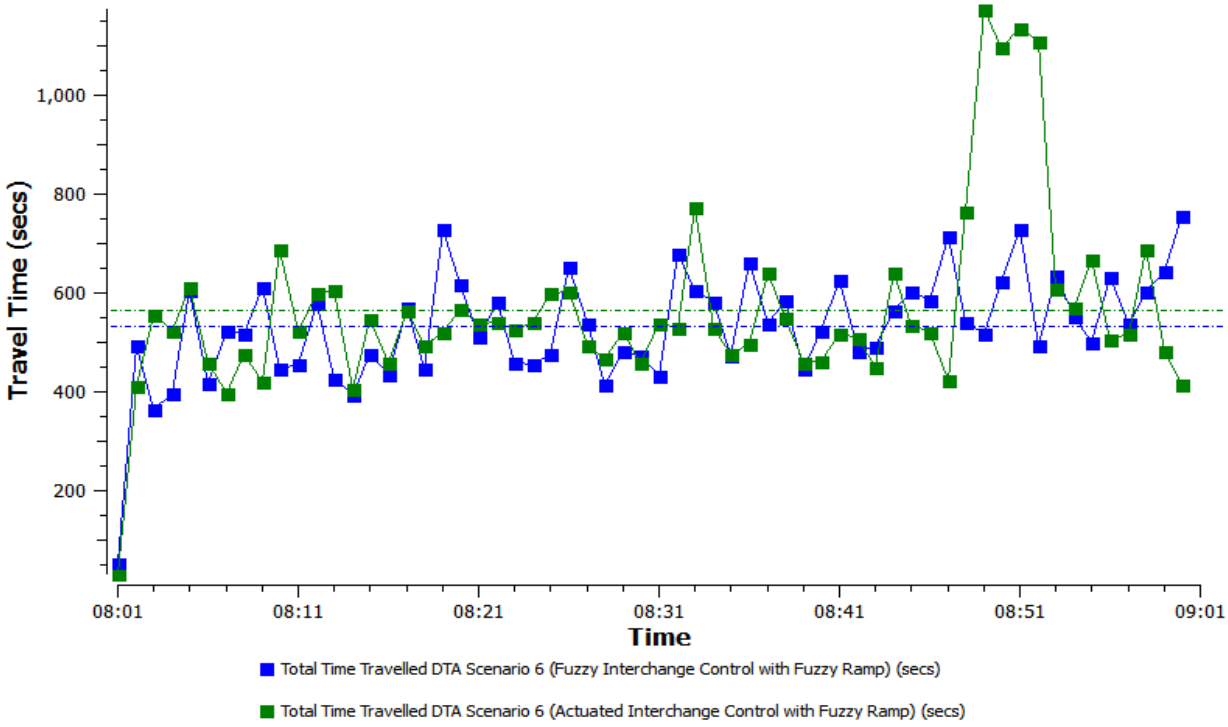
SCENARIO 6
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
AVERAGE DELAY





SCENARIO 6
DOWNSTREAM MEASURE OF EFFECTIVENESS BETWEEN THE 3 MODELS
TOTAL TRAVEL TIME





APPENDIX D: TRAFFIC SURVEY PICTURES