

Developing a Cyber-Physical System for Real-Time Proactive Traffic Control and
Management of Mixed Fleet of Vehicles with Various Levels of Autonomy

by

Viswanath Potluri

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved January 2023 by the
Graduate Supervisory Committee:

Pitu Mirchandani, Chair
Xuesong Zhou
Feng Ju
Jorge Sefair

ARIZONA STATE UNIVERSITY

May 2023

ABSTRACT

In this dissertation, a cyber-physical system called MIDAS (Managing Interacting Demand And Supply) has been developed, where the “supply” refers to the transportation infrastructure including traffic controls while the “demand” refers to its dynamic traffic loads. The strength of MIDAS lies in its ability to proactively control and manage mixed vehicular traffic, having various levels of autonomy, through traffic intersections. Using real-time traffic control algorithms MIDAS minimizes wait times, congestion, and travel times on existing roadways.

For traffic engineers, efficient control of complicated traffic movements used at diamond interchanges (DI), which interface streets with freeways, is challenging for normal human driven vehicular traffic, let alone for communicationally-connected vehicles (CVs) due to stochastic demand and uncertainties. This dissertation first develops a proactive traffic control algorithm, MIDAS, using forward-recursion dynamic programming (DP), for scheduling large set of traffic movements of non-connected vehicles and CVs at the DIs, over a finite-time horizon. MIDAS captures measurements from fixed detectors and captures Lagrangian measurements from CVs, to estimate link travel times, arrival times and turning movements. Simulation study shows MIDAS’ outperforms (a) a current optimal state-of-art optimal fixed-cycle time control scheme, and (b) a state-of-art traffic adaptive cycle-free scheme.

Subsequently, this dissertation addresses the challenges of improving the road capacity by platooning fully autonomous vehicles (AVs), resulting in smaller headways and greater road utilization. With the MIDAS AI (Autonomous Intersection) control, an effective platooning strategy is developed, and optimal release sequence of AVs is

determined using a new forward-recursive DP that minimizes the time-loss delays of AVs. MIDAS AI evaluates the DP decisions every second and communicates optimal actions to the AVs.

Although MIDAS AI's exact DP achieves optimal solution in almost real-time compared to other exact algorithms, it suffers from scalability. To address this challenge, the dissertation then develops MIDAS RAIC (Reinforced Autonomous Intersection Control), a deep reinforcement learning based real-time dynamic traffic control system for AVs at an intersection. Simulation results show the proposed deep Q-learning architecture trains MIDAS RAIC to learn a near-optimal policy that minimizes the total cumulative time loss delay and performs nearly as well as the MIDAS AI.

ACKNOWLEDGMENTS

First, I would like to express my deepest gratitude to my advisor & committee chair, Dr. Pitu Mirchandani for his enormous support all through my PhD journey. I'm so glad that I took Dr. Mirchandani network flows class in the final semester of my master's degree, where I got this opportunity to collaborate with him on cutting-edge research problems. I greatly appreciate his efforts towards admitting me into the PhD program. I would also like to thank him for the years of financial support through research assistantship and opening doors to new cutting-edge research in traffic and transportation. I am also thankful to my PhD committee members Dr. Xuesong Zhou, Dr. Jorge Sefair and Dr. Feng Ju for serving my committee and providing valuable feedback on my research work. I would like to extend my sincere thanks to Mark Poppe and Brent A. Cain from Arizona's Department of Transportation for their valuable collaboration and providing the detailed information about the diamond interchange at I-17 and 19th Ave in Phoenix, Arizona.

I would like to thank Kerem Demirtas for his guidance during our discussions in ATLAS research center and appreciate for being a great team player and help us win the "Re-imagining Transportation" contest held at Friends of Transit 15th annual conference, 2017. Thank you, Dening Peng (Polo), for being a great company during the TSL conference at Chicago. I would also like to extend thanks to my friends Kishore and Uday for their encouragement.

I'm greatly indebted to my dad Satish Potluri, and my mom Pramila Rani for their unconditional love and support without which I couldn't have undertaken this journey. I'm also grateful to my sister Madhuri Gopal and my grandparents for their constant

encouragement. Finally, I would like to thank my wife Winny for her unconditional support, believing in my strengths and encouraging me to pursue my goals.

TABLE OF CONTENTS

	Page
LIST OF TABLES	ix
LIST OF FIGURES	x
LIST OF ALGORITHMS	xiv
CHAPTER	
1 CONTROLLING MIXED TRAFFIC OF CONNECTED AND NON-CONNECTED VEHICLE TRAFFIC THROUGH A DIAMOND INTERCHANGE	
1.1 Introduction to Traffic Signal Control	1
1.2 Controlling a Diamond Interchange.....	5
1.2.1 Some Signal Plans at Diamond Interchanges used by Traffic-Engineers	9
1.2.2 MIDAS Signal Stages at Diamond Intersections	13
1.3 MIDAS Proactive Traffic Control Strategy.....	16
1.3.1 Optimal Fixed Time Signal Control (OFTC).....	16
1.3.2 MIDAS Underlying Architecture and Algorithms.....	18
1.3.2.1 Estimation and Prediction of Vehicle Arrivals at Intersections.....	19
1.3.2.2 MIDAS Control Optimization Problem	24
1.3.2.3 Dynamic Programming Approach for Control Optimization	26

1.3.2.4	Dealing with Limited Internal Storage Capacity at the Interchange	28
1.4	Evaluation and Results	30
1.4.1	Average Delay	33
1.4.2	Average Stops.....	34
1.4.3	Average Queue Length	35
1.4.4	Market Penetration	36
2	PROACTIVE TRAFFIC CONTROL AND PLATOON MANAGEMENT OF AUTONOMOUS VEHICLES THROUGH AN UNSIGNALIZED INTERSECTION	38
2.1	Introduction to Autonomous Intersection Control and Management	38
2.2	MIDAS AI System Architecture and Development.....	42
2.2.1	MIDAS AI System Overview	42
2.2.2	Platoon Scoping and Management System	47
2.2.2.1	Leader-Follower Behavior	48
2.2.2.2	Autonomous Car Following Model.....	59
2.2.2.3	MIDAS Platooning Strategy	51
2.2.3	MIDAS Autonomous Intersection Scheduling and Control Unit (AISCU).....	55
2.2.3.1	Autonomous Intersection Control Problem Formulation ...	55

CHAPTER	Page
2.2.3.2	Dynamic Programming Approach for Scheduling AV Departures.....57
2.2.3.3	Unsignalized Intersection Control Logic for Autonomous Vehicular Traffic67
2.3	MIDAS AI Simulation Framework71
2.4	Simulation Results.....74
3	REAL-TIME PROACTIVE TRAFFIC CONTROL OF AUTONOMOUS VEHICLES THROUGH UNSIGNALIZED INTERSECTION USING DEEP REINFORCEMENT LEARNING 77
3.1	Introduction to Traffic Signal Control at Intersection Using Reinforcement Learning.....77
3.2	A Comprehensive Overview of Key Concepts in Reinforcement Learning82
3.2.1	Markov Decision Process Framework.....82
3.2.2	Model-Based Reinforcement Learning.....85
3.2.2.1	Policy Evaluation86
3.2.2.2	Policy Iteration86
3.2.2.3	Value Iteration.....86
3.2.3	Model-Free Reinforcement learning.....86
3.3	MIDAS RAIC System Architecture88
3.3.1	Proposed System Model and Deep-RL Problem Formulation88

CHAPTER	Page
3.3.2 DQN Algorithm for Controlling Unsignalized Autonomous Intersection	102
3.4 MIDAS RAIC System Simulation and Evaluation	107
4 CONCLUSIONS AND FUTURE RESEARCH	114
4.1 Dissertation Contributions and Conclusions	114
4.2 Future Research	116
REFERENCES	118
APPENDIX	
A COMMON SIGNAL PHASES USED BY TRAFFIC ENGINEERS AT DIAMOND INTERCHANGES.....	128
B ROAD DESIGNS TO CONSIDER FOR HETEROGENOUS TRAFFIC ENVIRONMENTS.....	138

LIST OF TABLES

Table		Page
1.1	Diamond Interchange Notation to Describe Signal Timings	6
1.2	“4-Phase with Overlaps” Signal Stages	13
1.3	MIDAS Signal Stages	15
1.4	Network Level Performance	32
1.5	Interchange Level Performance	32
2.1	Improved Intelligent Driver Model Control Parameters (No Platooning Scenario)	74
2.2	Cooperative Adaptive Cruise Control Model Parameters (Platooning Scenario)	75
3.1	Distribution of Traffic on Intersection Approaches	108
3.2	Description of MIDAS RAIC DQN Training Parameters	108
3.3	Performance Comparison Between MIDAS RAIC and MIDAS AI	112
3.4	Run Time Comparison Between MIDAS RAIC and MIDAS AI	113
A.1	“3-Phase Lag-Lag” Signal Stages Used in DP	129
A.2	“4-Phase Lead/Lag – Lag” Signal Stages Used in DP	130
A.3	“4-Phase Lead/Lag – Lag” Signal Stages Used in DP	131
A.4	“3-Phase with Overlaps” Signal Stages	133

LIST OF FIGURES

Figure	Page
1.1	Layout of a Diamond Interchange with Freeway Underpass.....6
1.2	Phase $\Phi 3$ Example Shown with Movements in Bold Green8
1.3	Signal Stage Example Shown with Concurrent Movements in Bold Green.....9
1.4.1	“4-Phase with Overlaps” Phase Movement..... 11
1.4.2	“4-Phase with Overlaps” Signal Scheme..... 12
1.5.1	MIDAS Phase Movement 14
1.5.2	MIDAS DP Decision Process 14
1.6	Projection of Queue Over a Time Horizon.....22
1.7	Sequence (Top to Bottom) of Snapshots for Queue Estimation Logic23
1.8	DP Solution Representation: an Example of Optimal Phases and their Durations Over Time Horizon T27
1.9	The Lane Weight vs Queue Length for I-17/19 th Ave., Phoenix, AZ Interchange30
1.10	I-17/19 th AVE., Phoenix Diamond Interchange in VISSISM.....31
1.11	Comparison of Average. Delays vs Traffic Load for Different Signal Control Strategies.....34
1.12	Comparison of Average Number of Stops vs Traffic Load for Different Signal Control Strategies35
1.13	Comparison of Average Queue Length vs Traffic Load for Different Signal Control Policies36

Figure	Page
1.14 MIDAS Performance Graph for Different Market Penetration Rates for Traffic Load 6800 Vehicles/Hr.	37
2.1 MIDAS-AI System Architecture	43
2.2 MIDAS-AI System Concept of Operations.....	44
2.3 PSMS Concepts of Operation	45
2.4 AISCU Concept of Operations	47
2.5 Newell Car Following Model	48
2.6 MIDAS Platooning Strategy	52
2.7 DP Solution with Release Sequence of Vehicles Approaching the Intersection.....	59
2.8 Car Following Model Activation Logic for AVs	66
2.9 Unsignalized Intersection Control Logic for Autonomous Vehicles.....	69
2.10 Autonomous Intersection Control & Management of Unsignalized Intersection.....	71
2.11 Vehicular Communication Architecture.....	72
2.12 PLEXE Configuration.....	73
2.13 MIDAS Simulation Architecture	73
2.14 MIDAS AI Throughput Performance with & without Platooning Strategy.....	76
2.15 MIDAS AI Time-Loss Delay Performance with & without Platooning Strategy	76
3.1 Artificial Intelligence Sub Fields and Conceptual Overlap	81
3.2 Basic Reinforcement Learning Feedback Loop	83

Figure	Page
3.3 Q-Learning Algorithm Overview	88
3.4 Illustration of Vehicles State Representation for a 4-Legged Unsignalized Intersection.....	93
3.5 Unsignalized Intersection Control Logic for Autonomous Vehicles.....	95
3.6.a Illustration of 4-legged Unsignalized Intersection	96
3.6.b Agent’s ϵ -greedy policy	98
3.6.c Illustration of New Agent Action Step	99
3.7 Illustration of The Training Procedure for MIDAS RAIC	106
3.8 Illustration of Training with Experience Replay for MIDAS RAIC	106
3.9 MIDAS RAIC DQN Architecture Overview	107
3.10 Training Performance of MIDAS RAIC in Earning Average Reward	111
3.11 Training Performance of MIDAS RAIC in Minimizing Avg. Time Loss Delay of Vehicles	111
3.12 Comparison of Average Time Loss Delay in MIDAS RAIC and MIDAS AI.....	112
A.1 “3-Phase Lag-Lag” Phase Movements	129
A.2 “3-Phase Lag-Lag” Signal Plan	129
A.3 “4-Phase Lead/Lag - Lag” Phase Movement	130
A.4 “4-Phase Lead/Lag – Lag” signal scheme	130
A.5 “4-Phase Lead/Lag” phase movement	131
A.6 “4-Phase Lead/Lag” Signal Scheme	131
A.7 “3-Phase with Overlaps” Phase Movement.....	132

Figure	Page
A.8 “3-Phase with Overlaps” Signal Scheme.....	132
B.1 Road Designs for Heterogenous Vehicular Traffic Environments	135

LIST OF ALGORITHMS

Algorithm	Page
2.1 MIDAS Platooning Strategy for Autonomous Vehicular Traffic.....	54
2.2 MIDAS DP Logic to Determine Optimal AV/Platoon Release Sequence.....	67
2.3 Unsignalized (Traffic-Lights-Free) Intersection Control Logic for AV Traffic	70
3.1 Autonomous Vehicular Traffic State Information Extraction at Agent Step τ	96
3.2 Evaluation of the Start of New Agent Time Step.....	100
3.3 Unsignalized (traffic-lights-free) Intersection Control Logic for AV Traffic	102
3.4 MIDAS RAIC DQN Agent Training.....	109

CHAPTER 1

CONTROLLING MIXED TRAFFIC OF CONNECTED AND NON-CONNECTED VEHICLE TRAFFIC THROUGH A DIAMOND INTERCHANGE

Preview of Contributions

- a) Proposed a data fusion approach to estimate link travel times of vehicles using vehicle GPS data and loop detector data.
- b) Proposed a rolling horizon based dynamic programming to determine a cycle-free optimal signal plan.
- c) Proposed a proactive traffic control architecture for diamond interchanges with limited internal storage capacity.

1.1 Introduction to Traffic Signal Control

Traffic signals are crucial for controlling the conflicts among intersecting traffic streams. Basically, green lights allow non-conflicting streams to proceed while the red signal lights stop conflicting streams. Much has been researched on optimal signal timings for a single isolated intersection [3][23][24][51][53]. Effectively, signal timings are described in terms of “phases”, where a phase is a set of green signal lights that allow vehicles to move in non-conflicting directions, for example North and South straight throughs while the red lights stop East and West traffic streams. Even a single green light, for example one that allows only North stream while red lights stop all others, is a valid phase. Hence a timing scheme for a single intersection can have many non-conflicting green phases to control it. Thus, basically designing a signal control scheme is to come up with a sequence of non-conflicting phases where the signal lights corresponding to each phase have specified green durations that optimize a traffic performance function such as minimize total stopped delay.

When traffic signals were first developed and installed, signal timings were mechanically controlled with relays and the timings were set based on manually observed traffic over a period of days. The green times were “*fixed*” in a cyclic fashion, that is, for example, the durations of say four phases A, B, C, D are fixed and repeated over and over. Later, with implemented electric timing controllers with clocks, green-time durations could be provided in a “*time-of-day*” (TOD) fashion, since traffic streams were observed to have time-of day patterns, for example morning rush hours, evening rush hours, other moderate traffic hours, and late-night hours with light traffic. So even before the advent of computerized traffic signals, controlling a single intersection satisfactorily involved (a) developing an appropriate set of non-conflicting green phases and then (b) developing fixed TOD cyclic durations for these phases to address TOD patterns. In the sequel, fixed cycle time control, and related control schemes based on fixed cycle times, will be simply referred to as fixed time control as compared with cycle-free control schemes described below.

With the advent of traffic sensors, specifically inductive loop detectors in the pavement, traffic engineers have been able to implement “*actuated*” traffic signal systems where some phase durations are time “*extended*” or are ended early (often referred to as “*forced off*” early), depending on the traffic detected just upstream of the signals [35][36]. Actuated phases with TOD signal timings have been observed to somewhat improve the performance of non-actuated TOD timings when the traffic patterns have more variance. Further use of available computational resources with a traffic control system have seen the introduction of *traffic-responsive signal control systems* e.g., SCOOT [22] where the phase timing plans are changed based on traffic patterns observed in the recent past, say in the last 15 mins.

Gartner and others [17][18][20] developed and tested the OPAC *traffic-adaptive control* scheme based on observed traffic but now the phase durations are *cycle free*. Here durations for each phase in sequence ABCD need not be equal and some phases may even be skipped if the corresponding traffic stream is absent or very low. The underlying engine algorithm for OPAC is dynamic programming [7]. In the last three decades, traffic signal schemes have been tested that have included prediction of traffic patterns based on observed past patterns and current upstream detections. Most successful are the ones that *proactively* set cycle-free signal phases, notably RHODES that was first proposed about 25 years ago, see e.g. [31] and recently SURTAC [42].

Researchers and practitioners have also addressed the problem of controlling traffic streams on a network of intersections where vehicle departures from one signal impact the arriving traffic at downstream intersections. Optimally “coordinating” or “synchronizing” phases has been a much-studied problem [11][25][26][30][34]. Widely researched optimization algorithms for network of intersections (a) for TOD fixed cycles include TRANSYT [40], UTCS [16], TRANSYT-7F [49] and PASSER [9], (b) for traffic responsive fixed cycles include SCOOT [22] and SCATS [41], and (c) for cycle free phasing include RHODES [31] and RT-TRACS [37], In all of the above network approaches, each set of intersection signals are controlled by a single controller and all controllers are coordinated by the plans received from the underlying algorithms. In the paper [43] Stevanovic et. al., compared the influence of SCOOT and SCATS signal timings, cycle length, and offsets, etc. on the traffic performance, using VISSIM microsimulation. Recently, Stevanovic et. al., in a comprehensive study [4] compared fixed time cyclic plans that were optimized by three methods: (1) based on Highway Capacity

Software [27], (2) Tru-Traffic, used by some practitioners [8] and (3) VISTRO [38], a genetic-optimization based method available within VISSIM. They evaluated these plans using a calibrated simulation model of a network within Fort Lauderdale, Florida, very much in the vein of this paper that addresses cycle-free plans for a DI in Phoenix, AZ

Also, related to our DP -based optimization discussed in this chapter, several offline value-based reinforcement learning techniques like Q-learning, etc. have been studied to improve the coordination of intersection movements [5][6][13].

In many situations, a single controller directly controls a pair of closely spaced set of signals which normally is cheaper to install than two separate controllers. Developing a signal timing plan, for two closely spaced intersections, for example for a diamond interchange, is often a challenge for traffic engineers. The number of non-conflicting movements for the combined two intersections are significantly larger and so are the corresponding phases. Frequently, for practicality, diamond interchanges are controlled using TOD fixed time cyclic “3-phase” or “4-phase” signal control plans (these are described shortly in the next section).

Several signal timing strategies have been developed for diamond interchanges per se. PASSER III [14] model is one of the earliest works for optimizing fixed signal control at diamond interchanges. This approach is an extension to PASSER, which determines fixed time signal plan based on data collected offline. Later Tian and co-researchers [44][45][46][52] developed signal control approaches that include both the traffic signals and ramp meters, still with fixed time using data collected offline. Messer and Chang [28][29] studied traffic actuated diamond interchanges and showed improved traffic

performance. Mirchandani and co-researchers [21], were among the first to develop and implement proactive traffic control for diamond interchanges based on the RHODES logic. Conducted field tests showed a significant performance improvement compared to pre-timed signal control strategies. Later Fang and Elefteriadou [15] reported a similar DP-approach to adaptively control a diamond interchange; but their solution approach was an approximation and was also not field evaluated.

In this chapter, a dynamic programming based proactive, cycle-free traffic control model in a connected vehicle (CV) environment has been developed. Basically, in this scenario, a percentage of vehicles are assumed to be connected to the signal infrastructure through a communication mechanism, using wireless telephony like 4G/5G-LTE [1][12] or radio such as DSRC [55], so that exact locations of these vehicles is always known.

The proposed traffic control approach, which we refer to as MIDAS (Managing Interacting Demands And Supplies) since it is based on the MIDAS Cyber-physical System (CPS) architecture developed by Mirchandani and co-researchers [32][33]. This work also compares MIDAS with RHODES and an optimal fixed time signal control (OFTC) and shows that the MIDAS indeed performs significantly better than the other two approaches on a real simulation of a diamond interchange in Phoenix, Arizona.

1.2 Controlling a Diamond Interchange

A diamond interchange (DI) consists of two closely spaced intersections with complicated traffic movements. DIs are equipped with a traffic controller that controls two sets of traffic lights for the two intersections that are within the diamond interchange. The phases at each of the intersections need to be synchronized to accommodate heavy traffic movements on

off ramps and arterial streets. A typical DI is shown in Figure 1.1, with 8 possible traffic movements to accommodate on-ramp, off-ramp, and arterial street traffic streams. Since it is assumed that right turn on red is allowed, right turn arrows are left out.

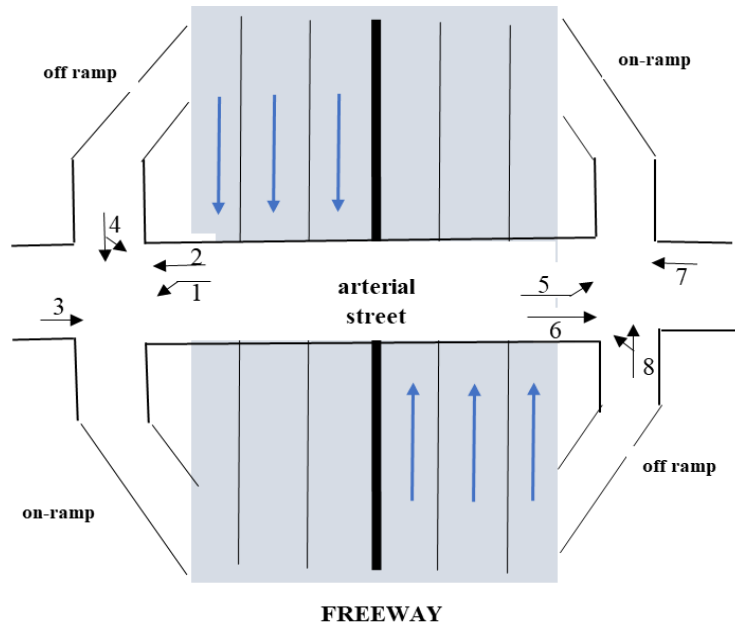




Figure 1.1. Layout of a Diamond Interchange with Freeway Underpass

Some common notations used in this chapter related to diamond interchange are defined in Table 1.1.

Table 1.1. Diamond Interchange Notation to Describe Signal Timings

Terminology	Definition
Movement (→)	Any given direction of traffic flow at an intersection is defined as a movement, i.e., right, through, and left movements, which are denoted by arrows.

Phase (Φ_i)	Any phase Φ_i is defined as set of non-conflicting movements. All movements in a phase are set to green when that phase is operating. Figure 1.2 shows an example where phase Φ_3 is green.
Signal Stage	A stage is defined as a set of coordinating phases that are set to green. For example, Φ_1 & Φ_5 is a signal stage when Φ_1 and Φ_5 are operating. This terminology is in consistent with the stage in the DP that will be developed below. A stage can be a single phase or set of phases as in Figure 1.3 where signal stage Φ_3 & Φ_5 is green.
(+)	To make it easy to understand the traffic movements participating in a phase, often in this document a movement is labeled with a '+' notation to indicate its participation in more than one phase. For example, in Figure 1.2 through movement denoted by $\Phi_1 + \Phi_2$ means that this movement is green during both Φ_1 and Φ_2 phases
Cycle length	Cycle length is defined as time required for a controller to cycle through all the defined signal stages of the intersection.
Split	Split is the portion of time allocated to a signal stage in a cycle. This is terminology applies to fixed timings with fixed cycles and splits
Offset	Used in describing signal coordination on arterials using fixed signal cycles. Offset is the time difference between the start of cycle of two neighboring intersections and either expressed in seconds or percentage of cycle length.
Lead-Lead	Terminology used by practicing traffic engineers. In this setting, both arterial left-turn phases turn green before the coordinated through movements

Lag-Lag	Terminology used by practicing traffic engineers. Here, both arterial left-turn phases start after the coordinated through movements end (at the same time)
Lead-Lag/Lag-Lead	Terminology used by practicing traffic engineers. Here, left turn on one direction starts before through movement and left turn on the other direction starts after the end of its coordinated through movement
	A movement or phase is said to be protected if it has the right of way, during a signal stage. Often denoted by solid arrows in movement diagrams
	A movement is said to be permissive when it yields to oncoming conflicting traffic movements during a signal phase. Often denoted by dashed arrows

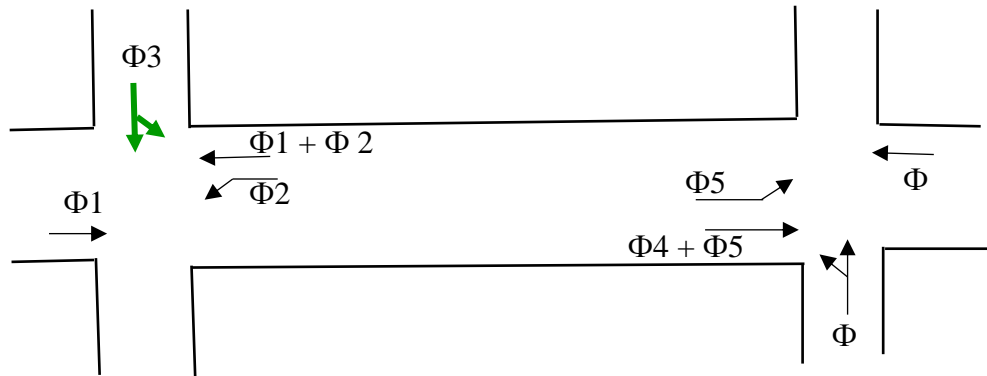


Figure 1.2. Phase $\Phi 3$ Example Shown with Movements in Bold Green.

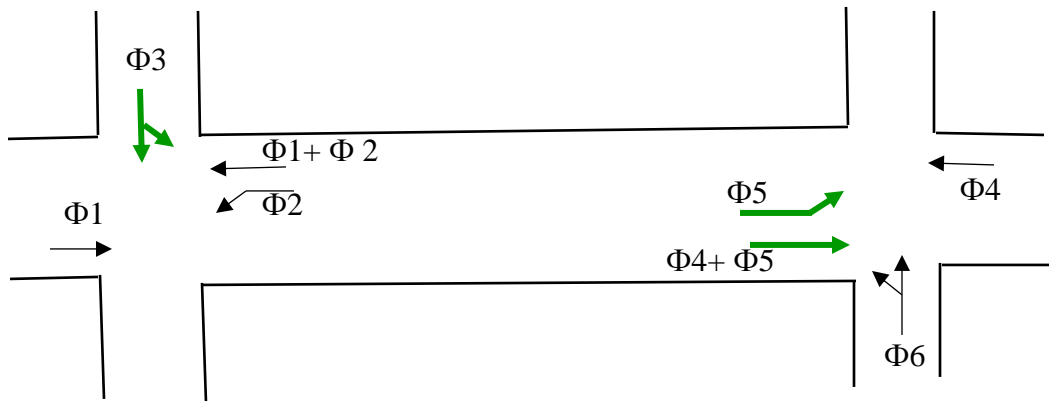


Figure 1.3. Signal Stage Example Shown with Concurrent Movements in Bold Green.

1.2.1 Some Signal Plans at Diamond Interchanges used by Traffic Engineers

Below we illustrate some common signal plans used at DIs.

a) 3-Phase LAG-LAG:

This signal plan shown in Figures A.1 & A.2 and Table A.1 has east-west through phase Φ_1 , arterial left turns phase Φ_2 , and off-ramp phase movements Φ_3 . Traffic engineers have designed this plan to serve heavy through-traffic on the arterial but will not service well heavy ramp traffic volumes or heavy crossroad left turn volumes, as these movements are affected by the limited internal storage capacity on the arterial between the intersections. This plan is good for two-way progression on the arterial when the interchange is very wide with zero-offset and for traffic demands that are directionally balanced and not too heavy. This is also a good plan for short cycle lengths.

b) 4-Phase LEAD/LAG - LAG:

This plan (see Figures A.3 and A.4 and Table A.2) is designed to serve heavy through traffic on the crossroad and heavy ramp traffic (on both ramps) but will not service well heavy arterial left turn volumes. Again, the arterial left turn volumes are limited by the internal storage capacity. This plan also favors two-way progression on arterial and short cycle lengths.

c) 4-Phase LEAD/LAG:

This phase plan (see Figures A.5 and A.6 and Table A.3) is designed to serve heavy through traffic, heavy ramp traffic (on both ramps) and a heavy arterial left turn demand in one direction. The left turn in the opposite direction will store vehicles between the intersections. This plan also has the potential for good two-way progression for conditions previously described in 3-phase LAG-LAG, but this plan favors crossroad left movements more and reduce the green-time (split) for the opposing through movement.

d) 3-Phase WITH "OVERLAPS":

This phase (see Figures A.7 and A.8 and Table A.4) is very similar to the 3-phase lag-lag pattern, with the added ability to service one heavy ramp movement and one heavy crossroad left turn movement at any given time. This phase also can focus on ramp or crossroad left turn movement that may be favored based on time of day. Figure A.8 explains possible signal sequences of this plan, where the box with heavy solid lines indicates the primary sequence of signal stages and dotted lined box indicates possible overlap stages when needed. Basically Figure A.8

indicates 3 primary signal stages and two additional stages, each of which has two possible overlap stages.

e) 4-Phase WITH "OVERLAPS" (TTI) PHASING:

This phase plan (see Figures 1.4.1 and 1.4.2 and Table 1.2), proposed by Texas Transportation Institute [56][47][48], aims to serve heavy traffic on all four external approaches including the crossroad left turn movements. This is achieved by serving one approach at a time; for example, once a through movement is initiated at one intersection, it clears the traffic through the other intersection without the vehicles having to make a stop at the internal storage. This phase plan has some shortcomings associated with the concept of serving one approach at a time. Some added efficiency may be achieved through the early release of a through movements to consider the internal storage capacity utilization. Figure 1.4.2 explains the signal sequence of this plan, where solid arrows denote primary signal stages and dotted arrows denote overlap stages. This phase plan could potentially use 6 stages as shown in Figure 1.4.2 and Table 1.2.

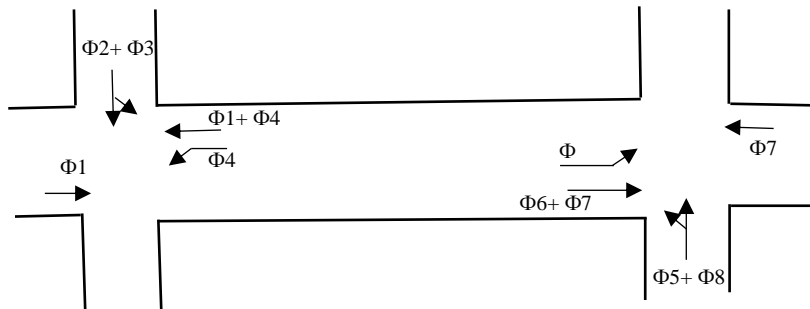


Figure 1.4.1. "4-Phase with Overlaps" Phase Movement

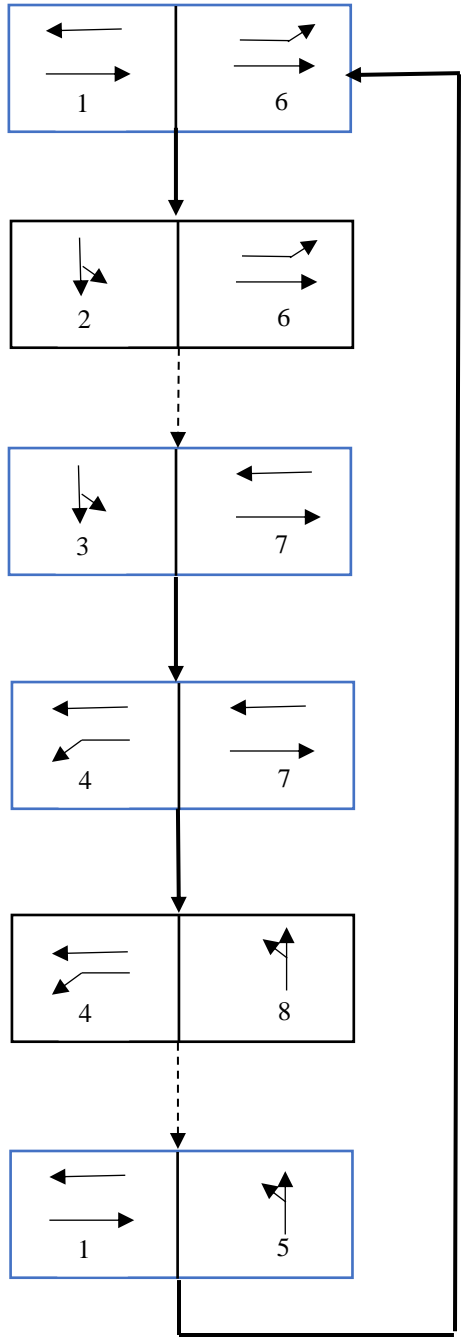














Figure 1.4.2. “4-Phase with Overlaps” Signal Scheme

Table 1.2. “4-Phase with Overlaps” Signal Stages

Stage	Green Phases
1	$\Phi 1$  $\Phi 6$ 
2	$\Phi 2$  $\Phi 6$ 
3	$\Phi 3$  $\Phi 7$ 
4	$\Phi 4$  $\Phi 7$ 
5	$\Phi 4$  $\Phi 8$ 
6	$\Phi 1$  $\Phi 5$ 

1.2.2 MIDAS Signal Stages at Diamond Intersections

The fixed signal plans discussed above address specific demand scenarios at the diamond interchanges, and none of these policies address managing stochastic demand. Demand fluctuations at the interchange corridor are inevitable and unpredictable. Implementing signal policies-based TOD scenarios leads to poor throughput and congestion management at the corridor. The proposed MIDAS concept employs flexible phases to proactively control all traffic movements as per demand fluctuations observed through sensors. The 6 different signal phases shown in Figure 1.5.1 and the 9 different signal stages shown in

Table 1.3 serve all possible non-conflicting movements at the interchange. These 9 stages include all possible phase combinations used in the 3-phase and 4-phase conventional plans discussed above. Figure 1.5.2 previews the DP decision flow diagram, which will be described in detail in the next section.

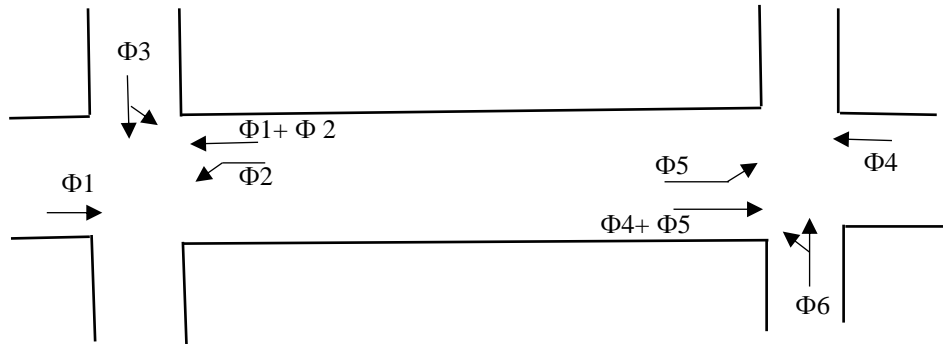


Figure 1.5.1. MIDAS Phase Movement

Phase durations recommended to signal control infrastructure.

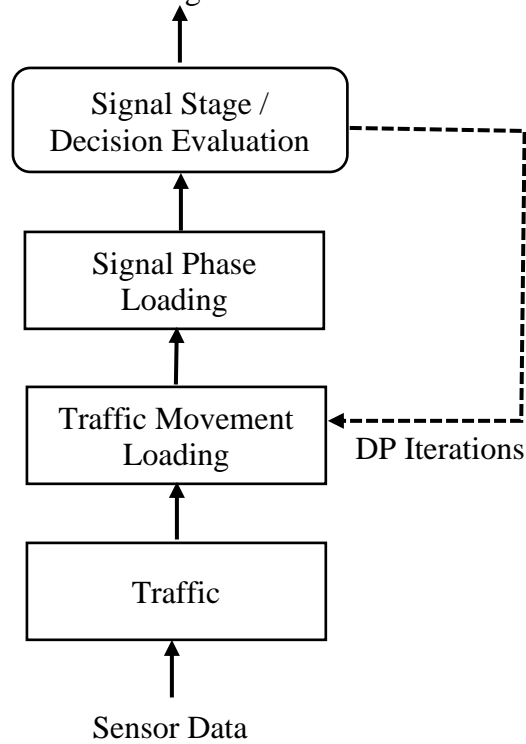




















Figure 1.5.2. MIDAS DP Decision Process.

Table 1.3. MIDAS Signal Stages

Stage	Green Phases
1	$\Phi 3$  $\Phi 6$ 
2	$\Phi 1$  $\Phi 4$ 
3	$\Phi 2$  $\Phi 5$ 
4	$\Phi 3$  $\Phi 5$ 
5	$\Phi 2$  $\Phi 6$ 
6	$\Phi 3$  $\Phi 4$ 
7	$\Phi 1$  $\Phi 6$ 
8	$\Phi 1$  $\Phi 5$ 
9	$\Phi 2$  $\Phi 4$ 

1.3 MIDAS Proactive Traffic Control Strategy

As described in the introduction, there are several traffic signal control and strategies, some implemented and deployed, some prototyped and some being developed and tested, such as MIDAS. A good, fixed time strategy, defined later as a baseline, is OFTC (optimal fixed time signal control). A good proactive cycle-free control strategy that has been deployed and field tested is RHODES [19]. It uses upstream fixed detectors, such as inductive loops and traffic cameras, to predict vehicle arrivals and to set phase durations at intersections being controlled cycle-free. In this chapter, the performance of MIDAS, cycle-free signal control strategy, is evaluated and compared to OFTC and RHODES.

1.3.1 Optimal Fixed Time Signal Control (OFTC)

To evaluate the expected performance of any proactive traffic control algorithm via simulation modeling, it is useful to compare the performance metrics of interest to driving public against a baseline that represents a predominant strategy that traffic engineers use: optimal fixed cycle signal timings. To this end, we used VISSIM's [39] signal control optimization procedure (described below); we will refer to it as "Optimal Fixed Time Signal Control" (OFTC) VISSIM simulation software performs green time optimization for stage-based fixed time controllers, where the sequence of stages is given. For each traffic load it performs numerous simulations of the entire network and finds the green times that minimizes total delay for that load through a search approach discussed below; the total green time for the stages gives the optimal fixed cycle time. For our OFTC plan for DI we simulated and used the stages of the 4-phase (TTI) plan since this TTI plan is suitable for DIs with heavy traffic loads on all approaches. We note similar comparisons can be used for any plan provided by the traffic engineer.

VISSIM's Signal Optimization Procedure

Starting with some initial green times, cycle time, and engineer-specified minimum green times, VISSIM, through simulations, first determines the average delay of all vehicles that have passed through the interchange on the lanes of the designated movements, using an automatically created evaluation procedure for each movement over the entire simulation run. Then the following steps are then performed to determine optimal fixed time signal plan.

1. The signal phase in which the vehicles have the highest average delay is determined for each stage.
2. The stage with the lowest maximum average delay is selected as the *best stage*.
3. The stage with the highest maximum average delay is selected as the *worst stage*.
4. One second of green time is deducted from the current best stage and one second of green time is added to the current worst stage.
5. If a second can no longer be deducted from the best stage, the second-best stage is used. If this stage cannot be shortened because a specified minimum green time is reached, then the next best stage is considered, this is considered iteratively. Until no other stage can be shortened by one second, the optimization is terminated.
6. The following rules are used to compare fixed time signal timing plans to proceed with the search for finding the optimal plan. A signal plan is better than another if one of the following criteria is met:
 - a. If the total vehicle flow through the intersection during the simulation run has increased by at least 25 vehicles, or by 10% if it is less than 25.

- b. If the flow has not significantly decreased by 25 vehicles or by 10% but the average delay across all vehicles has decreased.
- 7. If a signal program is better than the best found so far then it replaces this as the best.
- 8. The optimization terminates with a fixed time signal timing plan is declared optimal when one of the following criteria is met:
 - a. The identified best signal timing plan does not change in 10 simulation runs.
 - b. The flow decreases by more than 25% compared to the identified best signal plan.
 - c. The average delay increases by more than 25% compared to the identified best signal plan.

More details are given in PTV-VISSIM, [39]

1.3.2 MIDAS Underlying Architecture and Algorithms

In any proactive traffic control system, the signal control algorithm is the key component in determining the optimal phase sequence and phase durations, by minimizing some user defined traffic performance measure, such as total delays, stops or queues at the intersections. Many researchers have worked towards developing proactive traffic control systems in the past, of which RHODES is among the better performing real time proactive traffic control systems that has been field tested. But like existing reactive and adaptive traffic control systems, RHODES uses measurement data from fixed sensors such as loop detectors and fixed cameras that count vehicles, referred to as Eulerian data, for prediction and estimation of vehicle arrivals at intersections. On the other hand, MIDAS traffic control uses trajectory data from GPS devices on vehicles, referred to as Lagrangian data, along

with Eulerian data from upstream detectors at all the approaches to predict individual arrival times of approaching vehicles, as well as turn movements at the intersections. These predictions are dynamically updated for a user defined time horizon. MIDAS traffic control algorithm uses these predictions to estimate the queues that are going to be formed, during the time horizon for each approach lane at the intersection, and the associated stopped delays.

Like RHODES, MIDAS control algorithm predicts approaching traffic and estimates queues at the intersection based on the estimated or user defined queue discharge rates. From the navigation systems' GPS data of some vehicles that are using the intersection, MIDAS knows the O-D paths of the approaching traffic, so MIDAS knows exact turning movement of these vehicles. Also, GPS data can be used in estimating travel times on lanes, so that MIDAS can predict arrival time at the back of queue for each individual vehicle. Since this work deals with mixed types of vehicles, vehicles enabled with and without GPS tracking, we estimate the link travel times using both Eulerian and Lagrangian measurements as described in the next subsection.

1.3.2.1 Estimation and Prediction of Vehicle Arrivals at Intersections

Assuming, without loss of generality (WLOG), that a certain proportion of participating traffic is enabled with GPS devices, providing location and speed of the vehicle in real-time along with its origin and destination information. The following estimation approach uses both GPS data and loop detector data together to predict arrivals, turn movements and estimated queue lengths at the interchange (from here on model developments apply to either intersections or interchanges).

Notation: -

N_t = number of vehicles with GPS enabled, reaching the intersection in the time interval $(t - 1, t)$.

N'_t = number of vehicles without GPS enabled, reaching the intersection in the time interval $(t - 1, t)$.

$T_{(i,t)}^u$ = time of arrival of GPS enabled vehicle i at the upstream detector and reaching downstream detector during the time interval $(t - 1, t)$.

$T_{(i,t)}^d$ = time of arrival of GPS enabled vehicle i at the downstream detector during the time interval $(t - 1, t)$.

Δx = distance between upstream and downstream detector.

v_t^u = average speed registered at upstream detector at time t .

v_t^d = average speed registered at downstream detector at time t .

Travel time of vehicles from upstream to downstream, without GPS during the time interval $(t, t+1)$ can be estimated according to Chen, C. [10] as shown in (1.1a)

$$\theta_{t+1} = \frac{1}{2} \cdot \left(\frac{\Delta x}{v_t^u} + \frac{\Delta x}{v_t^d} \right) \quad (1.1a)$$

Now the link travel time of any vehicle in the interval $(t, t+1)$ can be estimated as shown in (1.1b)

$$\tau_{t+1} = \frac{\sum_{i=1}^{N_t} (T_{(i,t)}^d - T_{(i,t)}^u) + N_t' \cdot \theta_{t+1}}{N_t + N_t'} \quad (1.1b)$$

Turning probabilities at the intersection are estimated using GPS data, shown in equations (1.2a), (1.2b) & (1.2c)

$$\text{probability of vehicle turning left, } \rho_{t+1}^l = \frac{N_t^l}{N_t} \quad (1.2a)$$

$$\text{probability of vehicle going through, } \rho_{t+1}^s = \frac{N_t^s}{N_t} \quad (1.2b)$$

$$\text{probability of vehicle turning right, } \rho_{t+1}^r = \frac{N_t^r}{N_t} \quad (1.2c)$$

Estimating queues at the intersection is essential for efficient proactive control and discharge of vehicles. MIDAS in real-time estimates the queue lengths on each lane over a rolling horizon using the following equation. MIDAS in real time updates the data structures containing the predicted arrivals and committed queues every simulation second. As soon as the predicted vehicle arrives at the intersection it gets committed. At any given time, MIDAS estimates the anticipated queues for the following time horizon using the committed queues, predicted arrivals and departures as shown in Figure 1.6 and eqⁿ. (1.3).

The queue estimation process logic sequence is illustrated in Figure 1.7

$$Q_{T_n} = Q_{T_{n-1}} + A_{(T_n, T_{n-1})} - D_{(T_n, T_{n-1})} \quad (1.3)$$

Q_{T_n} = estimated queue length at the end of time horizon T_n .

$Q_{T_{n-1}}$ = leftover or committed queue at the beginning of time horizon T_n .

$A_{(T_n, T_{n-1})}$ = predicted arrivals during time horizon T_n .

$D_{(T_n, T_{n-1})}$ = observed or estimated departures during time horizon T_n .

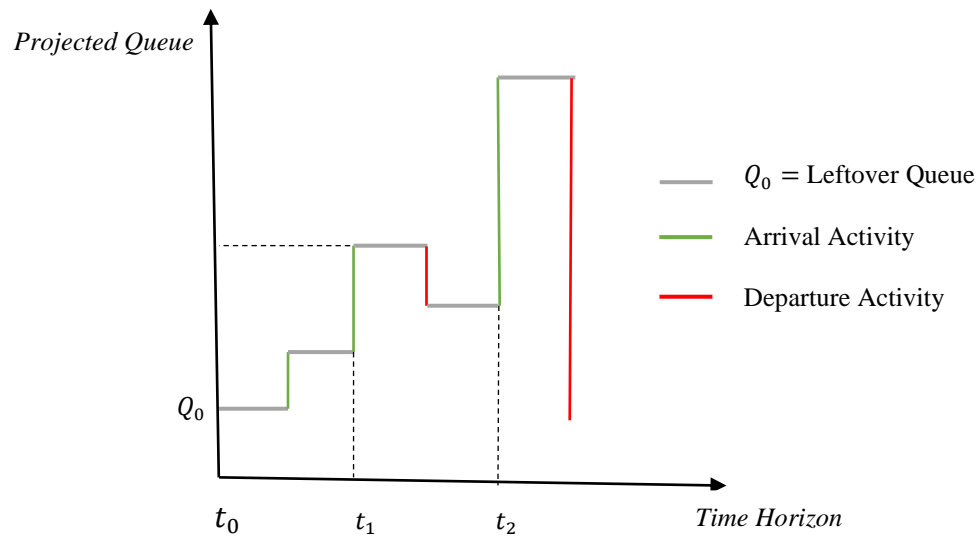


Figure 1.6. Projection of Queue Over a Time Horizon

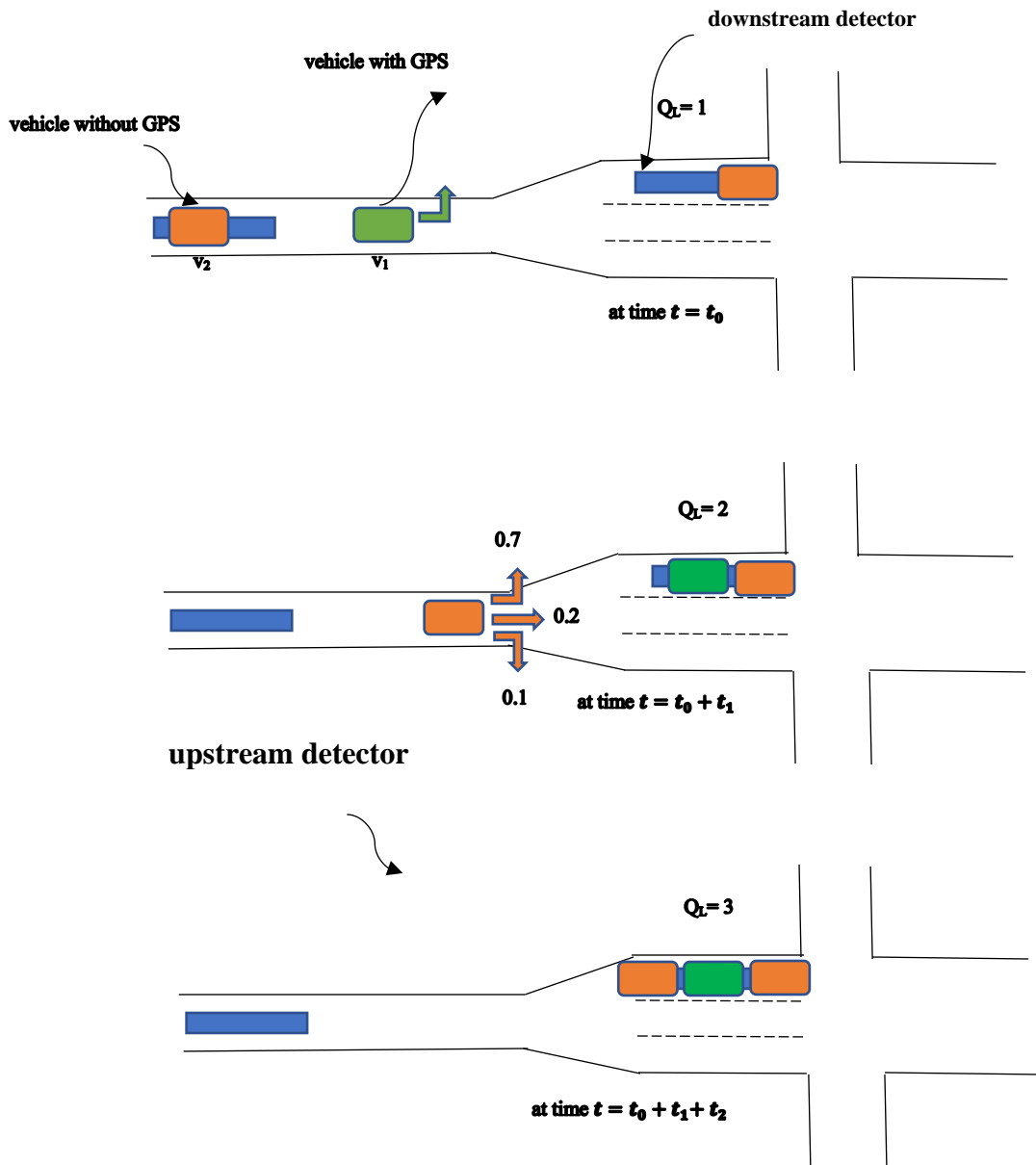


Figure 1.7. Sequence (Top to Bottom) of Snapshots for Queue Estimation Logic

1.3.2.2 MIDAS Control Optimization Problem

Signal control problem can be mathematically formulated similar to machine scheduling problem, by treating traffic signals at the intersection as a machine and approaching vehicles as jobs, using mixed integer programming (MIP) as follows.

Parameters

J = set of possible movements present at the intersection.

a_{ij} = arrival time of vehicle i in movement j at the back of the queue, at the intersection.

d_{ij} = departure time needed for vehicle i in the movement j to clear the intersection.

f_{ij} = free flow time needed for vehicle i in the movement j to clear the intersection.

V_j = set of vehicles in the movement j .

C_j = set of movements in conflict with movement j .

Decision Variables

r_{ij} = scheduled release time of vehicle i of movement j , through the intersection.

Objective

$$\min \sum_{i \in V_j} \sum_{\forall j} (r_{ij} + d_{ij}) - (a_{ij} + f_{ij})$$

Constraints

$$r_{ij} \geq a_{ij} \quad \forall i \in V_j, \forall j \in J \quad (1.4)$$

$$r_{i+1j} \geq r_{ij} \quad \forall i \in V_j, \forall j \in J \quad (1.5)$$

$$r_{i'j'} < r_{ij} + M \cdot z_{i'j'ij} \quad \forall i' \in V_{j'}, \forall j' \in C_j, \forall i \in V_j, \forall j \in J \quad (1.6)$$

$$r_{i'j'} \geq r_{ij} + d_{ij} - M \cdot (1 - z_{i'j'ij}) \quad \forall i' \in V_{j'}, \forall j' \in C_j, \forall i \in V_j, \forall j \in J \quad (1.7)$$

$$r_{ij} \in I, \quad z \in B, \quad M \text{ is a large integer} \quad (1.8)$$

The objective function above minimizes the waiting time of vehicles at the intersection. The first part of the objective function is the time at which vehicle i leaves intersection through movement j and the second part is the time it would have left the intersection if there was no signal control at the intersection or free flow time. Constraint (1.4) enforces that the release time of the vehicle from its corresponding queue should be at least equal to the arrival time of the vehicle at the intersection (or back of the queue). Constraint (1.5) in the formulation makes sure that platoons are discharged from the queue in FIFO order. Constraints (1.6) and (1.7) eliminate scenarios where conflicting traffic movements happen at the same time. Index i' is for vehicles in conflicting movement j' .

Using conventional solvers to solve this MIP becomes computationally tedious when dealing with a large networks and large numbers of vehicles and hence it will be difficult to obtain a solution in real-time. As shown below, MIDAS uses a DP approach to

significantly speed up the computation at the expense of memory. MIDAS takes advantage of the fact that DP solves the sub-problems sequentially and stores results for later use, this technique reduces the time complexity from exponential to polynomial.

1.3.2.3 Dynamic Programming Approach for Control Optimization

MIDAS signal control algorithm uses DP forward recursion to minimize a user-defined performance measure over a finite-time horizon (that rolls forward from run to run) and, then uses a backward recursion to retrieve the optimal phase schedule for that time horizon. The signal control algorithm of MIDAS is a modification of DP algorithm developed in RHODES [31] but with a more efficient data structure. MIDAS signal control algorithm determines the optimal signal phase sequence and duration of each phase in the sequence, by taking signal phases (refer to Figure 1.5.1 and Table 1.3) and time horizon T as input parameters. Control algorithm runs DP at some time stamp 't', with prescribed time horizon T , considering the currently estimated arrivals on all approaches of the interchange over the timeline $t+T$. The DP is formulated such that each "stage" of the DP is associated with a signal stage (Table 1.3) and number of time units allocated after completion of a particular stage is defined as DP's "state variable". The DP solution consists of the signal stage sequence and time units allocated to each signal stage over the time horizon of T time units as illustrated in Figure 1.8. At every DP run, the sequence of signal stages begins with the current signal stage green phases at the interchange, which allow for the phases of current signal stage to be terminated or extended based on the updated observations.

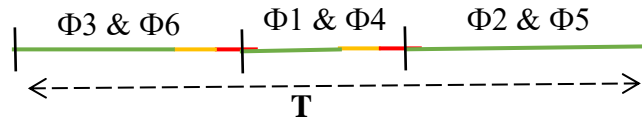


Figure 1.8. DP solution representation: an example of optimal phases and their durations over time horizon T

DP notation: -

O = unique set of phases, combination of non – conflicting vehicle movements.

T = length of time horizon considered.

g_{min} = min green time required for a phase.

g_{max} = max green time required for a phase.

r = red light time or clearance time required before phase change.

i = index of the stage in DP

s_i = state variable denoting total time allocated to phases until the completion of stage i

c_i = control decision variable denoting number of time units allocated to phase i

$C_i(s_i)$ = set of all feasible control decision values of c_i , given state s_i .

$f_i(s_i, c_i)$ = performance measure in stage i , given state s_i and control decision variable c_i .

$z_i(s_i) =$ cumulative function that stores the sum of performance measures of all stages until stage i given the state s_i .

$$C_i(s_i) = \{0,1,2 \dots T\}$$

$$s_i = \begin{cases} s_{i-1}, & \text{if } c_i = 0 \\ s_{i-1} + c_i + r, & \text{otherwise} \end{cases} \quad (1.9)$$

The performance index function f_i is generalized to take care variety of key performance metrics at the intersection like delays, stops and queue lengths, etc. WLOG, we can assume that $f_i(s_i, c_i) > 0$ for a given feasible state s_i defined according to eqn. (1.9) and control decision c_i . In this chapter, for explanation purpose, we assume the performance index is delay, in other words, the waiting times at the intersection. The cumulative delay is the minimization function that minimizes the total estimated delay up to the end of the decision horizon for the control decision of the allocated time units to signal stages. This objective can be expressed mathematically as (1.10):

$$z_i(s_i) = \min_{c_i} f_i(s_i, c_i) + z_{i-1}(s_{i-1}) \quad | \forall c_i \in C_i(s_i) \quad (1.10)$$

1.3.2.4 Dealing with Limited Internal Storage Capacity at the Interchange

One of the main challenges in setting optimal phase decisions for a diamond interchange is to avoid spill backs due to limited inter storage capacity between the closely spaced intersections. To address this challenge, MIDAS uses increasing weighted expected delays on lanes with limited internal storage capacity to minimize overspilling (we used exponentially increasing weights but could be replaced by any increasing function). Weight

used for the internal storage lane l at any time t is an exponential function of queue length and is computed by. (1.11).

$$W(l, t) = \begin{cases} e^{k_l \cdot Q_l^2(t)}, & \text{if } l \text{ is the internal storage lane.} \\ 1, & \text{otherwise} \end{cases} \quad (1.11)$$

$W(l, t)$ = weight of internal storage lane l at any time t .

l = internal storage lane l , between closely spaced intersections of the interchange.

$Q_l(t)$ = Queue length of internal storage lane l at time t .

k_l = a multiplicative constant for lane l (discussed below).

So, the performance index function can be re-written as shown is eqn. (1.12)

$$f_i(s_i, c_i) = \sum_{\forall l \in L_i} W(l, s_i) \cdot d_l(s_i, c_i) \quad (1.12)$$

L_i = set of lane movements possible during phase i

$d_l(s_i, c_i)$ = total delay of vehicles on lane l , given state s_i and control

decision c_i .

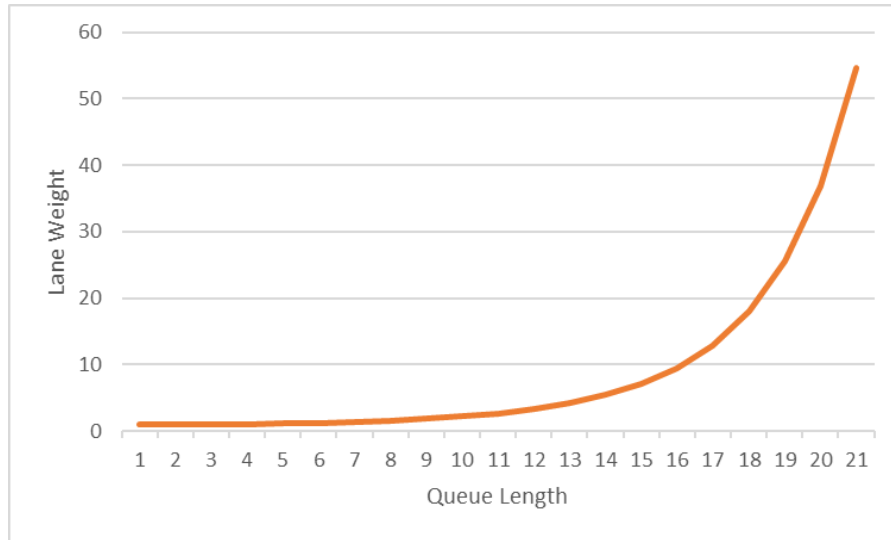


Figure 1.9. The lane weight vs queue length for I-17/19th Ave., Phoenix, AZ interchange

Multiplicative constant k_l in the expression can be fine-tuned based on the internal storage lane capacity and geometry of the diamond interchange. In the example of I-17/19th Ave., Phoenix, AZ interchange k_l value was set to 0.01 for all internal storage lanes based on simulation analysis, and the lane weights varied with queue length as shown in Figure 1.9.

1.4 Evaluation and Results

To evaluate the efficiency of MIDAS control, a freeway diamond interchange (at I-17/19th AVE., Phoenix) was simulated using VISSIM, a microscopic multi-modal traffic flow simulation platform, as shown in Figure 1.10. Simulated network layout was calibrated with the help of Arizona Department of Transportation [2].

MIDAS control is designed to minimize any user defined objective/metric. In this evaluation we used the objective of minimizing total delay of vehicles that arrive at the interchange from the off-ramps and arterial streets. In the evaluation experiments, MIDAS algorithm controlled the simulated interchange in VISSIM, via the Microsoft COM

interface [54]. Vehicle delay was measured as the additional time required to pass through the controlled diamond interchange due to the waiting times in queues. The performance of MIDAS signal control was evaluated by running multiple simulation runs with various traffic loads. All these loads were kept low enough so that there was sufficient capacity for the traffic loads without excessive spillover and the results simply focus on the delays due to management of traffic signal controls. After the end of each simulation run VISSIM reports simulation statistics collected during the run. A set of key performance results for the network simulated (see Figure 1.10) and the DI are given in Table 1.4 and 1.5

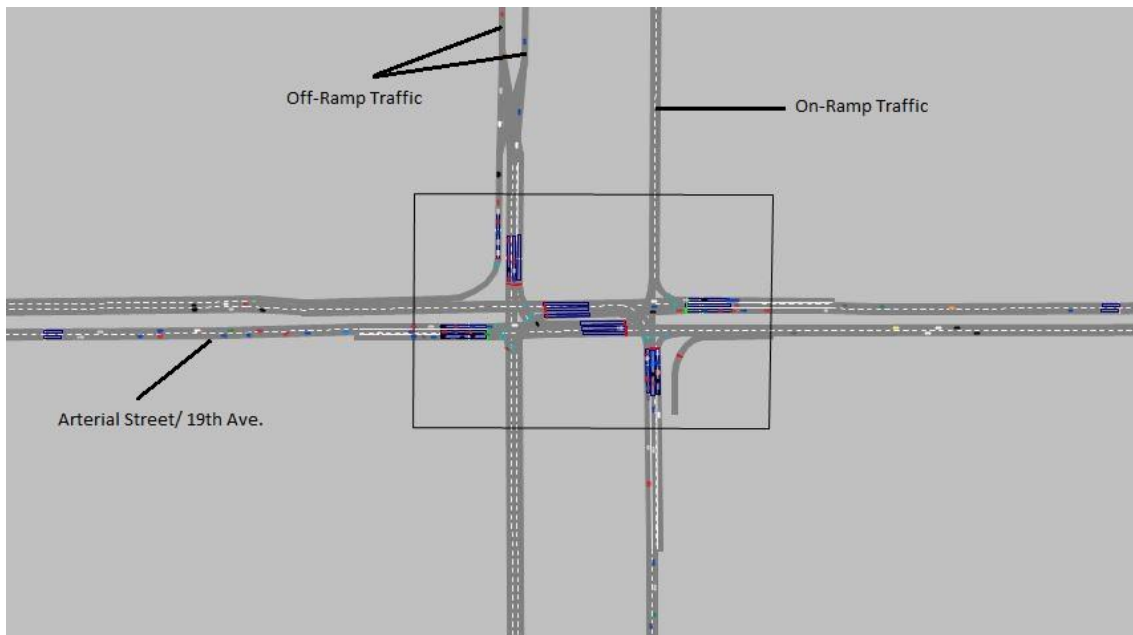


Figure 1.10. I-17/19th AVE., Phoenix Diamond Interchange in VISSIM

Table 1.4. Network Level Performance

Control	TrafficLoad(v/h)	DelayAvg(s)	TotalDelay(s)	TotalTravelTime(s)
MIDAS	4900	13.25	66235.87	414791
RHODES	4900	14.5	72472.17	421109
OFTC	4900	32.05	160218.47	467528

Table 1.5. Interchange Level Performance

Control	TrafficLoad(v/h)	DelayAvg(s)	StopsAvg	TotalStops	AvgQLEN
MIDAS	4900	11.5	0.68	3337	2.47
RHODES	4900	12.96	0.81	3964	2.75
OFTC	4900	25.74	1.01	5002	18.23

In the above tables, the performance metrics are.

- **DelayAvg:** Average of all vehicles delays due to waiting times in queues.
- **TotalDelay:** Sum of all vehicle delays in network, in seconds.
- **StopsAvg:** Average number of stops made by a vehicle at the interchange.
- **AvgQLEN:** Average queue length at the stop lines of the interchange.
- **TotalTravelTimes:** Sum of travel times of all vehicles in the network, in seconds.

In this simulation-based evaluation we assumed MIDAS to control at 100% market penetration of connected vehicles, meaning that every vehicle participating in the simulation is assumed to be equipped with GPS device and communicates with MIDAS

controllers. With GPS data MIDAS knows with certainty the turning movement of each vehicle at the interchange. Evaluations of MIDAS control with varying market penetrations are given in a later section.

We use the following notation for computing performance metrics. using eqn. 1.13.

N = total number of vehicles in simulation network

t_i^f = travel time of GPS enabled vehicle i from its origin to destination under free flow.

t_i^c = travel time of GPS enabled vehicle i from its origin to destination under controlled strategy.

s_i = number of stops made by GPS enabled vehicle i before reaching its destination

T = total simulation time in secs

Q_t^l = queue length of lane l at time t .

1.4.1 Average Delay

Average delay is a key performance metric used in traffic science to evaluate traffic control systems. Since many DIs in US implement pre-timed signal controls with fixed phase sequences like 3-phase or 4-phase plans, it would be legitimate to compare the average MIDAS' delays (eqⁿ. (1.13)) for the network shown in Figure 1.10 with optimal fixed time control for the DI. Figure 1.11 below compares the delays of MIDAS with RHODES and

OFTC for different traffic loads, using VISSIM simulations. MIDAS control shows a significant reduction in total delays when compared to the other two control strategies. RHODES, as shown before, has better delay times than OFTC at low loads, but it tends to increase at a faster rate than MIDAS, when the traffic load increases.

$$Average\ delay = \frac{\sum_{i=1}^N (t_i^c - t_i^f)}{N} \quad (1.13)$$

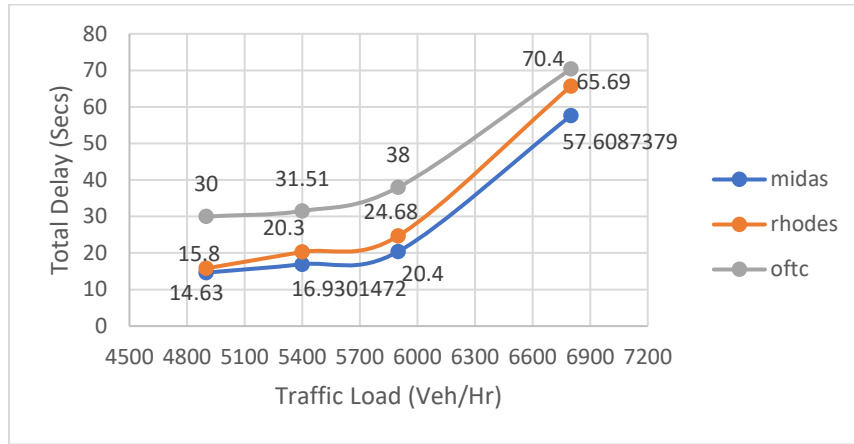


Figure 1.11 Comparison of Average. Delays vs Traffic Load for Different Signal Control Strategies

1.4.2 Average Stops

The average number of stops (eqⁿ. (1.14)) is another important performance metrics that traffic engineers are interested in decreasing, which is the average number of times a vehicle had to stop due to the traffic congestion and traffic signals in the network. Figure 1.12 compares stops due to MIDAS with other traffic control strategies for different traffic loads. Observe that MIDAS performs better with respect to stops when compared to RHODES and OFTC.

$$\text{Average stops} = \frac{\sum_{i=1}^N S_i}{N} \quad (1.14)$$

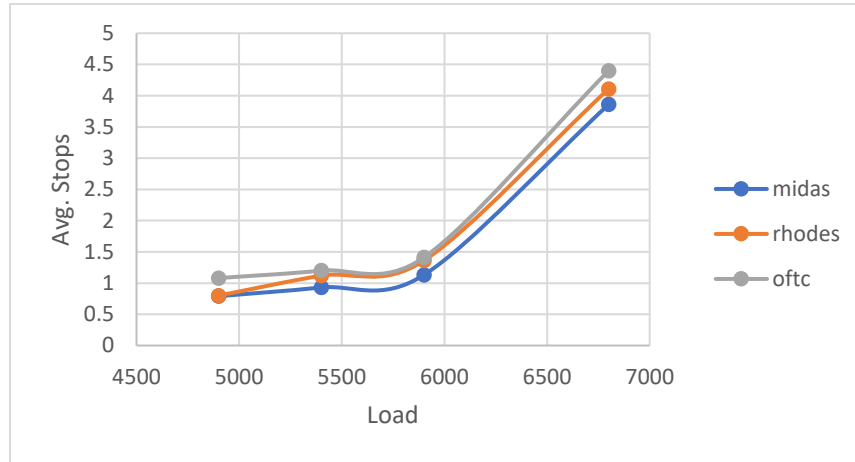


Figure 1.12. Comparison of Average Number of Stops vs Traffic Load for Different Signal Control Strategies.

1.4.3 Average Queue Length

Traffic engineers also try to come up with traffic policies to decrease the queues formed at the DIs due to limited capacity between the signals. Based on the VISSIM simulations for different traffic loads, a comparison of average queue lengths (eqⁿ. (1.15)) due to MIDAS, RHODES, and OFTC are shown in Figure 1.13. Again, MIDAS outperforms the other two traffic strategies. RHODES performed as well at low traffic loads, but queue lengths tend to increase at a higher rate than when the load increases.

$$\text{Average Qlength} = \frac{\sum_{t=1}^T \sum_{l \in L} Q_t^l}{|L|.T} \quad (1.15)$$

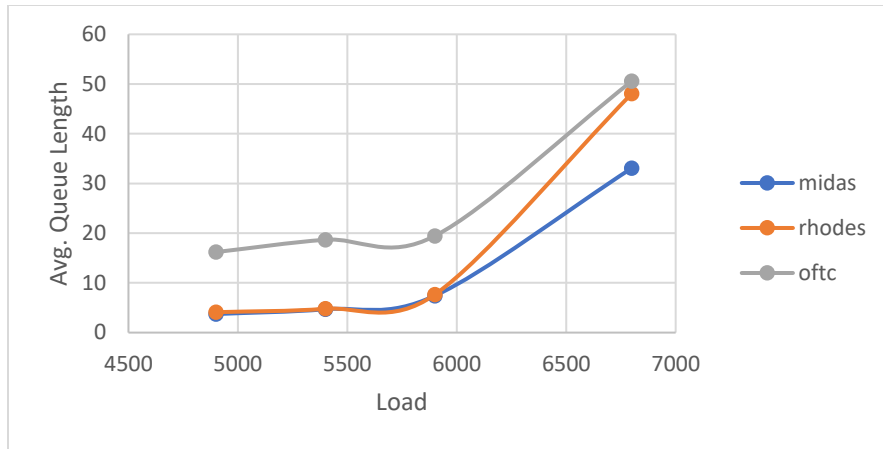


Figure 1.13. Comparison of Average Queue Length vs Traffic Load for Different Signal Control Policies

1.4.4 Market Penetration

We evaluated the performance of MIDAS for different market penetration rates of the connected vehicles, which is described as the percentage of vehicles that are GPS enabled and provide their routes to the MIDAS system. Our study shows that MIDAS delays decrease when market penetration increases, as would be expected (see Figure 1.14). Also, as would be expected, when market penetration is very low then these benefits are similar to RHODES (compare Figure 1.14 and Figure 1.11 for load 6800 v/h).

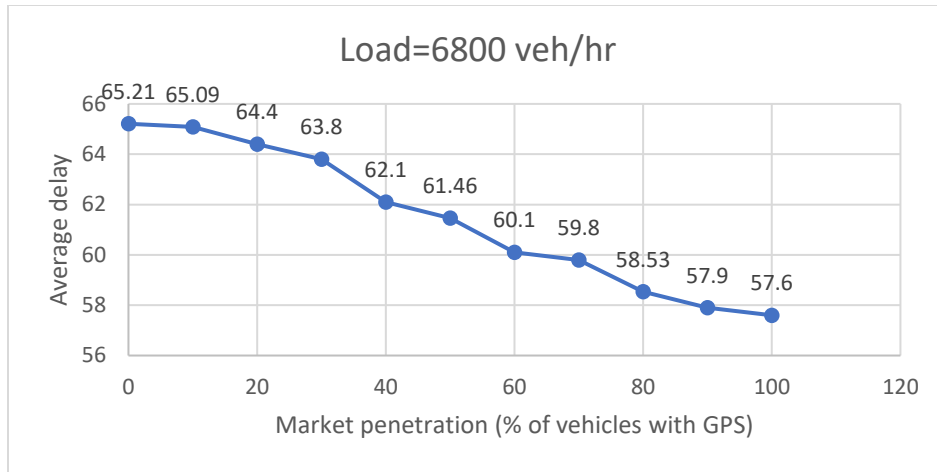


Figure 1.14. MIDAS Performance Graph for Different Market Penetration Rates for Traffic Load 6800 v/h

CHAPTER 2

PROACTIVE TRAFFIC CONTROL AND PLATOON MANAGEMENT OF AUTONOMOUS VEHICLES THROUGH AN UNSIGNALIZED INTERSECTION

Preview of Contributions

- a) Proposed an unsignalized intersection control logic for fully autonomous vehicles without a need for traffic signal lights.
- b) Proposed an effective platoon management logic for AVs.
- c) Proposed an efficient dynamic programming algorithm that determines the optimal discharge sequence of platoons/ AVs through an unsignalized intersection.
- d) Demonstrated the benefits of platooning AVs using CACC through simulation study.

2.1 Introduction to Autonomous Intersection Control and Management

Artificial intelligence (AI) has been shaping the future of numerous industries like aviation, healthcare, manufacturing, and transportation. Transportation problems are inherently complex problems to solve as the system behavior is complicated to model according to a predictable pattern. The complexity of the system arises from underlying uncertainties like traffic demand, capacity, human errors, and accidents, controlling different driver agents with conflicting objectives. In such an unpredictable environment, AI helps to predict appropriate actions or decisions using observed data, predictive and prescriptive models to promote safe, efficient, and reliable transportation. Among the plethora of intelligent automotive technologies developed so far, AI implementation in today's cars is notably preeminent. Today's cars are equipped with AI-driven smart technologies like driver

assistance system, collision avoidance, high-tech cruise controls, and night vision enhancements, etc.

Soon, traffic on the roads will partially become fully autonomous and it's inevitable. Before we get into the literature of autonomous intersection control and management it is important to understand different driving technologies associated with autonomous driving.

Autonomous driving autonomy is a state in which an agent acts independently making self-decisions. An Autonomous vehicle can make driving decisions independently to maximize its own objective. Autonomous driving itself doesn't improve the traffic congestion intrinsically until it adopts the cooperative driving strategy.

Automated driving enables automatic driving capabilities without human intervention. According to SAE there are 6 different levels of automation and are defined based on the automation features executed by the automation system.

Level 0 doesn't exhibit any automation capabilities. Human acts as the master of the control, with the human driver responsible for all aspects of driving like steering, pedaling, monitoring surrounding, navigating, and maneuvering through the traffic, etc.

Level 1 vehicles have automatic throttling and braking system. Some cars at this level obtain information about the environment, provide lane assistance and self-parking. But the human driver always has the main control of the vehicle.

Level 2 vehicles can handle steering, throttling, braking, and automatic lane changing capabilities. It includes driver assistance technologies with adaptive cruise control models using LIDAR or RADAR.

Level 3 automation responds in accordance with the surrounding environments, changes lanes and provides conditional assistance without human intervention. But the human driver still is required to take control when necessary.

Level 4 automation enables cars to drive by themselves without the need for a human to intervene. Although a human driver is present, his actions remain passive with the vehicle able to take over complete control.

Level 5 automated vehicles don't require human control at all and are fully automated without the need for pedals or a steering wheel and only contains passenger seating.

Connected driving involves information exchange between automated vehicles, non-automated vehicles, and infrastructure installed on the road network. Although connected technology provides the potential opportunity to improve the traffic situation, the information can be misused for individual objectives.

Cooperative driving lets the individual vehicles cooperate with each other by sharing information through Inter-vehicular communication (IVC) and employ microscopic driving actions in accordance with optimizing system objective. Platooning is a cooperative driving strategy that improves capacity utilization, safety, and reliability of traffic movement. Autonomous vehicles (AV) equipped with IVC technology are capable of driving in platoons using vehicle-to-vehicle communication (V2V), at very close distances

safely and efficiently. Platooning improves utilization of road capacity, fuel economy, and throughput at the intersections significantly.

Even with fully autonomous vehicular traffic there is a need for safe and efficient intersection management strategy. Traditionally traffic lights are used as a solution to intersection management (IM) of human driven vehicles. However, the advancements in autonomous driving [64] and communication technology [74][75][76] has motivated researchers to extensively investigate autonomous intersection management (AIM) strategies. Google's Waymo has made fully autonomous driving a reality by commercially introducing public self-driving ride-hailing service to commuters in Phoenix, AZ [63]. The early research on intersection management for fully connected and autonomous vehicular traffic in the literature is based on multiagent systems approach [77] which is a subfield of artificial intelligence. In this research authors used a reservation-based intersection control mechanism that schedules space and time for vehicles to move through the intersection based on FIFO and eliminates the need for conventional traffic lights or stop signs but it neither proactively optimize the flow through the intersection nor platoon the AVs for better delays or capacity utilization. Later some MILP based strategies [78][79][80][81] were investigated to determine the release order and optimal speeds for the fixed-path vehicles. In this chapter a real-time DP proactively optimizes the release sequence of AVs and communicates the decisions to a conflict zone based unsignalized intersection control that efficiently controls the approaching AV traffic using V2I communication and schedules a safe passage through the intersection for the AVs with conflicting movements.

2.2 MIDAS AI System Architecture and Development

2.2.1 MIDAS AI System Overview

MIDAS AI stands for managing interacting demand and supply through both centralized and decentralized cloud computing architecture. MIDAS AI is a three-layered architecture that efficiently controls and safely manages the movements of autonomous vehicular (AV) platoons through a network of automated unsignalized (traffic lights free) intersections controlled through MIDAS-AISCU and connected via cloud computing infrastructure, in real time. The very top layer of the architecture (MIDAS-NLC) shown in Figure 2.1 is responsible for managing and scoping network level characteristics. The middle layer (MIDAS-PSMS) effectively communicates with the AV platoons traversing through the network to efficiently manage the formation of platoons based on the origin and destination of AVs, link properties and traffic dynamics and the last layer is called MIDAS-AISCU. AISCU proactively optimizes the departure schedule of AV platoons by minimizing their delays using a forward-recursion dynamic programming problem with a rolling horizon and future state estimation equations (see 2.2.3.2) and safely controls the platoon movements through an automated intersection that is free from conventional signal lights.

NLC stands for network level curator as it captures the network level information about dynamically varying traffic characteristics like link flows, link travel times and wait-times at the intersection nodes in the network. It also captures the slow varying characteristics of physical road network like the link or lane closures, and speed limits, etc. NLC provides critical information to both PSMS and AISCU during several centralized and localized decision epochs to minimize overall delays in the network. The transmitted information is

exchanged to other layers of the system through connector modules as shown in Figure 2.2. The primary goal of this chapter is to highlight the model development and algorithmic implementation behind PSMS and AISCU layers of MIDAS-AI system.

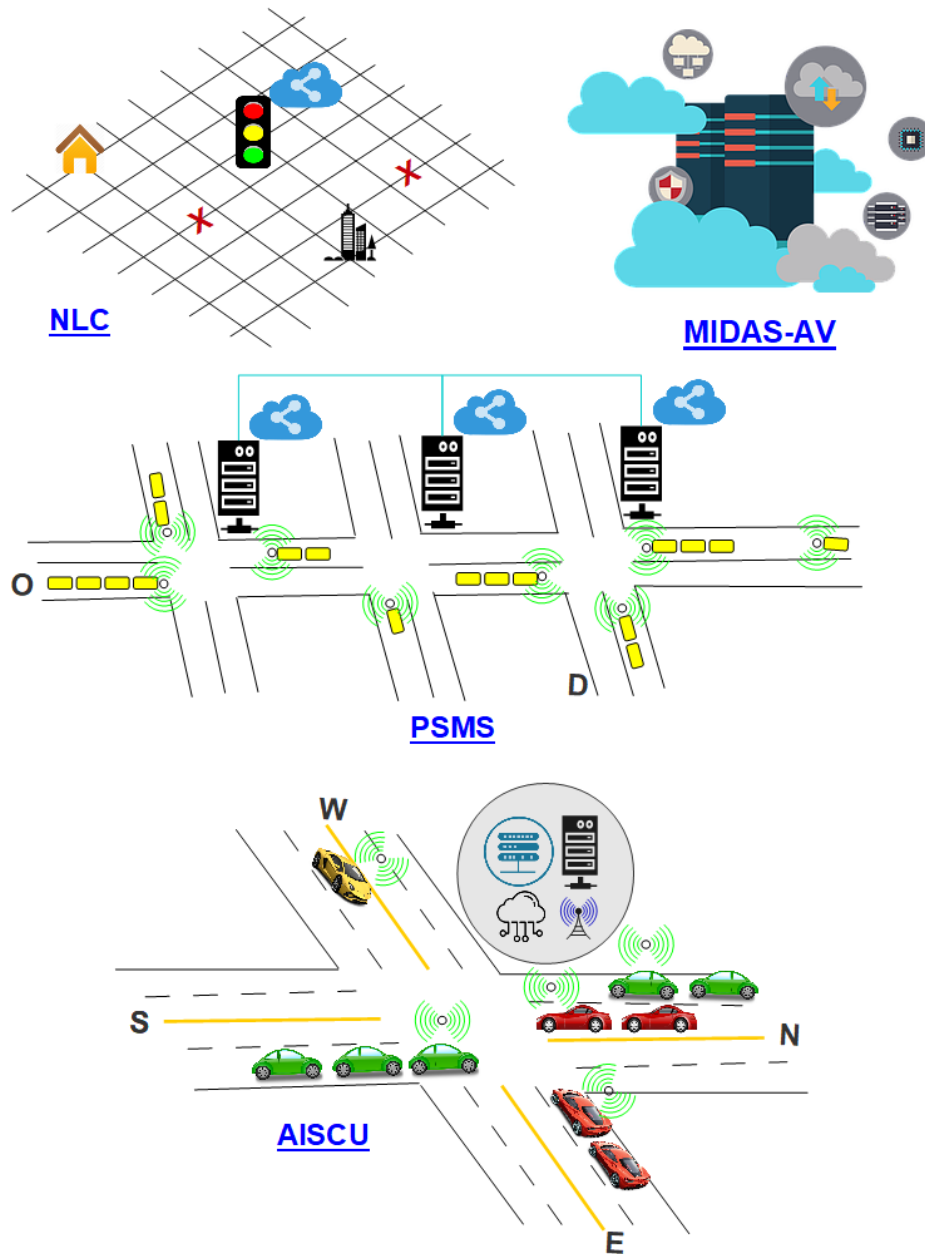


Figure 2.1 MIDAS-AI System Architecture

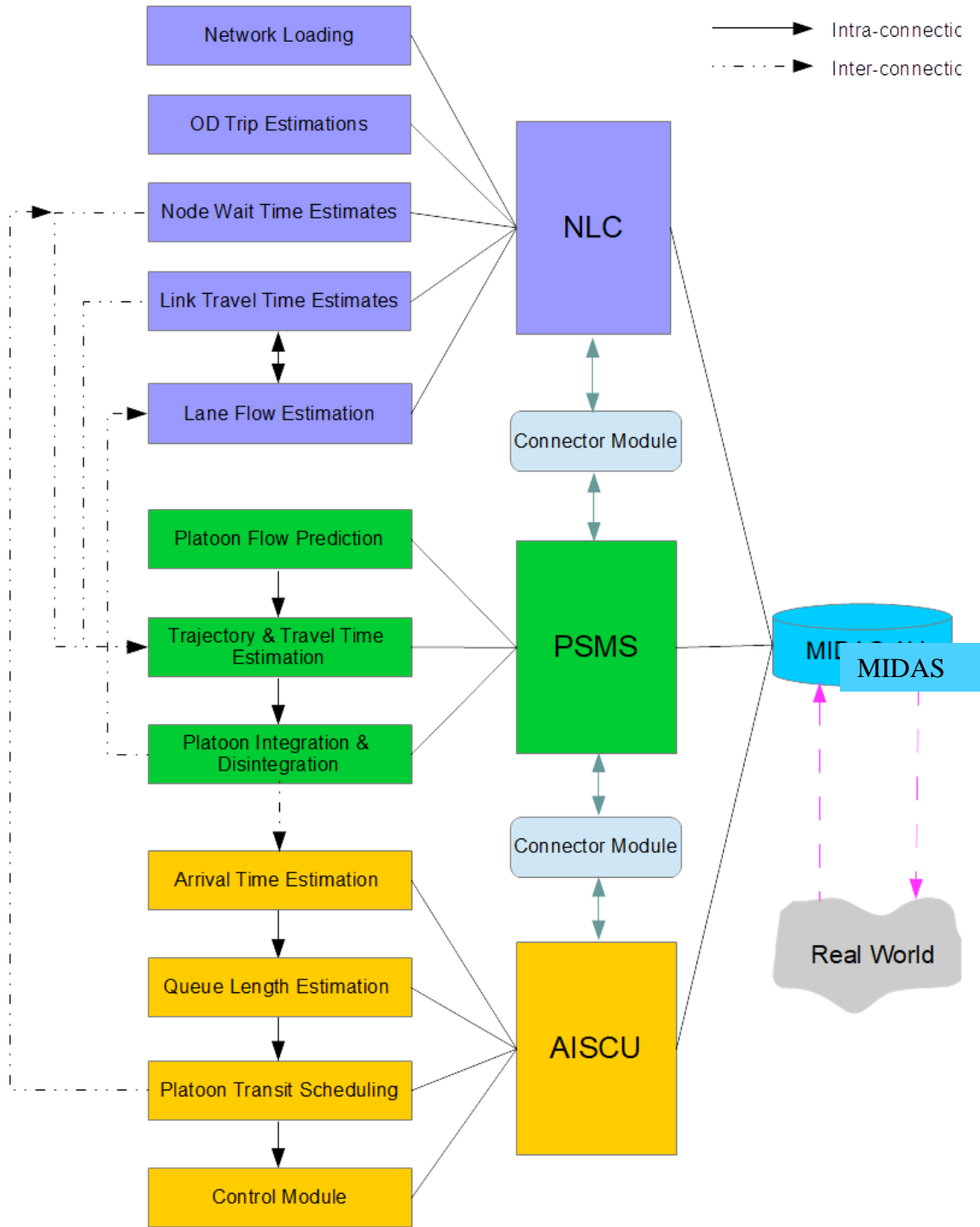


Figure 2.2 MIDAS-AI System Concept of Operations

PSMS stands for platoon scoping and management system. Efficient platooning of AVs based on individual AV origin & destination (OD) information is crucial in improving road capacity utilization and system wide objectives and PSMS system achieves this by predicting mesoscopic characteristics of the network that includes platoon flow prediction on links. PSMS uses efficient platooning logic and data structure to integrate and disintegrate the AV platoons on the network links. The concept of operations for PSMS is shown in Figure 2.3

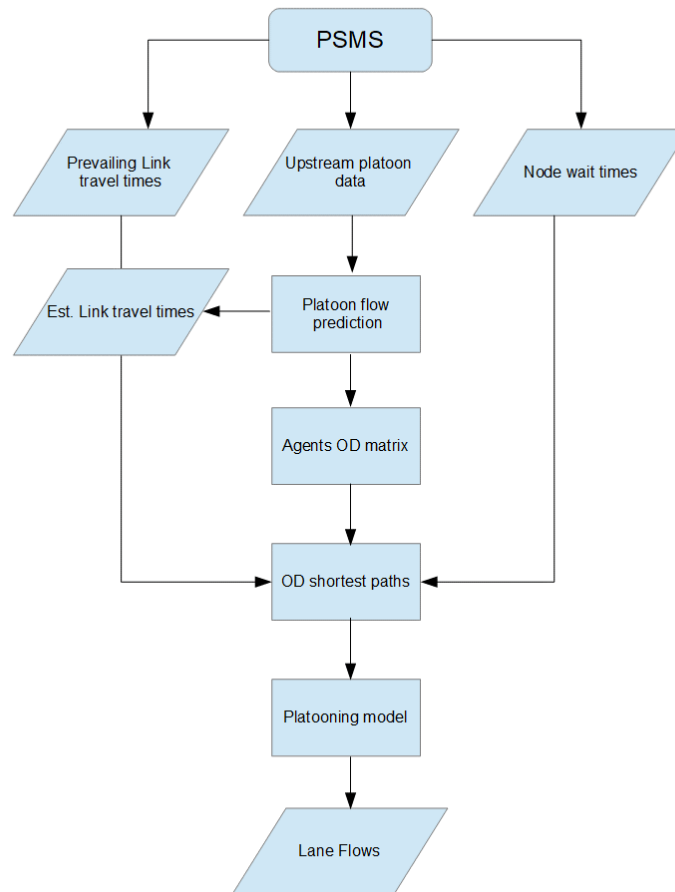


Figure 2.3 PSMS Concepts of Operation

AISCU stands for autonomous intersection scheduling and control unit. The AISCU system is responsible for control and management of AV platoons approaching an intersection. It proactively schedules the transit of platoons/AVs and controls the safe passage through the intersection. AISCU solves a forward recursive Dynamic Programming (DP) with rolling horizon approach, using forward state estimation equations, to schedule AV platoons/AVs movements through the intersection, such that the total time-loss delays due to the conflict movements at the intersection are minimized. The AISCU is a stand-alone system located at each unsignalized intersection, always connected to MIDAS network through V2I communication protocol and can act independently for safer and reliable passage of fleets of AVs through the intersection in real time, without the need for conventional traffic signal lights. AISCU receives AV engine control information like engine variables, vehicle speed, maximum acceleration, etc., in real-time and implements control decisions upon assessing the conflicts for each individual AV/platoon when it reaches the control decision point (refer section 2.2.3.3).

The concept of operations is shown in Figure 2.4. The mathematical formulations and DP algorithm are further investigated in the later sections.

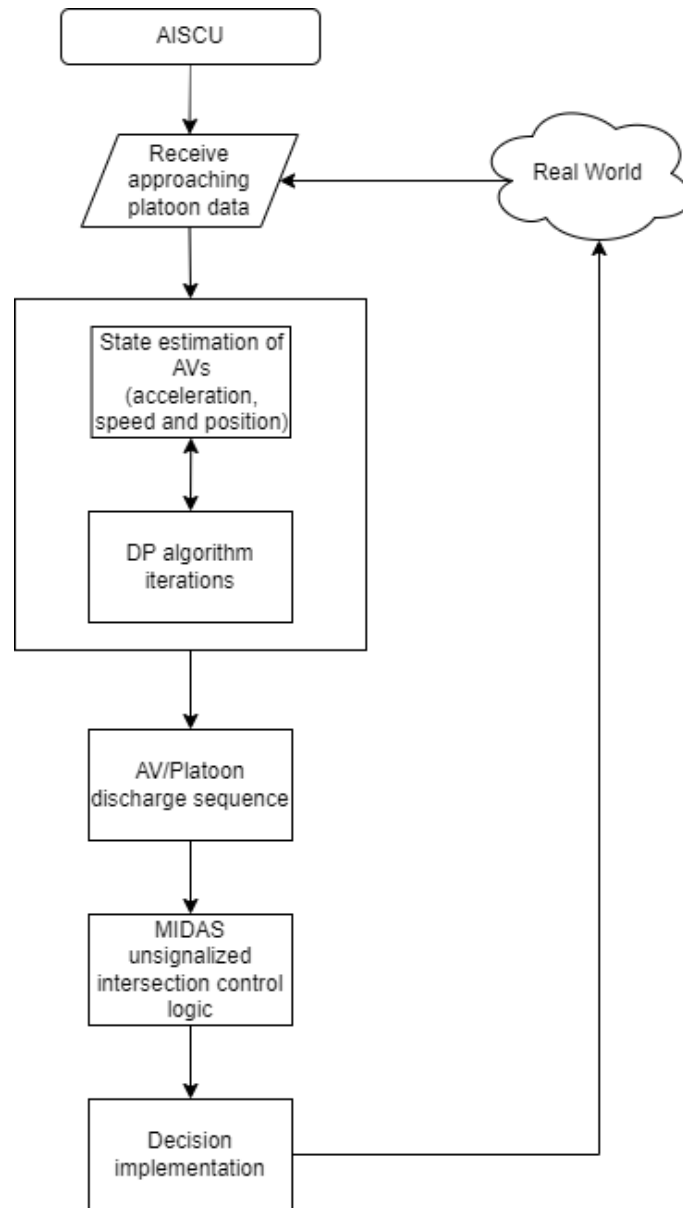


Figure 2.4 AISCUC Concept of Operations

2.2.2 Platoon Scoping and Management System

Coordinating vehicles to travel closely by maintaining only the minimum required safety distances, improves the road capacity utilization by greater margins [65]. A fleet of vehicles propagating closely together for certain time in a road network is called a platoon. To achieve and maintain an optimal platoon formation, constant evaluation of minimum safety

distance or minimum time headway between leading and following vehicle is required [66], which depends on the individual vehicle dynamics. In this section a brief introduction to car following models and advancements in vehicle control and safety systems are discussed. Later in this section, the MIDAS platooning strategy to manage autonomous vehicle platoon formation is introduced and the underlying car following model simulation architecture is also discussed.

2.2.2.1 Leader-Follower Behavior

In general vehicles travel at desired speeds on a congestion-free or free flow road link. But in congestion a vehicle tends to follow the preceding vehicle, maintaining a safe distance by adjusting the speed along its trajectory. This kind of behavior is called a leader-follower behavior. According to simplified car following model [57], the time-space trajectory of follower vehicle is essentially the same as the leader vehicle except for a translation in space and in time as shown in Figure 2.5 below.

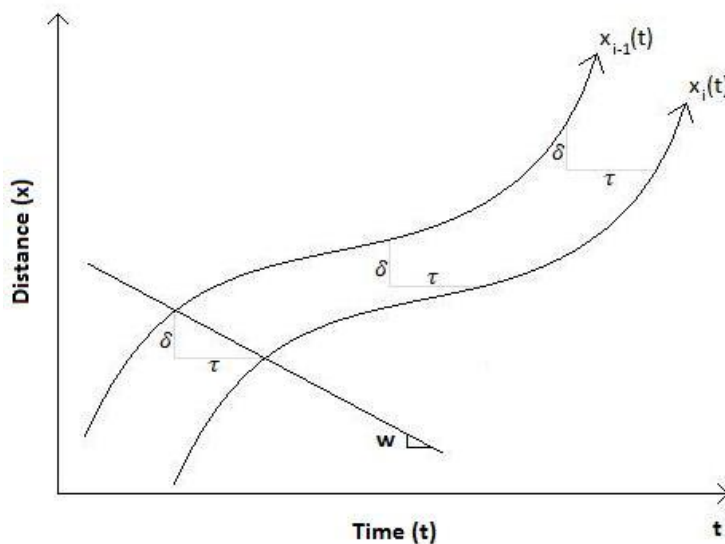


Figure 2.5 Newell Car Following Model

There were several car-following models developed in the literature for human-driven vehicular traffic. The very first car following models were proposed by Pipes & Chandler [57], Some other known models are Gazis-Herman-Rothery model [58], Helly [59], Gipps [60], Wiedemann [61], Krauss [62], and the intelligent driver model [68]. Even with the discovery and implementation of sophisticated car-following models it is highly challenging to accurately model and predict the uncertainty involved in human-driven vehicular traffic. This challenge can be addressed with the help of V2V communication protocol in case of fully autonomous vehicular traffic that can precisely model the driving behavior of AVs in the event of car following by exchanging the engine control parameters wirelessly between the neighboring vehicles, explained in detailed in the next sections.

2.2.2.2 Autonomous Car Following Model

With the development of advanced driver-assistance (ADA) systems like cruise control, lane detection and collision avoidance systems, driving has become more reliable and safer. For driver safety and comfort ADA developed an adaptive cruise control (ACC) system. ACC uses sensors like RADAR, LIDAR, and video cameras to detect the movement of the preceding car and adjusts its speed to maintain a safe distance. ACC enables a safer and reliable car following behavior, also improves the utilization of road capacity due to shorter inter-vehicular gap. Several researchers analyzed the impact of ACC on traffic flow using simulation studies [84][85][86].

The further improvement of longitudinal and lateral control systems leads to the development of vehicle to vehicle (V2V) communication in cars with self-driving capability. A vehicle equipped with V2V or inter vehicular communication (IVC) uses

dedicated short-range communication (DSRC), a wireless protocol to exchange information with other IVC enabled vehicles [87][88]. Cooperative adaptive cruise control is an extension of ACC, uses V2V communication as a feedback loop to obtain the speed, acceleration, position, and other engine variables from the preceding vehicle to adjust its speed and gap accordingly.

Platooning of SDVs is made possible using CACC car following architecture. CACC exhibits better string stability over ACC, as string stability is a measure of disturbances amplifying while the vehicle is propagating downstream through the platoon. CACC is capable of exchanging messages several times a second, leads to accurate estimation of traffic flow. On the other hand, ACC amplifies the measurements in the upstream direction due to braking and acceleration of vehicles in the platoon, leading to poor string stability [89]. In the past several researchers have developed longitudinal control models enabling cooperative driving capability for vehicles with onboard IVC technology, achieving high string stability [82][89][95][96][97]. Researchers have also studied the impact of CACC longitudinal control on the traffic flow characteristics in [90]-[94]. Most of the research efforts made so far had been focused towards developing a string stable longitudinal control for traffic with IVC communication and understanding the benefits of such automated driving on highway capacity and flows. But there is need for developing platooning strategies to determine optimal joining and unjoining decisions based on the ODs and the objective to minimize delays at the intersections in a road network. In this chapter an efficient platooning strategy has been developed to control and manage platooning of AVs through an unsignalized autonomous intersection.

2.2.2.3 MIDAS Platooning Strategy

With the advent of AVs, it is very critical to study and understand the consequences of platooning on traffic flow and road capacity utilization. Unfortunately, there isn't much research study available related to the impact of platooning on capacity utilization and intersection delays. In this dissertation we develop a platooning strategy to proactively control and manage AV platoons in a traffic network, in real-time and perform simulation studies to understand time loss, throughput, and travel times, etc.

At every decision epoch (usually when the vehicle enters an intersection corridor), MIDAS can calculate the estimated trajectories of agents (AVs) upstream, arriving at the intersection by calculating shortest paths based on estimated traffic flow in the network. Once the trajectory information of the arriving AVs is updated MIDAS platooning model encourages an AV to join a platoon with maximum overlap time in the updated platoon trajectory, subject to maximum platoon length and posted link speeds etc. Similarly, an AV unjoins a platoon when it doesn't have any matching arc in the updated platoon trajectory, like shown in Figure 2.6

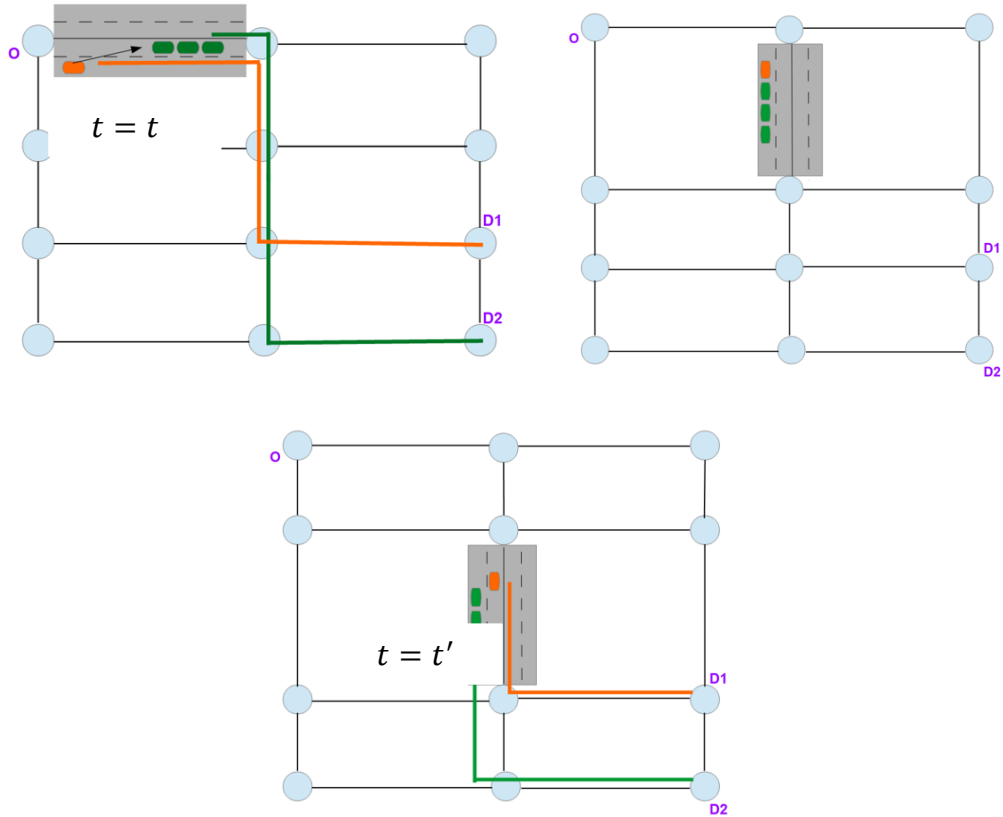


Figure 2.6 MIDAS Platooning Strategy

u_{ij}^t = speed of AV or platoon i in movement j at time t .

pos_{ij}^t = position of AV or platoon i in movement j at time t from downstream intersection.

A_{ij}^{max} = maximum allowed acceleration for AV or platoon i in movement.

l_j = allowed length of platoon on movement j ; number of vehicles in a platoon

p_{ij}^t = best path determined for AV or platoon i in movement j to it's destination at time t .

P_i^t = set of platoons paths that has overlap with p_{ij}^t

g_{ij}^{avg} = minimum average gap maintained by vehicles in platoon i in

movement j .

$T_{(i,i-1)j}^t =$ estimated time for vehicle i to join platoon $i - 1$ in movement j

at time t .

$\tau_{ij}^t =$ estimated max time for platoon i in movement j to reach the

downstream intersection at time t .

$$T_{(i,i-1)j}^t = \frac{-(u_{i-1j}^t - u_{ij}^t) \pm \sqrt{(u_{i-1j}^t - u_{ij}^t)^2 - 2 * A_{ij}^{max} * (pos_{ij}^t - pos_{i-1j}^t - g_{ij}^{avg})}}{A_{ij}^{max}} \quad (2.1)$$

$$\tau_{i-1j}^t = \frac{pos_{i-1j}^t}{u_{i-1j}^t} \quad (2.2)$$

MIDAS continuously evaluates the best path for each vehicle to its destination at every epoch (usually when the vehicle enters an intersection corridor) and determines the set of platoons ahead of vehicle with overlapping trajectories. Assuming MIDAS provides an estimated path p_{ij}^t to travel for vehicle i in movement j at time t .

Step 1: Given $u_{ij}^t, p_{ij}^t, P_i^t, l_j, A_{ij}^{max}, g_{ij}^{avg}$

for each $(p \in P_i^t, p_{ij}^t)$ pair:

calculate the estimated overlap path-time duration $T = t' - t$ for the path pairs, for example as shown in Fig 2.6 vehicle joins platoon at $t = t'$ and disjoins at $t = t'$

Step 2: sort set P_i^t by T by descending order.

Step 3: for each $(p \in P_i^t, p_{ij}^t)$ pair:

If $|p| + 1 \leq l_j$:

Estimate $T_{(i,i-1)j}^t$ and τ_{i-1j}^t using equation (2.1) & (2.2)

If $T_{(i,i-1)j}^t < \tau_{i-1j}^t$:

Go to Step 4

Else:

Continue

Else:

Continue.

Step 4: accelerate vehicle i to join platoon p

Step 5: Terminate

Algorithm 2.1. MIDAS Platooning Strategy for Autonomous Vehicular Traffic

2.2.3 MIDAS Autonomous Intersection Scheduling and Control Unit (AISCU)

2.2.3.1 Autonomous Intersection Control Problem Formulation

Efficient platoon movement at the intersection contributes to less time loss, higher throughputs, and minimization of overall travel times of traffic in the network. Unlike fixed time or reactive control, MIDAS- AISCU proactively optimizes platoon scheduling at the intersection. Autonomous intersection control optimization is similar to a job scheduling problem with platoons as jobs and intersection movements as machines. Hence this problem is formulated as integer linear programming (ILP) as below.

Parameters: -

J = set of possible movements present at the intersection.

a_{ij} = estimated arrival time of platoon i in movement j at the back of the queue, at the intersection.

d_{ij} = departure time needed for platoon i in the movement j to clear the intersection.

f_{ij} = free flow time needed for platoon i in the movement j to clear the intersection.

P_j = set of platoons in the movement j .

C'_j = set of movements in conflict with movement j .

Decision Variables: -

r_{ij} = scheduled release time of platoon i of movement j , through the intersection.

Objective

$$\min \sum_{i \in P_j} \sum_{\forall j} (r_{ij} + d_{ij}) - (a_{ij} + f_{ij})$$

Constraints

$$r_{ij} \geq a_{ij} \quad \forall i \in P_j, \forall j \in J \quad (2.3)$$

$$r_{i+1j} \geq r_{ij} \quad \forall i \in P_j, \forall j \in J \quad (2.4)$$

$$r_{i'j'} < r_{ij} + M \cdot z_{ij i'j'} \quad \forall i' \in P_{j'}, \forall j' \in C'_j, \quad \forall i \in P_j, \forall j \in J \quad (2.5)$$

$$r_{i'j'} \geq r_{ij} + d_{ij} - M \cdot (1 - z_{ij i'j'}) \quad \forall i' \in P_{j'}, \forall j' \in C'_j, \forall i \in P_j, \forall j \in J \quad (2.6)$$

$$r_{ij} \in I^+, \quad \mathbf{z} \in \mathbf{B}, \quad M \text{ is a large + integer} \quad (2.7)$$

The objective function minimizes the time loss of platoons at the intersection proactively.

The first part of the objective function is the time at which platoon i leaves intersection through movement j and the second part is the time it would've left the intersection if there wasn't any delay (traveling at free flow speed with no time loss). Constraint (2.3) enforces that the release time of the platoon from its corresponding queue should be at least equal to the arrival time of the platoon at the intersection (or back of the queue). Constraint (2.4) in the formulation makes sure that platoons are released from the queue in FIFO order.

Constraints (2.5) & (2.6) take care of the scenarios where conflicting platoon movements happening at the same time. Constraint (2.7) defines the variable types.

2.2.3.2 Dynamic Programming Approach for Scheduling AV Departures

The dynamic programming (DP) approach discussed in chapter 1 controls connected vehicular traffic efficiently by determine sequence of phases and duration of green time units allotted to each phase by minimizing the delays at the intersection. But the concept of traffic lights and green time units become obsolete in the era of completely autonomous vehicular traffic. In this chapter a signal free traffic control system is introduced to control the autonomous vehicular movements through the intersection. A new DP approach is developed to determine the optimal sequence of AVs to release from the intersection by minimizing the time loss of vehicles due to propagated congestion upstream from intersection stop delays. It is assumed that all the AVs are connected to AISCU in real-time and share their engine parameters. At every second AISCU receives the state of the intersection corridor that includes the position, speed, and other engine parameters of every individual vehicle in the intersection corridor and DP stage and states are estimated. AISCU also communicates the control decisions with the AVs in real-time and without loss of generality (WLOG) it is assumed that 100% of the vehicles obey the control decisions transmitted by AISCU. The traffic lights free autonomous intersection control (IC) logic of MIDAS AISCU is responsible for safe and efficient vehicle movements at the intersection. AISCU seamlessly implements MIDAS DP decisions in real-time by performing series of safety checks, more details in the section 2.2.3.3.

The following section introduces the MIDAS DP model methodology and the underlying algorithmic construct. In the beginning of this section, DP model notation is defined and followed by the equations to estimate the system state variables. For the sake of brevity and DP illustration, we assume a simple intersection setting with 4-legs with through and left-turn movements.

DP Notation: -

N is set of stages in DP or Number of vehicles at the intersection corridor at the beginning of DP.

\mathbf{a}_j is an action (to release the next feasible AV) from the movement j .

C_j is set of complementary movements associated with movement j that doesn't conflict with action \mathbf{a}_j .

\mathbf{s}_n is a state in the current stage. It is number of vehicles released from each approach at the intersection by the end of the stage n . Example (W: 2, E: 5, N: 0, S: 3)

\mathbf{s}'_{n-1} is the state in previous stage $n-1$ from where you get to \mathbf{s}_n by taking an action \mathbf{a}_j

$V(\mathbf{s}_n)$ is value of being in the state \mathbf{s}_n

$A^\pi(\mathbf{s}_n)$ is a decision function that determines the optimal action to take in the state \mathbf{s}_n

$L(\mathbf{s}_n, \mathbf{a}_j)$ is the total loss time associated with an action \mathbf{a}_j , in the state \mathbf{s}_n

$$V(\mathbf{s}_n) = \min_{\mathbf{a}_j} (L(\mathbf{s}_n, \mathbf{a}_j) + V(\mathbf{s}'_{n-1})) \quad (2.8)$$

$$A^\pi(\mathbf{s}_n) = \arg \min_{\mathbf{a}_j} (L(\mathbf{s}_n, \mathbf{a}_j) + V(\mathbf{s}'_{n-1})) \quad (2.9)$$

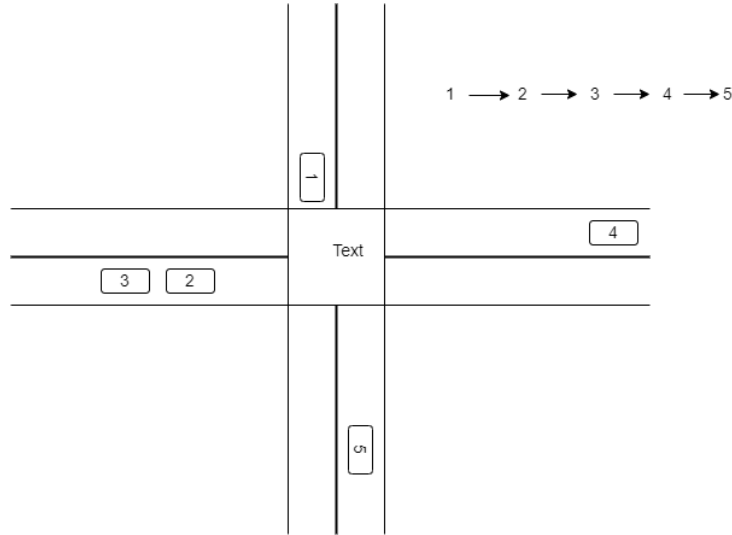


Figure 2.7. DP Solution with Release Sequence of Vehicles Approaching the Intersection.

MIDAS DP is scheduled to run every second by capturing the dynamics of the intersection corridor to determine the optimal sequence of AVs/Platoon releases and a typical DP decision sequence is shown in Figure 2.7. As per Bellman’s principle of optimality [67] the optimal value of being in a state can be expressed using (2.8). The horizon length of DP is defined as the number of AVs/Platoons present in the intersection corridor at the beginning of each DP run. DP consists of sequence of stages and a stage in DP is defined as the number of vehicles released by the end of that stage with number of vehicles released from each movement bound as a state in the considered stage. An action in each state in this DP formulation is a decision to release an AV/ platoon from one of the movement bounds. For every DP state transition, there will be an update to MIDAS system state variables like speed, position of AVs. The update to the variables is made for each AV using iterative kinematic estimation equations as proposed below.

$u_j = \text{speed of the vehicle in the first position on the movement } j, \text{ at the}$

beginning of stage n.

V_j^{max} = *max link speed allowed on the movement j or desired speed.*

A_{ij}^{free} = *free acceleration given to vehicle i in movement j.*

B_{ij}^{max} = *maximum deceleration given to vehicle i in movement j.*

A_{ij}^t = *acceleration of vehicle i in movement j.*

t_j = *block time, max time required to clear off the vehicle in the first position on the movement j*

d_j = *safety distance required for the vehicle in the first position on the movement j to travel before an AV from conflicting movement is released, to clear off the intersection.*

$d_{pot_{ij}}^t$ = *estimated distance travelled by AV i in movement j during t and t + 1*

$u_{pot_{ij}}^t$ = *estimated speed of AV i in movement j during t and t + 1*

n_{ij}^t = *length of platoon; number of AVs in platoon i in movement j at time t*

x_{ij}^t = *distance between AV i and AV i - 1 on movement j at time t.*

Besides the proposal of autonomous intersection control and management, one of the major contributions in this chapter is to study the impact on vehicle throughput and delays at the intersection by platooning AV traffic through cooperative driving strategy, using IVC technology. To achieve these two different autonomous driving strategies are tested using a brand-new DP algorithm proposed in algorithm 2.2.

1) AV traffic with disabled IVC technology but assumed to have enabled continuous V2I communication.

2) AV traffic with both IVC and V2I communications enabled and is assumed to obey the MIDAS platooning strategy 100% of the time.

$$x_{ij}^t = \begin{cases} pos_{ij}^t & \text{if } i = 0, \\ pos_{ij}^t - pos_{i-1j}^t, & \text{otherwise} \end{cases} \quad (2.10)$$

State estimation for AV traffic in case of disabled IVC and an effective platooning strategy

In this scenario MIDAS DP assumes that AVs have disabled the onboard IVC communication and are not able to adopt CACC car following behavior and hence follows an improved version of intelligent driver model (IDM) [68] for longitudinal dynamics like acceleration or deceleration etc. The original IDM model formulates the acceleration or deceleration of a following vehicle as a function of current speed, desired speed (max allowed speed on the link), actual distance and desired safe distance between the follower vehicle and leading vehicle as shown in (2.11). The downside of the original IDM is that when AV travels at speeds close to the desired speed of the link model overestimates large safe distance and vehicles tend to disperse at such speeds. To overcome this problem Treiber and Kesting [69] proposed an improved IDM model that estimates the dynamics of vehicles at even desired link speed as defined in (2.13) - (2.16).

$\tau = \text{safe time gap}$

$x^*(u_{ij}^t, \nabla u_{ij}^t) = \text{desired distance between AV } i \text{ and AV } i - 1 \text{ on movement } j$

at time t .

$x_0 =$ minimum bumper to bumper gap needed

$\delta =$ acceleration exponent constant

$$A_{ij}^{t+1} = A_{ij}^{max} \left[1 - \left(\frac{u_{ij}^t}{V_j^{max}} \right)^\delta - \left(\frac{x^*(u_{ij}^t, \nabla u_{ij}^t)}{x_{ij}^t} \right)^2 \right] \quad (2.11)$$

$$x^*(u_{ij}^t, \nabla u_{ij}^t) = \left\{ \begin{array}{l} \frac{(u_{0j}^t)^2}{2\sqrt{A_{0j}^{max} B_{0j}^{max}}} \quad \text{if } i = 0, \\ x_0 + \max(0, u_{ij}^t * \tau + \frac{u_{ij}^t \nabla u_{ij}^t}{2\sqrt{A_{ij}^{max} B_{ij}^{max}}}), \quad \text{otherwise} \end{array} \right\} \quad (2.12)$$

$$A_{ij}^{free} = \left\{ \begin{array}{l} A_{ij}^{max} \left[1 - \left(\frac{u_{ij}^t}{V_j^{max}} \right)^\delta \right] \quad \text{if } u_{ij}^t \leq V_j^{max} \\ -B_{ij}^{max} \left[1 - \left(\frac{V_j^{max}}{u_{ij}^t} \right)^{\frac{A_{ij}^{max} \delta}{B_{ij}^{max}}} \right] \quad \text{if } u_{ij}^t > V_j^{max} \end{array} \right\} \quad (2.13)$$

$$A_j^{free} = A_{ij}^{free} \mid i = 0 \ \& \ u_{ij}^t = u_j \quad (2.14)$$

$$A_{ij}^t \mid u_{ij}^t \leq V_j^{max} = \left\{ \begin{array}{l} A_{ij}^{max} (1 - z^2) \quad z = \frac{x^*(u_{ij}^t, \nabla u_{ij}^t)}{x_{ij}^t} \geq 1, \\ A_{ij}^{free} \left(1 - z^{2A_{ij}^{max}/A_{ij}^{free}} \right) \quad \text{otherwise} \end{array} \right\} \quad (2.15)$$

$$A_{ij}^t \mid u_{ij}^t > V_j^{max} = \left\{ \begin{array}{l} A_{ij}^{free} + A_{ij}^{max} (1 - z^2) \quad z = \frac{x^*(u_{ij}^t, \nabla u_{ij}^t)}{x_{ij}^t} \geq 1, \\ A_{ij}^{free} \quad \text{otherwise} \end{array} \right\} \quad (2.16)$$

$$t_j = \frac{-u_j \pm \sqrt{u_j^2 - 2 * A_j^{free} * d_j}}{A_j^{free}} \quad (2.17)$$

$$u_{pot_{ij}^{t+1}} = u_{ij}^t + A_{ij}^{free} \quad (2.18)$$

$$d_{pot_{ij}^{t+1}} = u_{ij}^t + \frac{A_{ij}^{free}}{2} \quad (2.19)$$

$$u_{ij}^{t+1} = \left\{ \begin{array}{l} u_{pot_{ij}^{t+1}} \text{ if } i = 0, A^\pi(s_n) = a_j, \text{ and } A^\pi(s_{n+1}) = a_j: j \in C_{j'} \\ u_{ij}^t + A_{ij}^t |_{u_{ij}^t \leq V_j^{max}} \text{ if } u_{ij}^t \leq V_j^{max} \\ u_{ij}^t + A_{ij}^t |_{u_{ij}^t > V_j^{max}}, \quad \text{otherwise} \end{array} \right\} \quad (2.20)$$

$$pos_{ij}^{t+1} = \left\{ \begin{array}{l} pos_{ij}^t - d_{pot_{ij}^{t+1}} \text{ if } i = 0, A^\pi(s_n) = a_j, \text{ and } A^\pi(s_{n+1}) = a_j: j \in C_{j'} \\ pos_{ij}^t + u_{ij}^t + \frac{A_{ij}^t |_{u_{ij}^t \leq V_j^{max}}}{2} \text{ if } u_{ij}^t \leq V_j^{max} \\ pos_{ij}^t + u_{ij}^t + \frac{A_{ij}^t |_{u_{ij}^t > V_j^{max}}}{2}, \quad \text{otherwise.} \end{array} \right\} \quad (2.21)$$

State estimation for AV traffic in case of enabled IVC and an effective platooning strategy

t_{sg} = inter string time gap maintained between AVs with in a platoon

t_{dg} = desired time gap maintained between AVs under ACC mode

k_1 = gain constant for speed difference between link free flow speed and subject AV current speed

k_2 = gain constant for positional difference between subject AV and preceding AV

k_3 = gain constant for speed difference between subject AV and preceding AV

k_p, k_d = gain constants for adjusting time gap between subject AV and preceding AV

$e_{ij}^t = \text{time gap error for vehicle } i \text{ in movement } j \text{ at time } t$

In this scenario MIDAS DP assumes that AVs have enabled the onboard IVC communication and are able to adopt CACC car following behavior that supports the formation of AV platoons controlled by MIDAS platooning strategy. The cooperative driving strategy adopted by the AV traffic in this scenario is based on the CACC car following model described in [82] [83]. It is assumed that an AV under this driving strategy would be capable of driving in 3 different modes as per the conditional logic defined in Figure 2.8.

1. *Speed regulation mode*

In this driving mode controller recommends the subject AV to attain the free flow link speed with an acceleration defined in (2.22) when the gap between the subject AV and the preceding AV is greater than the on-board sensor detection range which is usually set as a maximum threshold distance value of 100 meters.

$$A_{ij}^t = \min(A_{ij}^{max}, k_1(V_j^{max} - u_{ij}^{t-1})) \quad (2.22)$$

2. *Gap regulation mode*

In this driving mode controller recommends the subject AV to maintain a safe distance and follow the preceding AV using the on-board sensors and ACC mode acceleration described in (2.23) when the preceding AV is in the detection range.

$$A_{ij}^t = \min(A_{ij}^{max}, k_2(x_{ij}^t - t_{dg}u_{ij}^{t-1}) + k_3(u_{i-1j}^{t-1} - u_{ij}^{t-1})) \quad (2.23)$$

3. Follower gap regulation mode

In this driving mode controller recommends the subject AV to maintain a constant in-string gap with the preceding AV in the platoon using the CACC mode speed and acceleration described in (2.24-2.27)

$$u_{ij}^t = u_{ij}^{t-1} + k_p e_{ij}^t + k_d \dot{e}_{ij}^t \quad (2.24)$$

$$A_{ij}^t = u_{ij}^t - u_{ij}^{t-1} \quad (2.25)$$

$$e_{ij}^t = x_{ij}^t - t_{sg} u_{ij}^{t-1} \quad (2.26)$$

$$\dot{e}_{ij}^t = u_{i-1j}^{t-1} - u_{ij}^{t-1} - t_{sg} A_{ij}^{t-1} \quad (2.27)$$

In this scenario, the free flow acceleration of an AV A_{ij}^{free} is estimated using same speed regulation mode equation (2.22)

$$u_{ij}^{t+1} = \begin{cases} u_{pot_{ij}^{t+1}} & \text{if } i = 0, A^\pi(s_n) = a_{j'}, \text{ and } A^\pi(s_{n+1}) = a_j: j \in C_{j'} \\ u_{ij}^t + A_{ij}^t, & \text{otherwise} \end{cases} \quad (2.28)$$

$$pos_{ij}^{t+1} = \begin{cases} pos_{ij}^t - d_{pot_{ij}^{t+1}} & \text{if } i = 0, A^\pi(s_n) = a_{j'}, \text{ and } A^\pi(s_{n+1}) = a_j: j \in C_{j'} \\ pos_{ij}^t + u_{ij}^t + \frac{A_{ij}^t}{2}, & \text{otherwise} \end{cases} \quad (2.29)$$

$$L(s_n, a_j) = \sum_{t=1}^{t_j} \sum_{\forall j} \sum_{\forall i} \left(1 - \frac{u_{ij}^t}{V_{max}^j} \right) * n_{ij}^t \quad (2.30)$$

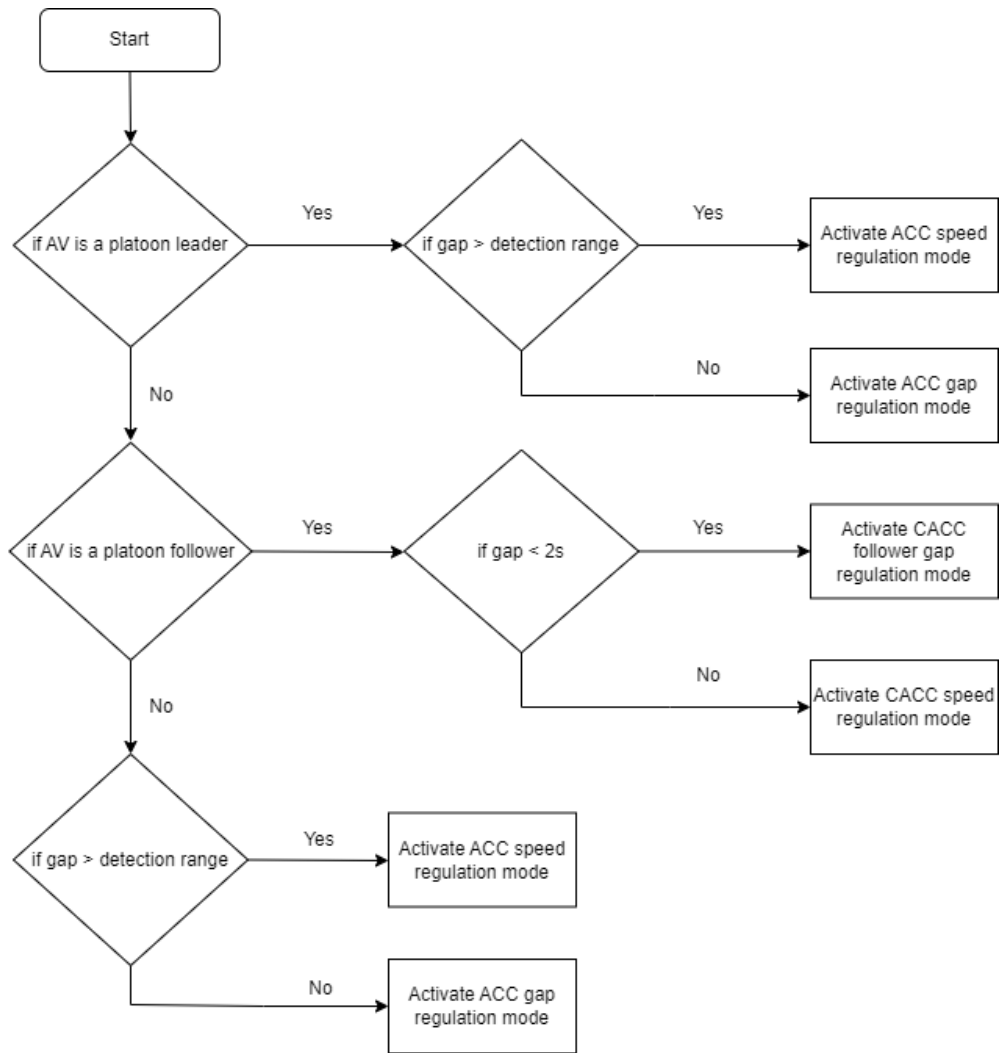


Figure 2.8. Car Following Model Activation Logic for AVs.

Step 1:

Capture the MIDAS system state from SIMULATION environment and initialize variable and parameters.

$$S_0 = [N_W, N_E, N_N, N_S]$$

$$N = N_W + N_E + N_N + N_S$$

Initialize DP stage $n = 1$

Step 2:

Generate all DP states possible in the current stage. Example stage $n=1$ has following states.

$$States(n = 1) = [1,0,0,0], [0,1,0,0], [0,0,1,0], [0,0,0,1]$$

Step 3:

for state s_n in $States(n)$:

a) Calculate $V(s_n) = \min_{a_j} (L(s_n, a_j) + V(s'_{n-1}))$, where $A^\pi(s_{n+1}) = a_{j'} : j' \in C_j$

b) Calculate $V(s_n) = \min_{a_j} (L'(s_n, a_j) + V(s'_{n-1}))$, where $A^\pi(s_{n+1}) = a_{j'} : j' \notin C_j$

c) delay function for a given state and an action, $L(s_n, a_j)$ is calculated using (2.30)

d) transition DP state as shown below

$$s'_{n-1} = M(s_n, a_j)$$

$$M(s_n, a_j) = s_n : s_n^j = s_n^j - 1$$

e) if platooning is ON:

estimate MIDAS system state variables using eqn (2.22) – (2.29)

else:

estimate MIDAS system state variables using eqn (2.11) – (2.21)

Step 4:

$n = n + 1$, go to Step 1 if $n < N$ else go to Step 5

Step 5:

Retrieve optimal sequence of release actions for the AVs accessing the intersection at the current time step using recursion.

Algorithm 2.2. MIDAS DP Logic to Determine Optimal AV/Platoon Release Sequence.

2.2.3.3 Unsignalized Intersection Control Logic for Autonomous Vehicular Traffic

MIDAS AI uses a signal free intersection control strategy that evaluates DP decisions calculated every sec. To ensure safe traffic movements at the intersection we define control points for AVs, upstream. Control point distance for a given AV is defined as the distance needed for the vehicle to decelerate safely to make a stop at the stop line of the intersection.

Control point distance of a vehicle is determined based on the current speed, location and the max deceleration allowed for the vehicle.

u_{ij}^t = speed of AV i in movement j at time t

pos_{ij}^t = position of AV i in movement j at time t from stop line.

A_{ij}^{max} = maximum acceleration $\frac{m}{s^2}$ given to vehicle i in movement j .

B_{ij}^{avg} = average deceleration in $\frac{m}{s^2}$ for vehicle i in movement j .

R^t = set of vehicles clearing the intersection or given clearance during time t .

D^t = release decision sequence determined by DP at time t .

d_{ij}^t = control point distance required for vehicle i in movement j from the stop line, to come to a stop.

τ_{ij}^t = min time required for vehicle i in movement j to reach the conflict zone.

$$d_{ij}^t = \frac{(u_{ij}^t)^2}{2B_{ij}^{avg}} + \sum_{i=0}^{i-1} n_{ij}^t \quad (2.31)$$

$$\tau_{ij}^t = \frac{-u_{ij}^t \pm \sqrt{(u_{ij}^t)^2 - 2 * A_{ij}^{max} * pos_{ij}^t}}{A_{ij}^{max}} \quad (2.32)$$

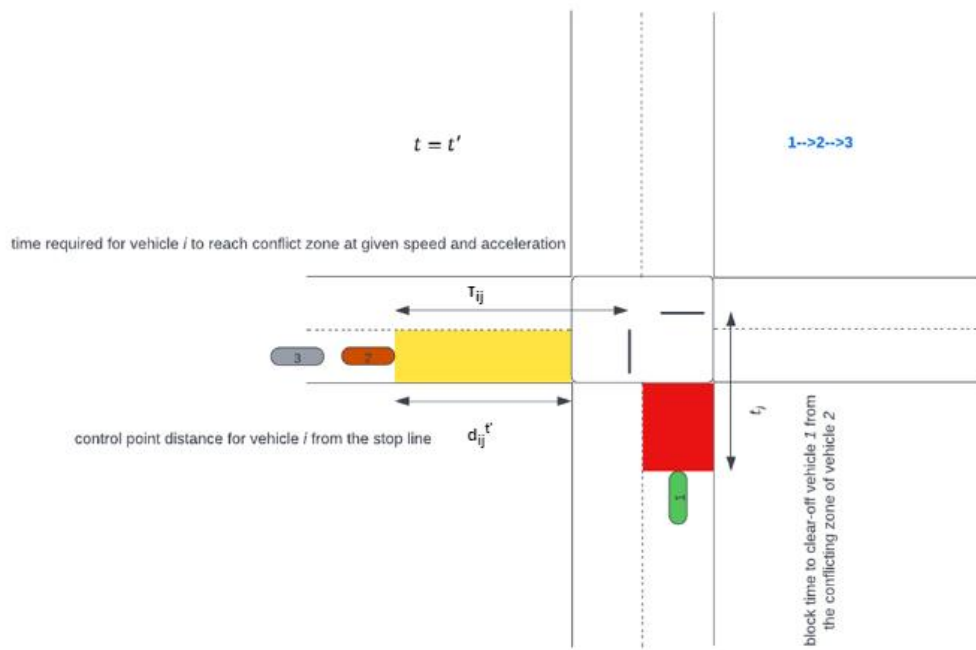


Figure 2.9 Unsignalized Intersection Control Logic for Autonomous Vehicles.

Step1: Given the DP vehicle release decision sequence D^t

- initialize $seq = 0$
- set $slow_down_flag = False$
- calculate the block time required $t_{ij} \forall v_{ij} \in R^t$ using eqn (2.17)

Step2: for vehicle $v_{ij} \leftarrow D^t(seq)$

- Calculate the control point distance d_{ij}^t for vehicle v_{ij} using eqn (2.31)
- Calculate τ_{ij}^t the time required to reach conflict zone for vehicle v_{ij} using eqn (2.32)
- if: $slow_down_flag$ is True
 - go to step4

Step3: if: $pos_{ij}^t \leq d_{ij}^t$

- if: $\tau_{ij}^t > t_{ij'} \forall v_{ij'} \in R^t$ and $j' \notin C_j$
 - give clearance to the vehicle v_{ij} and add v_{ij} to R^t
 - also calculate the block time required t_{ij} using eqn (2.17)
 - go to step2
- else:
 - slow down the vehicle v_{ij}
 - set $slow_down_flag = True$
- else: go to step5.

Step4: if: $pos_{ij}^t \leq d_{ij}^t$

- slow down the vehicle v_{ij}

Step5: if end of seq then terminate

- else: $seq \leftarrow seq + 1$
- go to step2

Algorithm 2.3. Unsignalized (Traffic-Lights-Free) Intersection Control Logic for AV Traffic

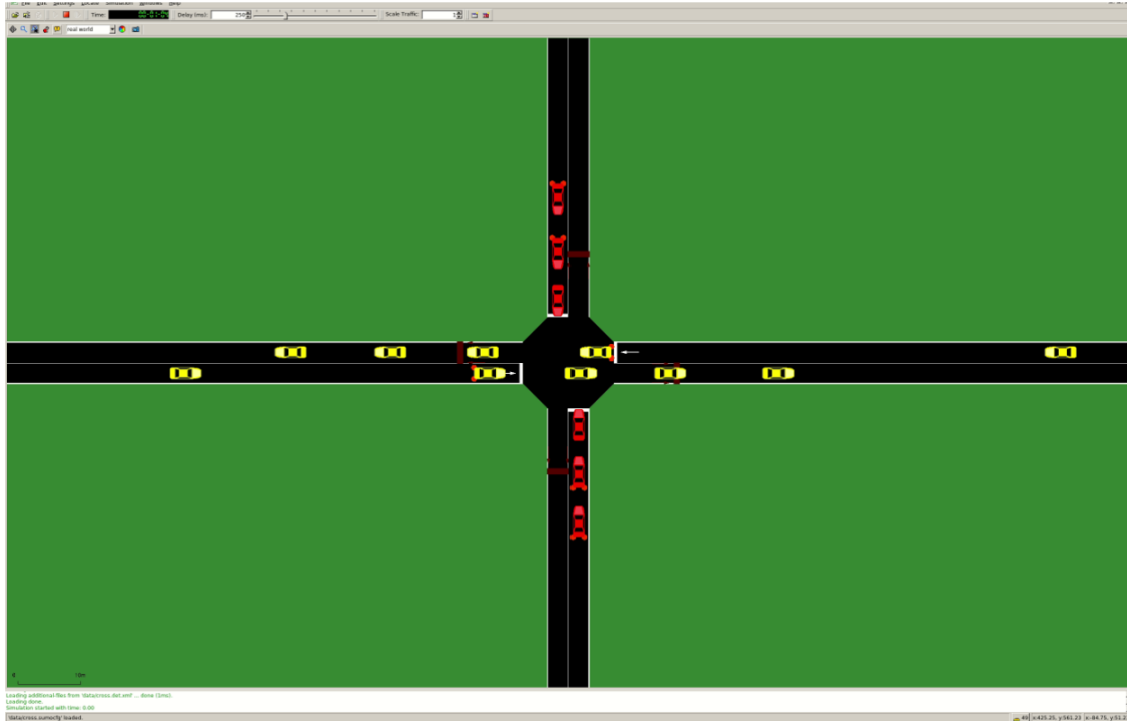


Figure 2.10. Autonomous Intersection Control & Management of Unsignalized Intersection

2.3 MIDAS AI Simulation Framework

To evaluate the MIDAS proactive autonomous intersection control logic, in the real world we require a complete cyber-physical infrastructure with signal-free intersection corridor that supports V2X communication capabilities within the participating AV traffic. To overcome this challenge, we implemented MIDAS AI in an open-source vehicular network simulation framework that combines a microscopic simulator called SUMO [71], a network simulator called VEINS [72] and a cooperative driving framework called PLEXE [73]. VEINS stands for vehicles in network simulation and this simulation framework is developed over OMNET++, an event-based network simulation.

Each simulation in VEINS triggers the network simulator (OMNET++) and the microscopic road traffic simulator (SUMO) in parallel.

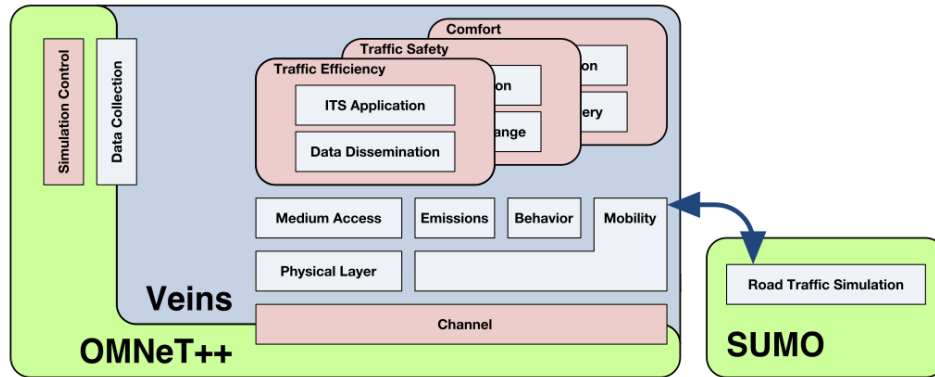


Figure 2.11. Vehicular Communication Architecture

Veins extends the OMNeT++ network simulator by providing a complete vehicular communication stack based on IEEE 802.11p [70], together with a way of modeling realistic node mobility based on the road traffic simulator SUMO as shown in Figure 2.11. For this it couples the network and the mobility simulator by creating a network node in OMNeT++ for each vehicle travelling in SUMO. Veins replicate the real-time movement of a vehicle in SUMO simulation in the corresponding OMNeT++ node by updating the mobility model. The communication between the OMNeT++ and SUMO is done through TRACI interface. By using this interface, Veins queries SUMO about current “traffic” status (e.g., number of vehicles, their position and speed, etc.), and it can modify the traffic dynamics, route, speed, or its acceleration. PLEXE further extends the interaction through the TraCI interface to fetch vehicles’ data from SUMO to be sent to other cars, and to be used by the platooning protocols and MIDAS application. The data received by vehicles in Veins are fed to the CACC enabled AVs simulating in SUMO via PLEXE as shown in

Figure 2.12, such that the platooning decisions of AVs can be controlled and managed by MIDAS platooning logic.

As shown in Figure 2.13 MIDAS communicates the platooning decisions through PLEXE module and simultaneously implements the DP control decisions along with the safe unsignalized intersection control of conflicting vehicle movements through the intersection via TRACI interface.

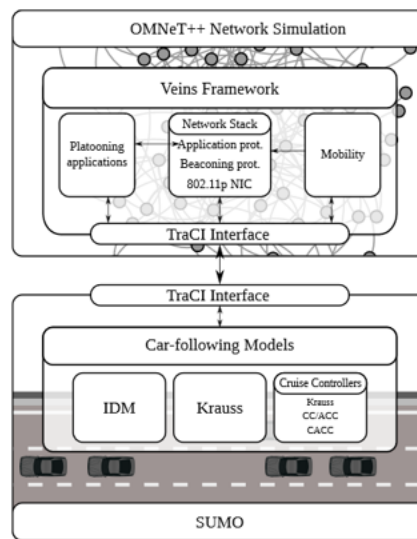


Figure 2.12. PLEXE Configuration

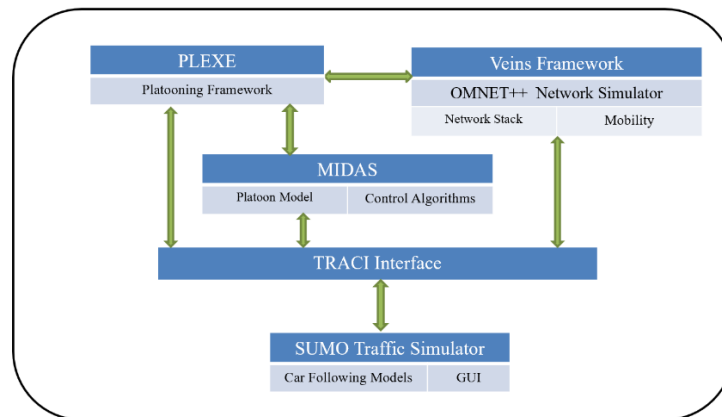


Figure 2.13. MIDAS Simulation Architecture

2.4 Simulation Results

MIDAS AI is implemented using the simulation architecture explained in Figure 2.13. An isolated unsignalized (traffic lights free) intersection control for autonomous vehicular traffic is implemented using SUMO microscopic traffic simulator and MIDAS AV intersection logic described in algorithm 2.3. MIDAS platooning strategy explained in algorithm 2.1 is evaluated using several simulations with varying AV traffic loads and compared to scenarios without any platooning strategy in place. The simulated AV traffic follows improved IDM driving control in the absence of MIDAS platooning strategy and a CACC based longitudinal and lateral control in the presence of MIDAS platooning strategy respectively as described in section 2.2.3.2. The parameter values used in the simulation for both the driving control modes are defined in Tables 2.1 & 2.2 respectively. The simulation study shows that MIDAS platooning has improved the autonomous intersection throughput by 8-10% and time-loss delays by 12-15%. Figures 2.14 & 2.15 show the performance of MIDAS AI for a traffic load of 5400 vehicles per hour.

Table 2.1. Improved Intelligent Driver Model Control Parameters (No Platooning Scenario)

Parameter	Value
Desired speed V_j^{max}	20 m/s
Safe time gap τ	1.0 s
Minimum gap x_0	2 m
Acceleration exponent δ	4
Maximum acceleration A^{max}	0.8 m/s ²
Maximum comfortable deceleration B^{max}	2 m/s ²

Table 2.2. Cooperative Adaptive Cruise Control Model Parameters (Platooning Scenario)

Parameter	Value
Free flow speed difference gain k_1	0.4 s^{-1}
Position difference gain k_2	0.23 s^{-2}
Position difference gain k_3	0.07 s^{-1}
Desired time gap of ACC mode t_{dg}	31.1% - 2.2 s 18.5% - 1.6 s 50.4% - 1.1 s
Inter-platoon constant time gap t_{sg}	57% - 0.6 s 24% - 0.7 s 7% - 0.9 s 12% - 1.1 s
Gain for adjusting time gap k_p	0.45 s^{-1}
Gain for adjusting time gap k_d	0.0125
Maximum acceleration A^{max}	0.8 m/s^2
Maximum comfortable deceleration B^{max}	2 m/s^2
Minimum average gap maintained in a platoon g^{avg}	2 m
Maximum allowed platoon string length l	10

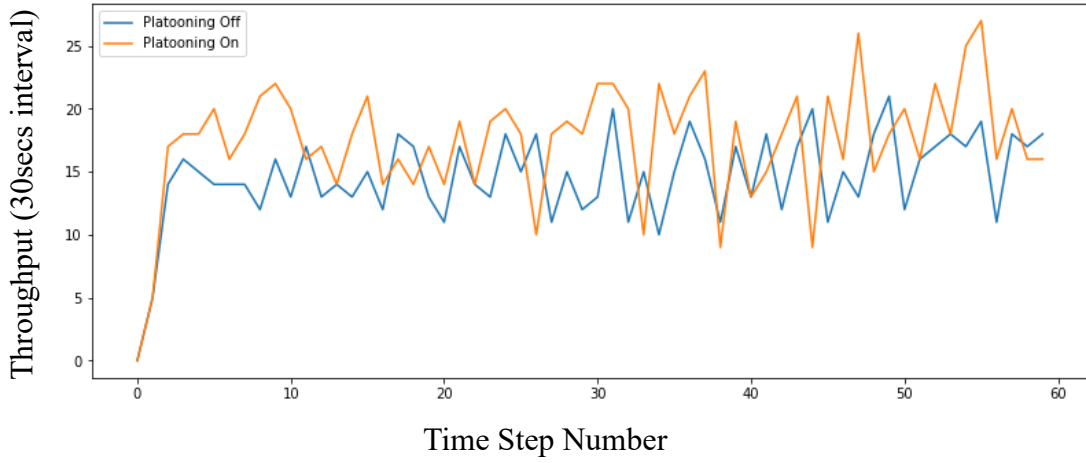


Figure 2.14. MIDAS AI Throughput Performance with & without Platooning Strategy.

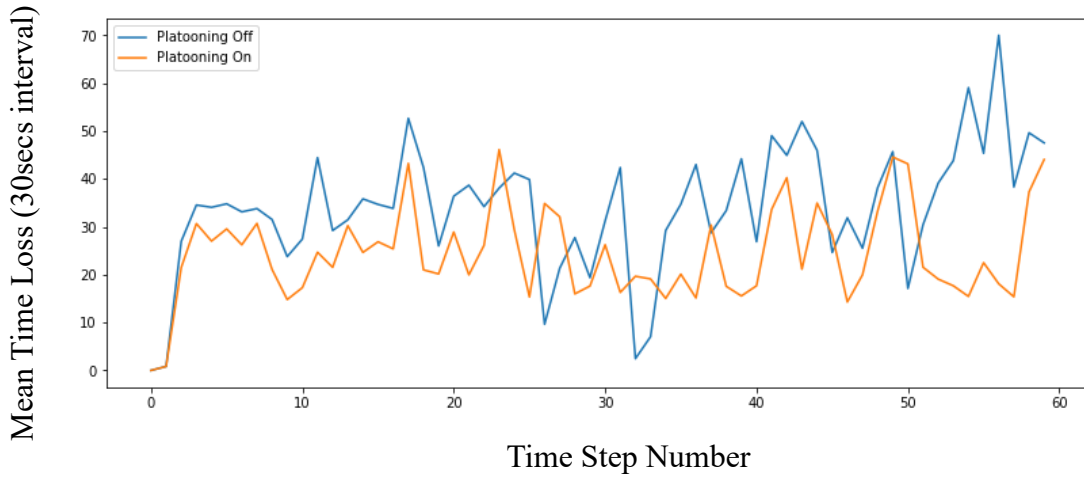


Figure 2.15 MIDAS AI Time-Loss Delay Performance with & without Platooning Strategy.

CHAPTER 3

REAL-TIME PROACTIVE TRAFFIC CONTROL OF AUTONOMOUS VEHICLES THROUGH UNSIGNALIZED INTERSECTION USING DEEP REINFORCEMENT LEARNING

Preview of Contributions

- a) Proposed a real-time proactive traffic control for autonomous vehicles at an unsignalized intersection using deep reinforcement learning.
- b) Proposed an effective representation of traffic state information for autonomous vehicles.
- c) Proposed a deep Q learning architecture using multiple stacks of convolutional neural networks and fully connected layers to estimate Q value function. Also implemented experience replay and target Q network techniques for improved training stability and convergence.

3.1 Introduction to Traffic Signal Control at Intersection Using Reinforcement Learning

Artificial intelligence (AI) is defined as the ability of machines to replicate human intelligence in recognizing patterns and making intelligent decisions in problem-solving environments. AI is an interdisciplinary field that combines concepts from mathematics, statistics, computer science and cognitive science to solve complex problems. AI has been used in a wide range of applications for solving problems in industry and academia. Some of the well-known applications are recommendations in ecommerce to provide personalized shopping experience for customers, developing voice-based assistants using

natural language processing, fraud detection for credit card transactions, facial recognition, automation using robotics in manufacturing, sophisticated medical devices in healthcare and very recently object detection and autopilot enhancements in automotive industry, etc. According to Statista by 2025, the revenue generated by AI applications is expected to reach 126 billion dollars [98] and customers interacting with AI applications would reach 95% [99]. AI is essentially a powerhouse with important sub fields like supervised machine learning, unsupervised machine learning and reinforcement learning. Both machine learning and reinforcement learning share a common sub field called deep learning as shown in figure (3.1).

There have been some significant early efforts [100][101][102] in applying reinforcement learning concepts towards developing an adaptive traffic control system. But these models suffer lack of ability to adapt dynamically to complex traffic situations since the features used to represent the system state were simple traffic metrics like vehicle queue lengths, average throughput of vehicles and average waiting times at the intersection, which are abstractions of real traffic information i.e speed and vehicle position. For example, vehicle queue lengths as a state ignores useful upstream vehicle information which is crucial for proactive traffic control at the intersection and leads to learning suboptimal signal plans. Also due to the limitation of computing power, most of the work was validated using abstract simulation models which lack real-world traffic behavior. In the last decade with the advancements in high-performance computing and development of complex traffic microsimulation models have encouraged researchers to study the implementation of supervised learning techniques in developing better traffic control policies than conventional fixed time traffic signal controls. In [103] authors trained a neural network

(NN) with a large set of generated traffic scenarios that were solved for optimal green times using dynamic programming. Control policies trained for specific traffic scenarios don't adapt well during new traffic situations arise in real-world on a day-to-day basis and require a lot of training data to train such models offline. Similarly, authors in [104] trained a fuzzy NN model to estimate the optimal signal plans for network of intersections. With ever changing traffic conditions and driving technologies there is a need for enhanced traffic control models and [105][106] provides a comprehensive review of self-adaptive traffic control systems and applications of reinforcement learning algorithms in controlling traffic dynamically. Later researchers [107][108][109] have developed and implemented deep reinforcement learning algorithms to achieve human-level control for computers (or agents) in competing with humans, by combining concepts from deep learning and reinforcement learning.

Until recently there hasn't been much research applying deep reinforcement learning in developing adaptive traffic control systems. Notably authors [110] have implemented better state space representation called DTSE by discretizing road segments into fixed length cells to precisely capture the position and speed of vehicles into a matrix. Also, authors have used deep learning architecture to obtain Q value function by training layers of convolutional neural networks. Later [111] improved the training stability of the deep Q network (DQN) proposed in [110] by implementing experience replay mechanism. Although the improved state space representation helps DQN to learn high dimensional features that can predict Q value function better than implementations which use abstract representation of traffic state, it is challenging to obtain accurate position and speed information of human-driven vehicular traffic in conventional setting to gain the merits of

DTSE representation. Later authors [112] implemented a modified DTSE representation to speed up the training process of DQN. Authors discretized cells of varying lengths where cells closer to intersection are shorter compared to the cells that are farther, assuming the dispersion is higher for upstream traffic. But finding an optimal cell length without compromising on the useful state information is difficult and traffic situations with congested upstream lead to DQN learning suboptimal traffic signal policies. Authors in [113] have used real image snapshots of intersection corridor at every agent time step as state input to train layers of convolutional neural networks. But obtaining image snapshots of the intersection corridor and processing in real-time is highly challenging in real-world and image-based state representation models require historical image stack to impute the speed of the vehicles during current iteration. The rectangular snapshots of intersection corridor also include pixels of non-road structures which do not contribute to the model training and increase training complexity. In [114] researchers used image like representation of state space evading the need for real image, but model training still suffers from overloaded state space due to the inclusion of unnecessary information.

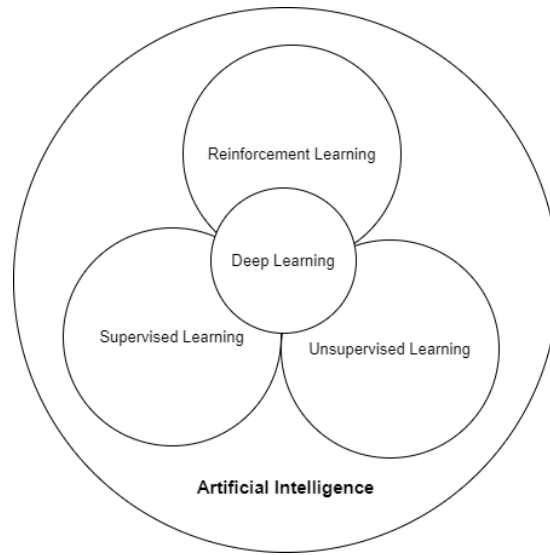


Figure 3.1. Artificial Intelligence Sub Fields and Conceptual Overlap

All the above discussed research work from the literature, related to the implementation of deep reinforcement learning concepts in the development of adaptive traffic control, is majorly focused towards controlling human-driven traffic and obtaining the sophisticated state information as mentioned in theory is difficult in real-world. In this dissertation, a novel adaptive traffic control system called MIDAS RAIC has been proposed to efficiently control autonomous vehicular traffic through an unsignalized intersection in real-time using deep reinforcement learning (DRL). A multi-convolutional neural network followed by a fully connected architecture with experience replay mechanism is used to train the MIDAS RAIC agent in estimating the deep Q network (DQN). To improve the training stability of the agent, a target Q network with soft parameter updates has been implemented. MIDAS RAIC uses DTSE approach to represent the vehicles position, speed, and clearance request status matrices along with the movement's clearance status at the

intersection as the state of the agent. A more detailed model formulation is presented in section 3.1.

3.2 A Comprehensive Overview of Key Concepts in Reinforcement Learning

3.2.1 Markov Decision Process Framework

Typically, in a reinforcement learning process, an agent who is a learner and decision maker is surrounded by an environment and interacts with it, takes actions from available action space. The environment, in return, rewards the agent for its actions and transitions to a new state based on the action taken by the agent. The agent then learns to differentiate good decisions from bad ones based on the reward value it receives. The agent-environment interaction loop is shown in figure (3.2). Reinforcement learning is undeniably the best choice for solving sequential decision-making problems like controlling dynamic traffic.

S = state space; set of possible states an agent can transition within an environment.

U = action space; set of possible actions an agent can take within an environment.

R = Reward space where R_{t+1} is a reward earned by agent for taking action u_t in state s_{t+1} .

T = transition function space that defines the probability to transition from state to state.

π = action policy

$\pi(u|s) = \text{probability of agent taking an action } u \text{ in state } s.$

$\gamma = \text{discount factor}$

$\mathcal{P}_{ss'}^u = \text{probability of agent transitioning to the state } s' \text{ from } s \text{ by taking an action } u.$

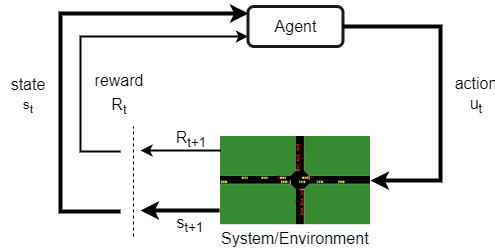


Figure 3.2. Basic Reinforcement Learning Feedback Loop

Reinforcement learning process tries to maximize the total reward received by the agent over a long run. To solve a sequential decision-making problem using RL, one needs to formulate the problem as Markov decision process (MDP) which follows the Markov property defined below.

Markov Property is defined as a memory less property of the system dynamics. It states that the future transitions of the system/ environment depend only on the current state of the system and don't depend on the states in the past. Mathematically shown in (3.1)

$$P[S_{t+1}|S_t] = P[S_{t+1}|S_1, S_2, S_3, \dots, S_t] \quad (3.1)$$

The overall goal of the agent is to learn an optimal action policy π^* by maximizing the total cumulative reward over a long run. Given R_t is the reward received by the agent at time t , the total return G_t received by the agent over time T is defined as

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + \dots + R_T \quad (3.2)$$

In reinforcement learning rewards received after a long time may not be as important as the reward received immediately. So, the equation (3.2) above is modified to incorporate a discount factor $0 < \gamma \leq 1$ such that each reward received into the future is discounted by this factor as shown in equation (3.3).

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \quad (3.3)$$

And an action policy π is defined as a probability distribution over action space and guides agent's choice of action at any given state, also mathematically denoted as

$$\pi(u|s) = P[U_t = u | S_t = s] \quad (3.4)$$

State-value function in MDP is defined as the expected return received by the agent starting from the given state s under an action policy π denoted by $V_\pi(s)$ and expressed as

$$V_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s \right] \quad \forall s \in \mathcal{S} \quad (3.5)$$

and now the optimal policy π^* is expressed as

$$\pi^* = \arg \max_{\pi} V_\pi(s) \quad \forall s \in \mathcal{S} \quad (3.6)$$

State-action-value function in MDP is defined as the expected return received by the agent from state s by taking an action u and then following an action policy π thereafter. It is also known as Q value function and denoted by $Q_\pi(s, u)$

$$Q_\pi(s, u) = \mathbb{E}_\pi[G_t | S_t = s, U_t = u] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, U_t = u \right] \quad \forall s \in \mathcal{S}, u \in \mathcal{U} \quad (3.7)$$

As per the Bellman's expectation equation [115][7], the state-value and action value equations above can be rewritten as follows

$$V_\pi(s) = \sum_{u \in \mathcal{U}} \pi(u|s) \left(R_s^u + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^u V_\pi(s') \right) \quad (3.8)$$

$$Q_\pi(s, u) = R_s^u + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^u \sum_{u' \in \mathcal{U}} \pi(u'|s') Q_\pi(s', u') \quad (3.9)$$

Similarly, the optimal state-value function and state-action value function are expressed using Bellman's optimality [67] as follows

$$V_*(s) = \max_u \left(R_s^u + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^u V_*(s') \right) \quad (3.10)$$

$$Q_*(s, u) = R_s^u + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^u \max_{u'} Q_*(s', u') \quad (3.11)$$

Where $R_s^u = \mathbb{E}[R_{t+1} | S_t = s, U_t = u]$.

3.2.2 Model-Based Reinforcement Learning

In model-based learning it is assumed that we know the transition probabilities of an agent. In model-based learning agent learns an optimal policy using 3 different paradigms in dynamic programming (DP). DP solves a complex problem by breaking it into smaller subproblems recursively by storing the solutions of the subproblem.

3.2.2.1 Policy Evaluation

In policy evaluation agent learns the state value function for a given arbitrary policy π . It starts with an initial guess of value function for all the states $s \in \mathcal{S}$ and iteratively applies Bellman state value equation (3.8) to update the value function of the states until the maximum difference between the value functions of two consecutive iterations is smaller than some small positive delta. Policy evaluation is combined with policy improvement to obtain the optimal policy.

3.2.2.2 Policy Iteration

In policy iteration agent learns an optimal policy by iteratively evaluating a policy π and improving the policy using the Bellman optimality equation (3.10) until previous and current policies are the same. It starts with a random policy and evaluates the policy from the previous iteration using the process described in section 2.2.1.

3.2.2.3 Value Iteration

In value iteration agent learns an optimal policy by iteratively updating the value functions of all the states $s \in \mathcal{S}$ using the Bellman optimality equation (3.11) until the maximum difference between the value functions of consecutive iterations is smaller than a small positive constant. It starts by initializing $V(s) = 0 \forall s \in \mathcal{S}$.

3.2.3 Model-Free Reinforcement learning

In the model-free reinforcement learning we don't assume the transition probabilities of the agent. In other words, agents learn an optimal policy by learning the consequences of its actions through experiences. In many real-world problems, it is difficult to gather knowledge or define the transition probabilities for the model. In such situations,

undoubtedly model-free reinforcement learning shines through. Model free RL can be either a policy-based or value-based learning algorithm. Value-based algorithms find optimal state action value and then determine the optimal action policy from it. Policy-based algorithms don't require optimal value rather they directly learn the best policy. These algorithms use simple look up table updates or a more generalized function approximator.

- a) Online Policy In online policy like SARSA [117] agent takes an action u from state s using $\epsilon - greedy$ policy and transitions to a new state s' . It then updates the state-action value function $Q(s, u)$ using the difference with the target Q function which is calculated by taking an action u' following the same current policy as u . The Q value updates are done using the following equation.

$$Q(s, u) = Q(s, u) + \alpha [R_s^u + \gamma Q(s', u') - Q(s, u)] \quad (3.12)$$

- b) Offline Policy In offline policy like Q-learning [118] agent takes an action u from state s using $\epsilon - greedy$ policy and transitions to a new state s' . It then updates the state-action value function $Q(s, u)$ using the difference with the target Q function which is calculated by taking a greedy action u' . The overview of the Q-learning algorithm is shown in (3.3) and the Q value updates are done using the following equation.

$$Q(s, u) = Q(s, u) + \alpha \left[R_s^u + \gamma \max_{u'} Q(s', u') - Q(s, u) \right] \quad (3.13)$$

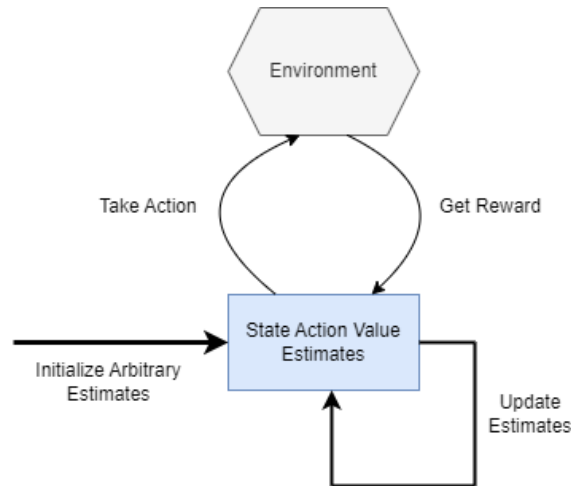


Figure 3.3. Q-Learning Algorithm Overview

As the number of states grows infinitely like in case of traffic control problem, updating Q values using look up tables become intractable. In this dissertation we use a deep Q neural network-based function approximator to estimate the state-action values. Detailed discussion on the DQN architecture and RL problem formulation is presented in the following sections.

3.3 MIDAS RAIC System Architecture

3.3.1 Proposed System Model and Deep-RL Problem Formulation

Notation: -

Z_j^t = set of AVs in movement j at simulation time step t requiring or waiting to have a clearance

H^t = ordered list of AVs that were given clearance by agent at simulation time step t

G^t = ordered list of AVs with clearance implemented by MIDAS RAIC

or clearing the intersection at simulation time step t

sd_{ij}^t = safety distance required for the AV i in movement j to travel before an AV from conflicting movement is released, to clear off the intersection.

\mathcal{T}_{ij}^t = min time required for AV i in movement j to reach the conflict zone.

t_{ij} = block time, max time required to clear off the intersection for AV i in movement j

L_τ = cumulative time loss delay of all AVs existed in the intersection corridor until the agent time step τ

L_{ij}^τ = cumulative time loss delay of AV i in movement j until the agent time step τ

$vStatus$ = attribute of vehicle that tracks the status of an AV and it takes values (0: free, 1: requested clearance, 2: granted clearance)

d_{ij}^t = control point distance from the intersection; min distance need for vehicle i in movement j to safely decelerate to a complete stop.

n_{ij}^t = length of platoon; number of AVs in platoon i in movement j at time t

v_{ij}^t = speed of vehicle i in movement j at simulation time t

v_j^{max} = maximum speed allowed on the movement j

a_{ij}^{max} = maximum acceleration of vehicle i in movement j

b_{ij}^{avg} = average or comfortable deceleration of vehicle i in movement j

p_{ij}^t = position of vehicle i in movement j at time simulation time t

f_{ij}^τ = requested clearance status of vehicle i in movement j at agent time step τ

s_t = state of the environment at time t

u_τ = action taken by agent at agent time step τ

R_τ = reward received by the agent at the agent time step τ

L_τ = cumulative time loss of all the vehicles in the intersection until agent time step τ

Agent's Environment: The RL environment is defined as a dynamic system that is controlled using a learned agent by maximizing a long-term reward metric. An RL agent lives within the environment and interacts with it by performing some actions but can't influence the governing rules of the environment. Agent's environment transitions to a new state every time the agent performs an action and sends the agent to the new state. Environment rewards the agent with a value which acts as feedback to the agent whether the action taken was good or bad. The high-level agent-environment relation is described in Figure 3.1. In MIDAS RAIC an unsignalized intersection with fully autonomous vehicular traffic is considered as the environment and is simulated in SUMO microscopic traffic simulator with MIDAS AI intersection control logic.

Environment State Space: Efficient state space representation plays the major role in determining how well a RL agent learns complex patterns to operate in a dynamic environment. The state representation problem in RL is similar to feature engineering or

feature selection process in supervised or unsupervised machine learning. The goal of identifying the right feature space and extracting the high-dimensional information of environment is to strike a balance between the learning time required for the agent and its ability to learn complex traffic patterns. There were several state representations for traffic control problems using RL, considered in the literature. However, all the representations were abstraction of raw traffic data and ignore useful traffic information leading to suboptimal traffic control. For example, considering vehicle queue lengths at the intersection as state information would limit the agent's ability to take actions based only on the current vehicle queues formed at the intersection, ignoring information about upstream traffic approaching the intersection. Similarly average vehicle delays and wait times are abstraction of historical traffic data and fail to represent the real-time dynamics of traffic at the intersection corridor. Later some researchers attempted to impute the state space information using image processing and CNN architecture. Although image representation provides high density spatial information of vehicles and guarantees better agent performance compared to the abstract representations, not all pixels in the image are relevant to traffic state representation as the image pixels also include side roads, buildings and other varying noisy data that is irrelevant to making traffic control decisions. Also, image representation-based RL methodologies require historical image stacking for CNNs to impute the vehicles speed based on sequential observations, which increases the training time of the agent significantly. In this chapter an efficient state space representation of raw traffic data like precise AV position and speed, AV clearance status (denoting if AV has arrived its control point or waiting for clearance decision) and information about current clearing movements at the intersection.

The state space is represented using positional matrix \mathbf{P} , normalized speed matrix \mathbf{V} , requested clearance status matrix \mathbf{F} of all vehicles present in the intersection corridor along with a Boolean vector \mathbf{M}_{status} representing the current clearing vehicle movements at the intersection. The spatial representation of AVs at the intersection is inspired from the discrete traffic state encoding [DTSE] described in [110]. In [110] authors implemented DTSE for human-driven traffic where extracting precise spatial information of vehicles is highly challenging and expensive. W.L.O.G, in this chapter it is assumed that traffic approaching the intersection controlled by MIDAS RAIC is 100% fully autonomous. Each approaching lane of length l at the intersection is discretized into cells of length c . The positional information of AVs on east-west bound is encoded in the matrix \mathbf{P}_E , where a value 1 is set to a cell entry in \mathbf{P}_E if a vehicle is present in that cell, otherwise 0. Similarly normalized speed of AVs on east-west bound using max link speed V_E^{max} are registered in the corresponding entries of matrix \mathbf{V}_E . Also, an entry in matrix \mathbf{F}_E is set to 1 if the vehicle in the corresponding cell is given a clearance already, otherwise 0. The positional and speed information of vehicles on all bounds are given by \mathbf{P} and \mathbf{V} as shown in (3.15) and (3.16) respectively. At every agent time-step MIDAS RAIC extracts the current state information of the intersection using the logic described in algorithm (3.1) and a demonstration is shown in (3.18) -(3.21) for the intersection snapshot shown in figure (3.4)

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_E \\ \mathbf{P}_N \\ \mathbf{P}_W \\ \mathbf{P}_S \end{bmatrix} \quad (3.15)$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{V}_E \\ \mathbf{V}_N \\ \mathbf{V}_W \\ \mathbf{V}_S \end{bmatrix} \quad (3.16)$$

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_E \\ \mathbf{F}_N \\ \mathbf{F}_W \\ \mathbf{F}_S \end{bmatrix} \quad (3.17)$$

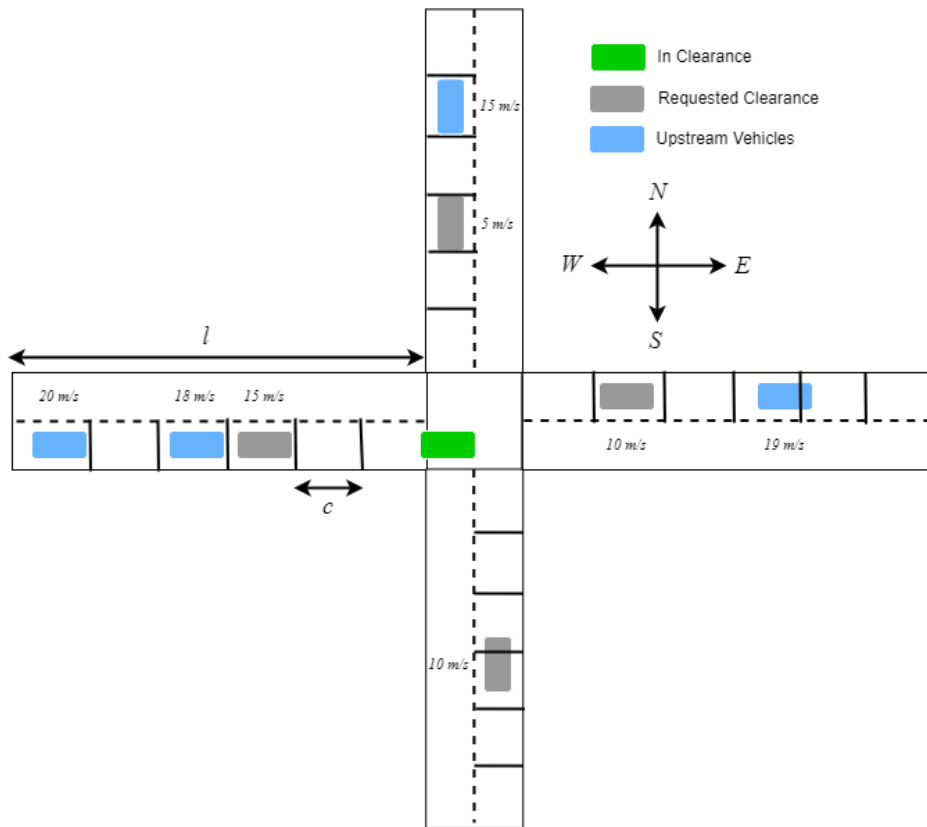


Figure 3.4. Illustration of Vehicles State Representation for a 4-Legged Unsignalized Intersection

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \quad (3.18)$$

$$\mathbf{V} = \begin{bmatrix} 0 & 0 & 0.75 & 0.9 & 0 & 1 \\ 0 & 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0.95 & 0 & 0 \\ 0 & 0 & 0.25 & 0 & 0.75 & 0 \end{bmatrix} \quad (3.19)$$

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (3.20)$$

$$\mathbf{M}_{status} = [1 \ 0] \quad (3.21)$$

$$d_{ij}^t = \frac{(v_{ij}^t)^2}{2b_{ij}^{avg}} + \sum_{i=0}^{i-1} n_{ij}^t \quad (3.22)$$

$$\mathcal{J}_{ij}^t = \frac{-v_{ij}^t \pm \sqrt{(v_{ij}^t)^2 - 2 * a_{ij}^{max} * p_{ij}^t}}{a_{ij}^{max}} \quad (3.23)$$

$$t_{ij} = \frac{sd_{ij}^t}{v_{ij}^t} \quad (3.24)$$

$$f_{ij}^\tau = \begin{cases} 1 & \text{if } vStatus(veh(i,j)) = 1 \text{ or } p_{ij}^{m(\tau)} \leq d_{ij}^{m(\tau)} \text{ or } v_{ij}^{m(\tau)} < 0.5 \\ 0, & \text{otherwise} \end{cases} \quad (3.25)$$

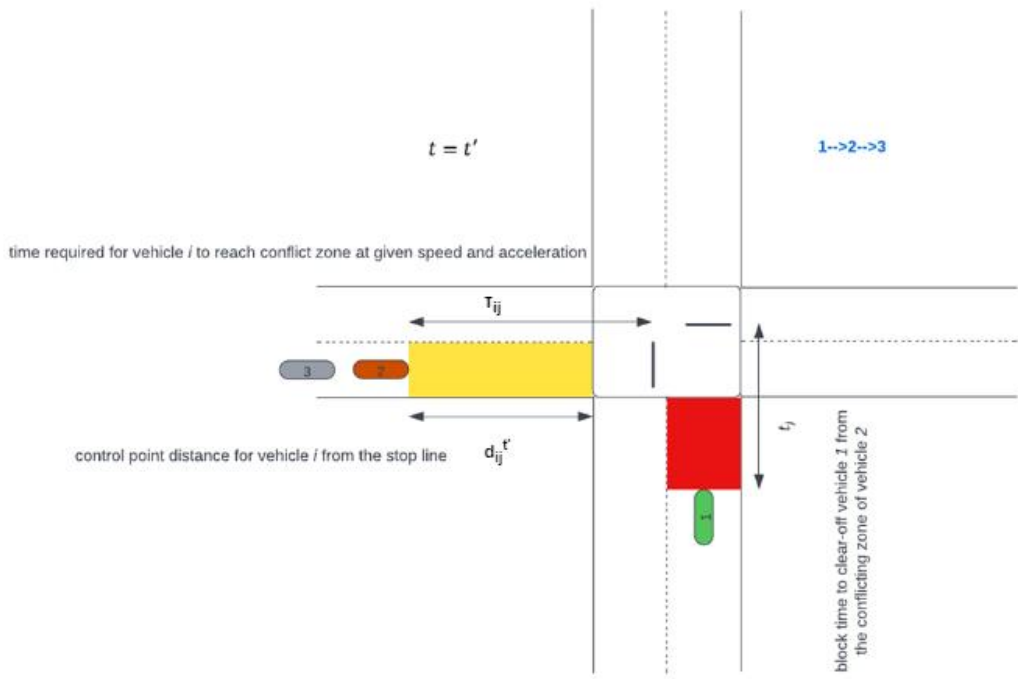


Figure 3.5. Unsignalized Intersection Control Logic for Autonomous Vehicles

Get position, speed, and acceleration information, etc. of all the vehicles that require clearance at the intersection denoted by Z_j^t using SUMO Traci API at agent time step τ or simulation step $m(\tau)$.

Step1: Initialize $\mathbf{P}, \mathbf{V}, \mathbf{F}$ matrices to all zeros.

Step2: for each movement $j \in \mathbf{J}$:

for each $veh(i, j) \in Z_j^t$:

→ set entry $\left[\frac{p_{ij}^{m(\tau)}}{c} \right]$ in matrix \mathbf{P}_j to 1

→ set entry $\left[\frac{p_{ij}^{m(\tau)}}{c} \right]$ in matrix \mathbf{V}_j to $\frac{v_{ij}^{m(\tau)}}{v_j^{max}}$

→ using eqns (3.22) and (3.25) evaluate f_{ij}^t value

→ set entry $\left[\frac{p_{ij}^{m(\tau)}}{c} \right]$ in matrix \mathbf{F}_j to f_{ij}^t

→ update vehicle status: $vStatus(veh(i, j)) \leftarrow f_{ij}^t$

Step3: call intersection control to evaluate current clearing movement

$$\rightarrow \mathbf{M}_{status} = \begin{cases} [1 \ 0] & \text{if clearing vehicles at time } t \text{ are from east and west bounds} \\ [0 \ 1], & \text{otherwise} \end{cases}$$

Algorithm 3.1. Autonomous Vehicular Traffic State Information Extraction at Agent-step τ

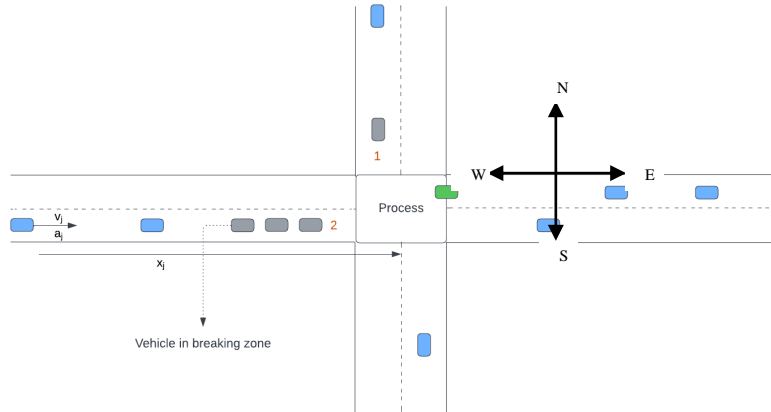


Figure 3.6.a. Illustration of 4-legged Unsignalized Intersection

Agent Action Space: At every agent time-step (also called action step), agent takes an action by observing the state of the defined traffic intersection environment. It is important to note the difference in agent time-step and traffic simulation time-step (simply called time-step or time). At any given agent time-step, for a simple 4-leg unsignalized intersection considered in this chapter as shown in figure (3.6.a), agent has 6 different choices to choose from

- a) Give clearance to the first vehicle on the east-west bound that has requested clearance.
- b) Give clearance to the first vehicle on west-east bound that has requested clearance.
- c) Give clearance to the first vehicles on both east-west and west-east bounds that have requested clearance.
- d) Give clearance to the first vehicle on the north-south bound that has requested clearance.
- e) Give clearance to the first vehicle on the south-north bound that has requested clearance.
- f) Give clearance to the first vehicles on both north-south and south-north bounds that have requested clearance.

Agent chooses an action by following an ϵ -greedy policy as illustrated in Figure 3.6.b. At every agent action step with a probability ϵ agent randomly gives clearance to the first vehicle in any movement bound and with a probability $1-\epsilon$ agent greedily selects the action that leads to maximum value of being in the current state.

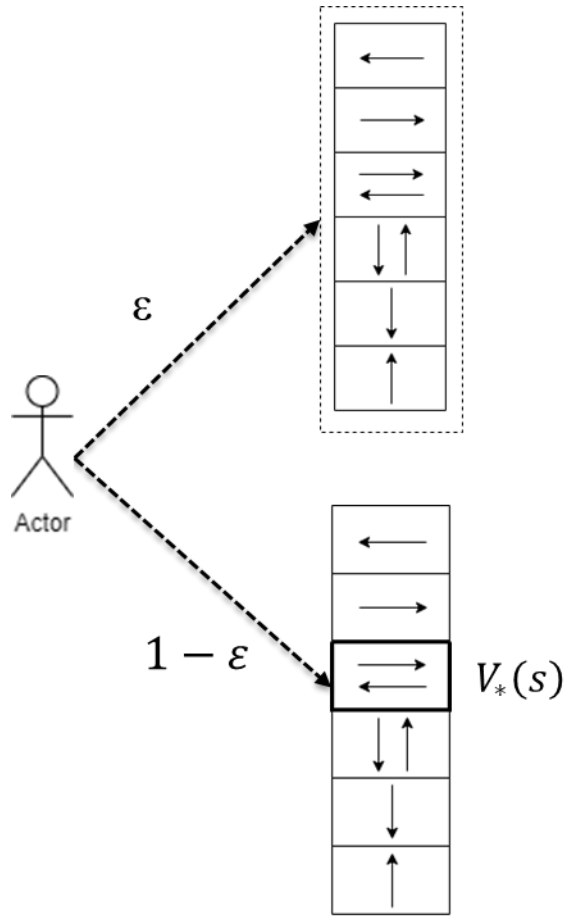


Figure 3.6.b. Agent's ϵ -greedy Policy.

A vehicle is said to have requested clearance if the vehicle has reached its control decision point, or its speed has been reduced below a threshold as defined in (3.22). This agent action representation explores the solution space more efficiently than action space representations used in literature [110][111]. Simple two-dimensional action space representation of vehicle movements from east to west or west to east and north to south or south to north isn't flexible in allowing a conflicting vehicle movement between two complementary vehicle movements. Also note that in certain states not all actions listed above are available to agent to choose from, as some actions are invalid to take. For example, as per our action space representation an agent can only give clearance to the first

AV in a certain movement only if there are any AVs in that movement with a requested clearance status as it's too early for AVs upstream to have clearance. In such action steps there is a possibility of agent taking invalid actions due exploitation or epsilon-greedy exploration criteria (see section 3.2 for more details). Later in this chapter an invalid action masking strategy is implemented to prevent the agent from taking unavailable actions in any given state.

The elapsed traffic simulation time between consecutive state transition is defined as the length of agent time-step. As mentioned before the length of agent time-step varies with state and action pair and the agent is notified of beginning of new agent time-step using the procedure illustrated in Figure 3.6.c and corresponding pseudo-code is described in algorithm (3.2)

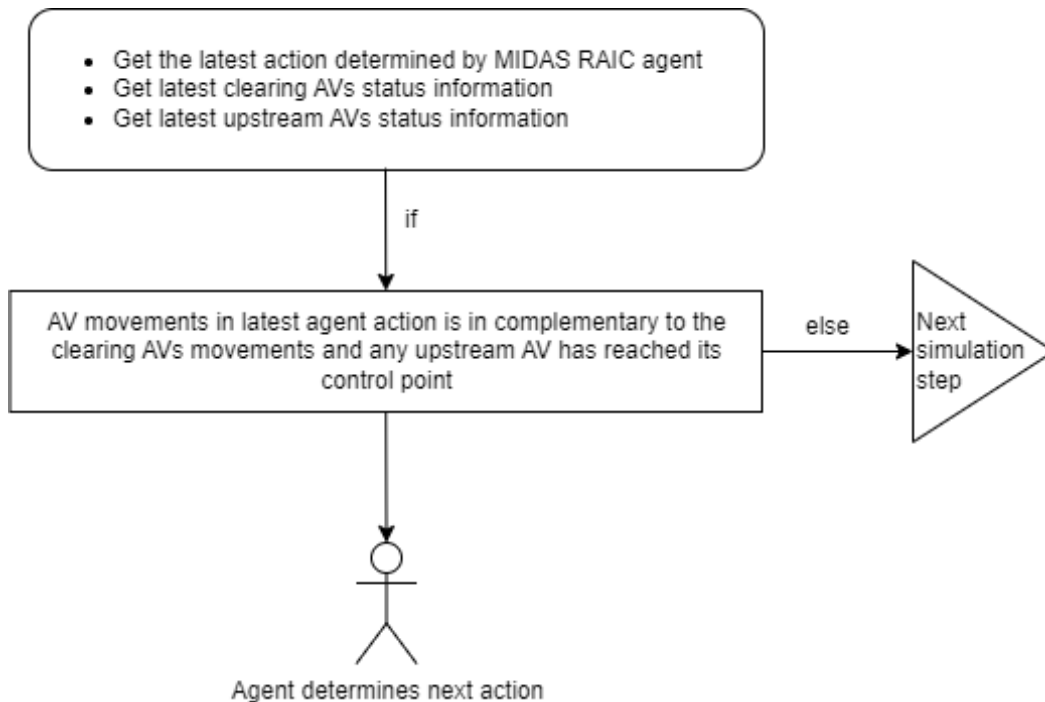


Figure 3.6.c. Illustration of new agent action step

Step1: \rightarrow get G^t and Z_j^t for current simulation time step t
 \rightarrow get $vStatus(veh(i, j)) \forall veh(i, j) \in Z_j^t, \forall j \in J$

Step2: if $\tau = 0$ and $\exists_j \in J, veh(i, j) \in Z_j^t | vStatus(veh(i, j)) = 1$:
 \rightarrow return *TRUE*
else:
 \rightarrow get latest agent action u_τ at agent time step τ or simulation time step $t = m(\tau)$

Step3: if $\forall veh(i, j) \in u_\tau | veh(i, j) \in G^t$ and $\exists_j \in J, veh(i, j) \in Z_j^t | vStatus(veh(i, j)) = 1$
 \rightarrow return *TRUE*

Step4: \rightarrow return *FALSE*

Algorithm 3.2. Evaluation of the Start of New Agent Time Step

Agent Reward: In RL an agent is encouraged or discouraged from taking certain actions by rewarding the agent with a positive or negative value. A good action taken by an agent is rewarded with a positive value and a bad action is rewarded with a negative value. Hence an agent is trained to maximize the total cumulative reward over a long run. At every agent time-step the agent observes the current traffic state of the intersection environment and takes an action that transitions the agent to a new state. A reward is feedback given to the agent to understand the consequences of its past actions and improve them in the future. So, defining an efficient reward function is crucial for the learning process of an agent. In literature several different reward functions have been implemented, for example, change in queue lengths, waiting times, etc. To calculate some of these rewards it requires some infrastructural assumptions for human-driven traffic scenarios like sensors and pointed cameras; also, they don't optimize overall traffic congestion. In this chapter the goal of the agent is to minimize the total cumulative time loss delay due to the movement conflicts at an unsignalized autonomous intersection. To achieve such an objective, a change in the

cumulative time-loss delay at consecutive agent time-steps is defined as the reward function of the agent. The reward function is shown in equation (3.27) where L_τ and $L_{\tau-1}$ are total cumulative time-loss delay which is sum of cumulative time-loss delay of all the AVs present until agent time-step τ and $\tau - 1$ respectively. Cumulative time-loss delay L_{ij}^τ of an AV i in movement j at agent time step τ is defined in equation (3.29). Note an agent time-step τ can be easily mapped to simulation time-step t when needed and for convenience the mapping is denoted as (3.26).

$$t = m(\tau) \quad (3.26)$$

$$R_\tau = L_{\tau-1} - L_\tau \quad (3.27)$$

$$L_\tau = \sum_{\forall j} \sum_{\forall i \in (Z_j^{m(\tau)} \cup u_{\tau-1})} L_{ij}^\tau \quad (3.28)$$

$$L_{ij}^\tau = \left\{ L_{ij}^{\tau-1} + \sum_{t=m(\tau-1)}^{m(\tau)} \left(1 - \frac{v_{ij}^t}{v_j^{max}} \right) \right\} \quad (3.29)$$

Transition function: The state transition of agent environment from state s_t to state s_{t+1} is implemented through a modified version of unsignalized autonomous intersection control (IC) logic originally developed in chapter 2 as shown in algorithm (3.3), which translates into microscopic AV movements simulated via SUMO, and triggered in response to an agent's action u_t .

Step1: Given the updated $H^t, G^t, Z_j^t \forall j \in \mathbf{J}$ at simulation time t

- set $slow_down_flag = False$
- calculate the block time required $t_{ij} \forall veh(i, j) \in G^t$ using eqn (3.24)

Step2: for $veh(i, j) \in G^t$:

- if $sd_{ij}^t \leq 0$:
 - remove $veh(i, j)$ from G^t

Step3: for $veh(i, j) \in H^t$:

- if $slow_down_flag = True$:
 - slow down the $veh(i, j)$
- else:
 - Calculate T_{ij}^t using eqn (3.23)
 - if $|G^t| = 0$ or $T_{ij}^t > t_{i'j'}$ $\forall veh(i', j') \in G^t$:
 - give clearance to the $veh(i, j)$
 - add $veh(i, j)$ to G^t
 - update vehicle status: $vStatus(veh(i, j)) \leftarrow 2$
 - remove $veh(i, j)$ from H^t
 - also calculate the block time required t_{ij}
 - else:
 - set $slow_down_flag = True$
 - slow down the $veh(i, j)$

Step5: for each $j \in \mathbf{J}$:

- for $veh(i, j) \in Z_j^t$:
 - if $p_{ij}^t \leq d_{ij}^t$:
 - slow down the vehicle v_{ij}
 - update vehicle status: $vStatus(veh(i, j)) \leftarrow 1$

Algorithm 3.3. Unsignalized (Traffic-Lights-Free) Intersection Control Logic for AV Traffic

3.3.2 DQN Algorithm for Controlling Unsignalized Autonomous Intersection

Notation: -

B = training sample batch

θ = parameter of predicted Q network

θ' = parameter of target Q network

$L(\theta)$ = training error of the agent calculated for DQN parameter θ

s_t^k = environment state in training sample k at time t

u_t^k = action taken by agent in training sample k at time t

R_t^k = reward received by agent in training sample k at time t

$Q(s_t^k, u_t^k; \theta)$ = predicted Q value for state s_t^k , action u_t^k using DQN parameter θ

In this dissertation, an off-policy based reinforcement-learning technique called Q-learning is implemented. Q-learning uses temporal-difference to estimate $Q^*(s, a)$, which is the expected value of cumulative discounted reward that can be earned by choosing action a while in state s and then taking actions according to the optimal policy. In temporal difference learning agent learns the Q-values by interacting with the environment through episodes with no prior knowledge of environment. Maintaining a Q-table with all possible state action pairs and updating the values in the table as the agent explores according to a greedy ϵ policy is a naïve approach to Q-learning. As the number of state and action pairs increase Q-table approach becomes computationally intractable to build a table. In the case of traffic control problem defined in section 3.1, even though the number of possible actions is finite, the state space is infinite due to the random information arriving every second. To overcome the curse of dimensionality of Q-table approach that is limited to deterministic number of state-action pairs, in this research a Q-function with a parameter θ is learned by the agent by training a deep neural network (DNN) using episodes of

training data. The learned Q-function maps the state to Q-values of all actions that can be taken from that state.

The DQN architecture used in MIDAS RAIC has two neural networks, the Q network which predicts the Q-values of actions for a given state and the target network which is identical to the Q network and predicts the target Q value and a component to train Q network called experience replay. The high-level overview of the implemented DQN architecture is shown in figure (3.9). The experience replay component interacts with the environment to generate experience data samples to train the Q network by storing the experiences in a replay buffer M , which replaces oldest experiences with newer experience when it reaches its capacity. As shown in figure (3.7) & (3.8) experience replay selects a random action using ϵ -greedy approach from the current agent state and forwards the action to MIDAS autonomous & unsignalized IC for validation and implementation as described in algorithm (3.3) and moves to next state while receiving a reward. Experience replay saves this experience as a training data sample in the replay memory M . After every experience MIDAS RAIC randomly samples a mini batch B of training data from replay memory buffer M so that the training data contains a diverse mix of old and new data samples for a stable training of the Q network. This batch of training samples are fed to both the Q network and target network. The Q network takes the current state and action taken from a training sample and predicts the Q value for the experience. Similarly target network takes the next state from the training sample to predict the best Q value out of all actions possible in that state and we call this as target Q value for the considered experience.

The objective of MIDAS RAIC DQN is to minimize the total estimated error in the training batch B , denoted by $L(\theta)$ and expressed as shown in equation (3.30) by optimizing the Q network weights denoted by parameter θ . The loss function $L(\theta)$ is represented as the sum of squared error of the target Q value and predicted Q value of all samples in the training batch. MIDAS RAIC DQN uses RMSprop [116] to update the network parameters, which is a stochastic gradient-based optimization technique that uses an adaptive learning rate that is normalized using a moving average of squared gradients. Similarly, the target Q network parameters θ' are updated every iteration using a soft update method shown in (3.31).

MIDAS RAIC DQN architecture uses 3 convolutional neural networks with 2 convolutional layers (CNN), 2 fully connected layers (FC) and 1 output layer. As shown in figure (3.9) input positional matrix \mathbf{P} is fed to a stacked sub-network where the first convolution layer convolves the input with 1 zero-padding, with 16 filters of size 4×4 and with stride 2. The convolution process of the first layer uses a rectifier linear unit (ReLU) activation function. The second layer convolves the first layer output with 32 filters of size 2×2 , with stride 1 and applies ReLU. The normalized speed matrix \mathbf{V} , and requested clearance status matrix \mathbf{F} are fed to 2nd and 3rd subnetworks respectively for convolution using similar 2-layer structure and hyperparameters but different filter weights. The outputs from 2nd layer of the 3 subnetworks are flattened and concatenated with the Boolean vector \mathbf{M}_{status} representing the status of current clearing movements at the intersection, into a single input vector that is fed to a fully connected layer with 128 neurons and with ReLU activation function followed by another fully connected layer with 64 neurons and with ReLU activation function followed by the final fully connected output

layer that outputs the Q values of all possible actions from the input state as shown in figure (3.9).

$$L(\theta) = \sum_{\forall k \in B} \left(\underbrace{\left(R_{t+1}^k + \gamma \max_{u_{t+1}^k} Q(s_{t+1}^k, u_{t+1}^k; \theta') \right)}_{\text{Target } Q} - \underbrace{Q(s_t^k, u_t^k; \theta)}_{\text{Predicted } Q} \right)^2 \quad (3.30)$$

$$\theta' = \alpha \theta' + (1 - \alpha) \theta \quad (3.31)$$

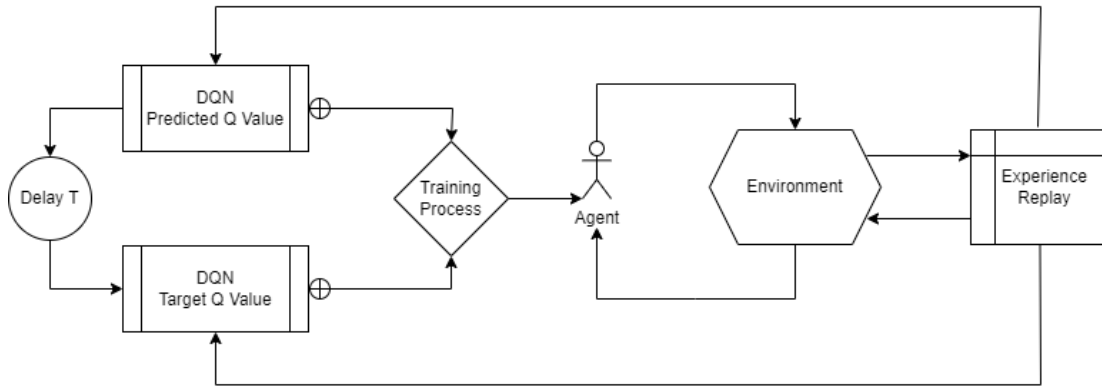


Figure 3.7. Illustration of The Training Procedure for MIDAS RAIC

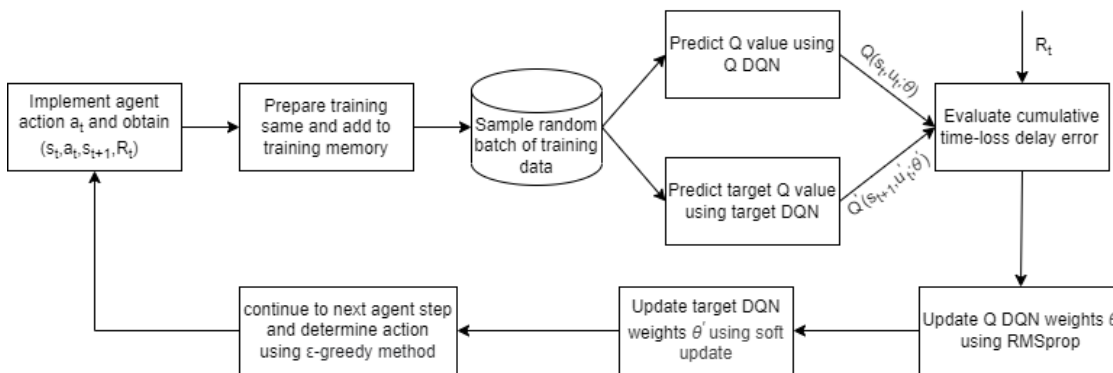


Figure 3.8. Illustration of Training with Experience Replay for MIDAS RAIC

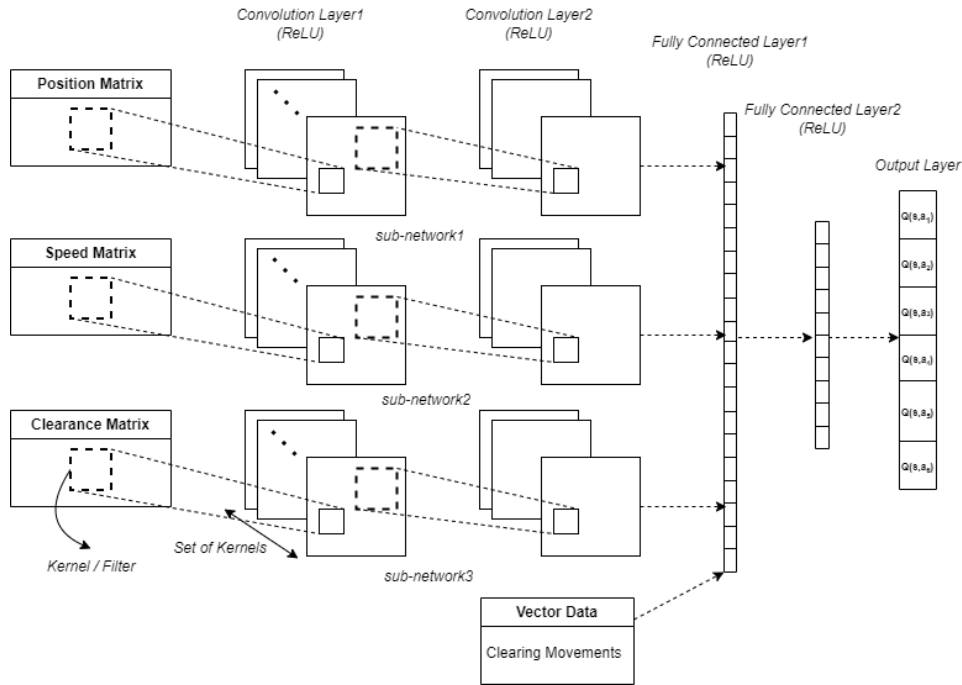


Figure 3.9. MIDAS RAIC DQN Architecture Overview

3.4 MIDAS RAIC System Simulation and Evaluation

MIDAS RAIC DQN is trained for 1200 episodes, where an episode is defined for a length of 4500 simulation seconds in SUMO. At the end of every episode SUMO simulation is reset and at the beginning of every episode a new state is initialized by restarting the SUMO simulation which runs for another 4500 seconds. The complete DQN training algorithm is summarized in the pseudo-code (3.4). Agent takes several action steps during an episode and at every agent action step, agent takes an action using the $\epsilon - greedy$ approach and receives a reward by transitioning to a new state. At the end of every agent action step MIDAS RAIC records the agent's experience and adds it into the memory buffer for training the DQN network. The replay memory buffer is finite in space with $|M|^{min} =$

10000, $|M|^{max} = 100000$. With the $\epsilon - greedy$ approach agent tends to explore less and exploit more as training progresses through the episodes.

The MIDAS RAIC agent is trained on a simple 4-leg unsignalized intersection shown in figure (3.6.a) and simulated using SUMO traffic simulator. During every episode of DQN training MIDAS RAIC simulates intersection corridor with a traffic load of 4800 vehicles per hour for 4500 simulation seconds in SUMO. The detailed traffic distribution by approach is shown in table (3.1) and the corresponding training parameters are shown in table (3.2).

Table 3.1. Distribution of Traffic on Intersection Approaches.

Traffic Approach	Veh/hr
East to West	1200
North to South	900
West to East	1800
South to North	900

Table 3.2. Description of MIDAS RAIC DQN Training Parameters.

Parameter	Value
Total training episodes N	1200
Episode length T	4500 s
Discount factor γ	0.95
RMSprop learning parameter δ	0.0002
Target network update parameter α	0.001
Min replay memory size $ M ^{min}$	10000
Max replay memory size $ M ^{max}$	100000
Mini-batch training size $ B $	32
Cell length c	5 meters

Step1: → initialize DQN network with parameters θ
→ initialize target DQN network with parameters θ'
→ initialize greedy policy parameter ϵ ; discount parameter γ ; target network soft update parameter α ; total number of episodes N ; episode length $T = 4500$ secs; replay memory M , $|M|^{min}$, $|M|^{max}$; training mini batch size $|B|$, RMSprop learning rate δ
→ initialize episode = 1

Step2: → initialize $t = 1$
→ start SUMO simulation of unsignalized intersection
→ initialize agent action step $\tau = 0$

Step3: if new agent action step begins; start(u_{t+1})(determine using algorithm (3.2)):
→ $\tau = \tau + 1$
→ agent observes current state information s_τ of intersection using algorithm (3.1)
→ agent takes action $u_\tau = \arg \max_{u \in U} Q(s_\tau, u; \theta)$ with probability $1 - \epsilon$ and randomly takes an action $u_\tau \in U$ with probability ϵ
→ $\forall \text{veh}(i, j) \in u_\tau$ add $\text{veh}(i, j)$ to H^t

Step4: if $t = \text{end}(u_\tau)$:
→ calculate reward $R_{\tau+1}$ received by agent using equation (3.27).
→ observe new transitioned state $s_{\tau+1}$
→ prepare training sample using experience $(s_\tau, u_\tau, R_{\tau+1}, s_{\tau+1})$
if $|M| < |M|^{max}$:
→ add new training sample to replay memory M
else:
→ delete oldest sample from replay memory M
→ add new training sample to replay memory M
if $|M| > |M|^{min}$:
→ randomly sample mini batch training data from M to train DQN network
→ update DQN network parameter θ using RMSprop and target DQN parameter θ' using equation (3.31)

Step5: → transition simulation to new time step $t = t + 1$ using unsignalized intersection control logic (3.3)

Step6: if $t < T$:
→ $t = t + 1$
→ go to step3

Step7: if episode $< N$:
→ $\epsilon = 1 - \frac{\text{episode}}{N}$
→ episode = episode + 1
→ go to step2
else:
→ terminate

Algorithm 3.4. MIDAS RAIC DQN Agent Training

The objective of MIDAS RAIC is to maximize the total cumulative reward received by the agent, where the reward is defined as change in cumulative time loss delay. MIDAS RAIC training performance can be seen in figure (3.10), which shows the average reward received by the agent during an episode. As the training progresses forward the variance in the average reward received by the agent decreases with episodes and converges to an optimal action policy. Similarly, the performance of MIDAS RAIC agent with respect to the average time loss delay of vehicles during the training period can be seen in the figure (3.11). In the initial stages of the training the variance in average reward received by the agent is high due to the agent's tendency to explore more by taking actions randomly as per the $\epsilon - greedy$ policy leading to lot of delays. Later as the training progresses agent gets better with learning the Q value function and begins to exploit more. The improved agent's performance to the end suggests that the agent's rate of exploration has been decreased and the training has converged to a good action-policy.

Trained MIDAS RAIC system performance has been tested against exact DP based autonomous intersection control system proposed in chapter 2. i.e MIDAS AI. Comparison study in figure (3.12) and table 3.3. shows that MIDAS RAIC performed as good as the MIDAS AI. The average time loss delay for trained MIDAS RAIC is within the 5-10% of the average time loss delay achieved by MIDAS AI. Also, table 3.4. shows that MIDAS RAIC performance is real time and preferred to MIDAS AI since the real time benefits wear out with increasing traffic load.

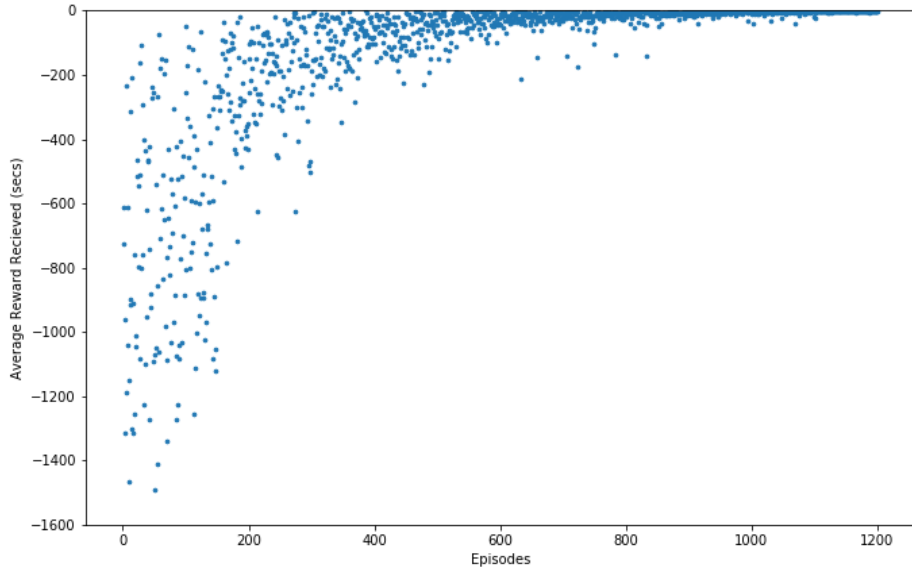


Figure 3.10. Training Performance of MIDAS RAIC in Earning Average Reward.

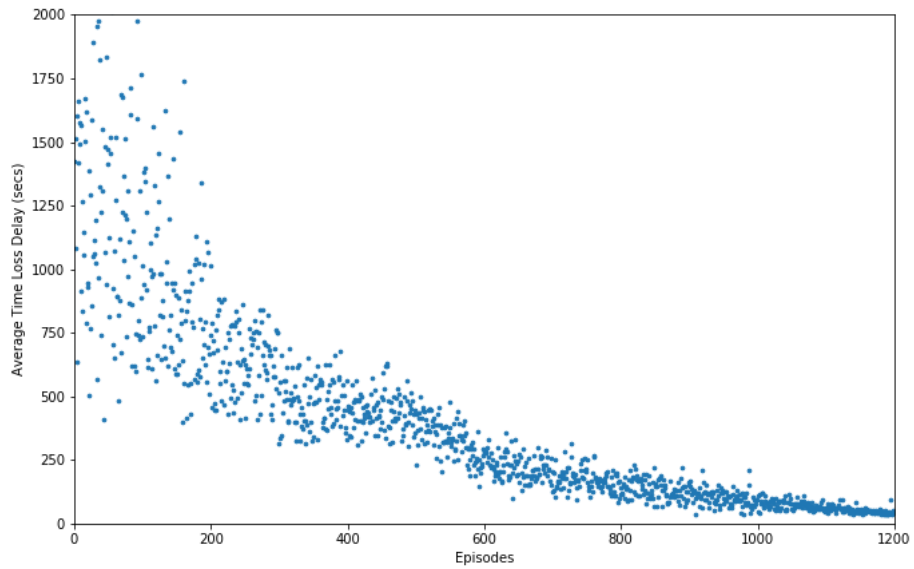


Figure 3.11. Training Performance of MIDAS RAIC in Minimizing Avg. Time Loss Delay of Vehicles.

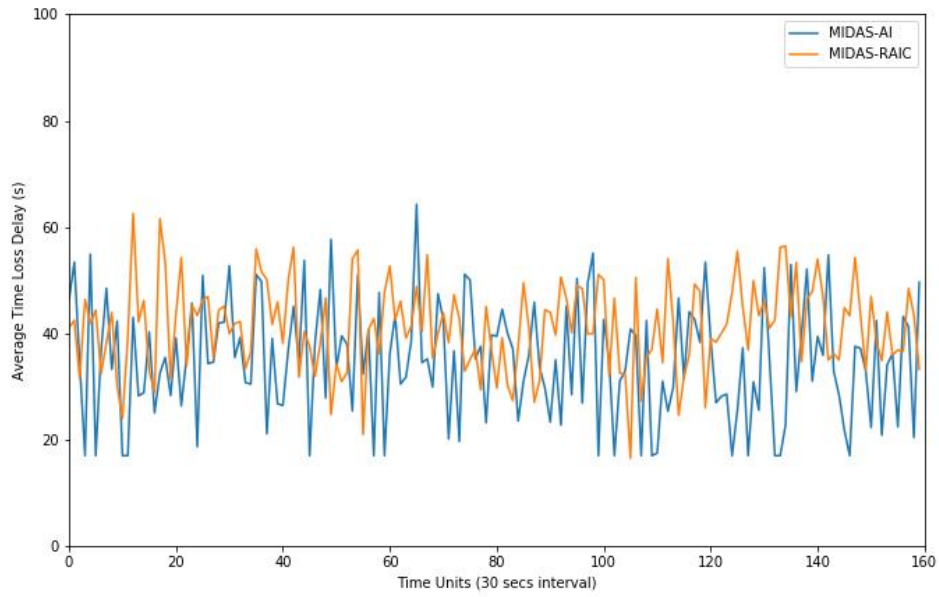


Figure 3.12. Comparison of Average Time Loss Delay in MIDAS RAIC and MIDAS AI.

Table 3.3. Performance Comparison Between MIDAS RAIC and MIDAS AI.

Control Policy	Time-Loss Delay (mean)(secs)	Time-Loss Delay (std)(secs)
MIDAS-AI	35.7	10.9
MIDAS-RAIC	39.25	8.27

Table 3.4 Run Time Comparison Between MIDAS RAIC and MIDAS AI

Traffic Flow (vehs/hr)	MIDAS-AI (solve time)	MIDAS-RAIC-trained (prediction time)
3600	<1s	<0.1s
4800	<1s	<0.1s
5400	1-3 s	<0.1s
5900	1-4s	<0.1s
6800	1-5s	<0.1s

CHAPTER 4

CONCLUSIONS AND FUTURE RESEARCH

4.1 Dissertation Contributions and Conclusions

In summary, this dissertation work develops a cyber-physical system called MIDAS, for real-time proactive traffic control and management of mixed fleet of vehicles with various levels of autonomy. The main contributions of this dissertation work are development and implementation of MIDAS framework in three different vehicular traffic environments. 1) Optimal control of mixed traffic of connected and non-connected vehicles through two closely spaced intersections formally known as diamond interchange, in human-driven environment 2) Optimal control of fully autonomous vehicular traffic and their platoon management through an unsignalized intersection 3) Real-time proactive control of full autonomous vehicles through an intersection using deep reinforcement learning techniques.

Chapter 1 of this dissertation introduces a new proactive traffic control system MIDAS to manage the movements of mixed traffic that consists of GPS-enabled connected vehicles and non-connected human driven vehicles, in almost real-time. MIDAS uses a forward recursive dynamic programming approach to set durations of green times in a cycle-free scheme for controlling potentially highly fluctuating demands, as one may expect at diamond interchanges with complicated traffic movements. MIDAS has been implemented in C++ and was evaluated using a calibrated VISSIM microsimulation model of a DI (on I-17/19th Ave., Phoenix, AZ). The evaluation showed that MIDAS outperforms OFTC and RHODES signal control strategies for common evaluation metrics: average queue delays, average queue lengths and vehicle stops. Further analysis showed that when market

penetration rates of GPS-enabled connected vehicles increase and more such vehicles are managed by MIDAS then benefits improve, as would be expected. This study was partially supported by the National Science Foundation (NSF) Award 166367. The views and conclusions contained in this dissertation work are those of the authors and not NSF.

Later in chapter 2 of this dissertation, a new proactive autonomous intersection control and management system has been developed. Unlike conventional traffic lights system to control human-driven traffic movements at the intersection, a new unsignalized (traffic-lights-free) traffic movement control has been implemented to control completely autonomous vehicular traffic safely and efficiently using IVC & V2I communication, without a need for conventional traffic lights. An effective platooning strategy has been introduced to command AVs to join or unjoin platoons to improve road capacity utilization and throughput at the intersection. Finally, a new dynamic programming-based sequence optimization algorithm has been formulated to determine the optimal release sequence of approaching AV/platoon traffic through the intersection by minimizing the total time-loss delays due to the conflicts at the intersection. MIDAS AI system performance has been studied by implementing the V2X communication network in simulation using SUMO microscopic traffic simulator, VEINS and PLEXE platooning protocol. Simulation study shows that MIDAS platooning strategy has significantly improved the throughput and time-loss delays of AVs through an unsignalized autonomous intersection.

Finally in chapter 3, a new architecture for dynamically controlling autonomous vehicular traffic in real-time by learning an optimal AV clearance policy using deep reinforcement learning concepts and unsignalized autonomous intersection control called MIDAS RAIC

is developed and the performance is tested with respect to time loss delay in simulation using SUMO micro-simulator. MIDAS RAIC uses efficient state space representation of unsignalized autonomous intersection corridor which proved to be effective in training the agent. MIDAS RAIC also uses effective techniques in deep reinforcement learning like experience replay and additional target Q network for achieving stable training and improving training convergence. Simulation results showed that the agent has learned a good action-policy by minimizing the total cumulative time-loss delays. By the end of the training, agent has significantly reduced the time loss delays of the autonomous vehicles approaching the unsignalized intersection and learned the Q value function. Results show that trained MIDAS RAIC agent performed as good as the DP based autonomous intersection control system, MIDAS AI. The trained agent can be deployed into real world traffic networks for achieving a real-time adaptive and dynamic traffic control solution for autonomous vehicular traffic. The proposed MIDAS RAIC architecture is generalized enough to control and manage autonomous vehicular platoons.

4.2 Future Research

The underlying assumption in chapter 2 and 3 is that the respective proposed MIDAS implementations control a homogenous traffic environment with fully autonomous vehicles. But the transformation of human driven traffic to completely autonomous traffic is only possible by going through a phase where human driven vehicles and autonomous vehicles share the infrastructural capacity of road network. So, the direction for future research is to develop a proactive traffic control and platoon management system for heterogenous traffic environment, considering the co-existence of both human-driven and

autonomous vehicular traffic. The future research should consider two different traffic flow and infrastructural designs to accommodate heterogenous traffic environment as follows.

a) *Designated lanes for autonomous vehicles*

In this proposed design, see Figure B.1, an upstream link has designated lanes for autonomous vehicles. As developed in chapter 2 of this dissertation, a platooning model needs to be implemented to manage and control autonomous vehicles on these designated lanes. Similarly, a hybrid traffic signal control optimization model is required at the intersection to schedule the arrivals and departures of heterogenous traffic safely and efficiently.

b) *Competing behavior of heterogenous traffic with lane sharing*

In this proposed design see Figure B.1 there aren't any designated lanes for autonomous vehicles and there exists a competing behavior among the heterogenous traffic. The erratic lane changing behavior, speed and car following behavior of human driven vehicles introduce stochasticity into the traffic control and management of heterogenous traffic. In such design, a probabilistic platooning model needs to be developed to manage and control the heterogenous mixture of vehicles by providing recommendations to human driven connected vehicles in the network. Also, an optimal signal control optimization methodology is required to control such flow of heterogenous traffic through the intersections.

REFERENCES

1. Andrews, Jeffrey G., et al., (2014), "What will 5G be?." *IEEE Journal on selected areas in communications* 32.6: 1065-1082.
2. Arizona Department of Transportation (n.d.-a). Retrieved from <https://www.azdot.gov/mobile/media/news/2017/07/18/diverging-diamond-interchange-proposed-for-i-17-at-happy-valley-road>
3. Allsop, Richard E., (1971), "SIGSET: a computer program for calculating traffic signal settings." *Traffic Engineering and Control*.
4. Al Shayeb, S., Dobrota, N., Stevanovic, A., and Mitrovic, N. (2021). *Assessment of Arterial Signal Timings based on Various Operational Policies and Optimization Tools* (No. TRBAM-21-03665).
5. Arel, I., Liu, C., Urbanik, T., and Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2), 128-135.
6. Balaji, P. G., German, X., and Srinivasan, D. (2010). Urban traffic signal control using reinforcement learning agents. *IET Intelligent Transport Systems*, 4(3), 177-188.
7. Bellman, R., (1966). "Dynamic programming." *Science* 153.3731: 34-37.
8. Bullock, G. L., (2012), Tru-Traffic 10 Quality-Assured Traffic Signal Coordination User's Manual.
9. Chang, E. C., and Messer, C. J. (1991). Arterial signal timing optimization using PASSER II-90-Program user's manual.
10. Chen, C. (2003) Freeway Performance Measurement System (PeMS), *California PATH Report UCB-ITS-PRR-2003-22*, University of California, Berkeley, CA.
11. Dell'Olmo, P., and Mirchandani, B. (1995). REALBAND: An approach for real-time coordination of traffic flows on networks.
12. Dahlman, E., Parkvall, S., and Skold, J., (2013). *4G: LTE/LTE-advanced for mobile broadband*. Academic press.
13. El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): methodology and large-scale application on downtown Toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3), 1140-1150.

14. Fambro, D.B., N.A. Chaudhary, J.A. Bonneson, C.J. Messer, and L.L. Arabie (1991), *PASSER III –90 User’s Manual and Application Guide*”, Texas Transportation Institute, College Station, TX, 104 pages.
15. Fang, F. C., L. Elefteriadou (2006). Development of an Optimization Methodology for Adaptive Traffic Signal Control at Diamond Interchanges, *Journal of Transportation Engineering* 132(8).
16. FHWA, The Urban Traffic Control System in Washington, DC." Federal Highway Administration, U.S. Department of Transportation, Washington, DC, September 1974.
17. Gartner, N. H. (1983). OPAC: A demand-responsive strategy for traffic signal control (No. 906).
18. Gartner, N. H., Tarnoff, P. J., and Andrews, C. M. (1991). Evaluation of optimized policies for adaptive control strategy. *Transportation Research Record*, (1324).
19. Gettman, D., Head, L., and Mirchandani, P. (1999). RHODES-ITMS corridor control project. Final report, September 1996--May 1999 (No. PB-99-163792/XAB). Arizona Univ., Dept. of Systems and Industrial Engineering, Tucson, AZ (United States); Federal Highway Administration, Phoenix, AZ (United States); Arizona Dept. of Transportation, Phoenix, AZ (United States).
20. Gartner, N. H., Pooran, F. J., and Andrews, C. M. (2002). Optimized policies for adaptive control strategy in real-time traffic adaptive control systems: Implementation and field testing. *Transportation Research Record*, 1811(1), 148-156.
21. Head, K.L, S. Joshua, S. Shelby, D. Gettman, P. Mirchandani (1998), RHODES-ITMS: Real-Time Traffic Signal Control at a Diamond Interchange, 77th TRB Annual Meeting, Washington DC., Paper no. 981429
22. Hunt, P.B., Robertson, D.I., Bretherton, R.D. and Winton, R.I., (1981). *SCOOT-a traffic responsive method of coordinating signals* (No. LR 1014 Monograph).
23. Improta, G., and G. E. Cantarella (1984). "Control system design for an individual signalized junction." *Transportation Research Part B: Methodological* 18.2: 147-167.
24. Khoudour, L., Lesort, J.-B., Farges, J.-L, (1991). PRODYN – three years of trials in the ZELT experimental zone. Recherche-Transports-Securite (English issue: Special Traffic Management)
25. Little, J. D., Kelson, M. D and Gartner, N. H. (1981). MAXBAND: A versatile program for setting signals on arteries and triangular networks.

26. Little J. D. C. (1966) The synchronization of traffic signals by mixed-integer linear programming. *Operations Research*, 14, 568-594.
27. McTrans, Urban Streets Users Guide, 2017
28. Messer, C.J., and M.S. Chang (1987). “Traffic Operations of Basic Traffic-Actuated Control Systems at Diamond Interchanges”, *Transportation Research Record* 1114, 54-62.
29. Messer, C. J., Fambro, D. B., and Richards, S. H. (1977). Optimization of pretimed signalized diamond interchanges. *Transportation Research Record*, (644).
30. Messer, C. J., Whitson, R. H., Dudek, C. L., and Romano, E. J. (1973). A variable-sequence multiphase progression optimization program. *Highway Research Record*, 445(1973), 24-33.
31. Mirchandani, P., and Head, L. (2001). A real-time traffic signal control system: architecture, algorithms, and analysis. *Transportation Research Part C: Emerging Technologies*, 9(6), 415-432.
32. Mirchandani, P., Principal Investigator (2012), *MIDAS Project: Proactive traffic management with temporal-spatial monitored vehicles and distributed network control*, National Science Foundation, (with ASU co-PIs Huang, D., and Li, B., and University of Florida co-PI Yin, Y.).
33. Mirchandani, P., (2015), “*MIDAS-CPS – The Possible Future of Proactive Traffic Management Systems*”, Invited PACTRANS Seminar at University of Washington Seattle, WA, s, Arlington VA, (<https://youtu.be/fNWrt9DHLnQ>).
34. Morgan J. T. and Little J. D. C. (1964) Synchronizing traffic signals for maximal bandwidth. *Operations Research*, 12, 896-912
35. National Transportation Communications for ITS Protocol (2005): Object Definitions for Actuated Traffic Signal Controller (ASC) Units – 1202 v01.07. *National Electrical Manufacturers Association, Rosslyn, Virginia*.
36. Papageorgiou, M., Diakaki, C., Dinopoulou, V., Kotsialos, A., and Wang, Y. (2003). Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12), 2043-2067.
37. Pooran, F. J., Tarnoff, P. J., and Kalaputapu, R. (1996). RT-TRACS: Development of the real-time control logic. *In Intelligent Transportation: Realizing the Benefits. Proceedings of the 1996 Annual Meeting of ITS America. ITS America*.
38. PTV VISTRO User Manual. PTV America, Vol. 2, 2014, p. 07.

39. PTV VISSIM, (2014), <http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/>
40. Robertson, D. I. (1969). TRANSYT: a traffic network study tool.
41. Sims, A. G., and Dobinson, K. W. (1980). The Sydney coordinated adaptive traffic (SCAT) system philosophy and benefits. *IEEE Transactions on vehicular technology*, 29(2), 130-137.
42. Smith, S., Barlow, G., Xie, X. F., and Rubinstein, Z., (2013), *Transportation Research Board 92nd Annual Meeting Compendium of Papers*.
43. Stevanovic, A., Kergaye, C., and Martin, P. T. (2009, January). Scoot and scats: A closer look into their operations. *In 88th Annual Meeting of the Transportation Research Board. Washington DC*.
44. Tian, Z. Z., Balke, K., Engelbrecht, R., and Rilett, L. (2002). Integrated control strategies for surface street and freeway systems. *Transportation Research Record*, 1811(1), 92-99.
45. Tian, Z., C Messer, K Balke, T Urbanik, (2005), "[Part 2: Traffic Signal Systems: Integration of Diamond Interchange and Ramp Metering Operations](#)" *Transportation Research Record*.
46. Tian, Z.,(2007), "[Modeling and implementation of an integrated ramp metering-diamond interchange control system](#)", *Journal of Transportation Systems Engineering*.
47. Urbanik, I. I., Turnbull, K. F., Lindquist, E., Middleton, D., Balke, K., Ullman, G., and Lobaugh, G. (1997). *Texas A&M ITS Research Center of Excellence Narrative Summary Report: Fiscal Year 1996-97* (No. FHWA/TX-98/1439-4).
48. Venglar, S. P., Koonce, P., and Urbanik II, T. (1998). PASSER TM III-98 Application and User's Guide. *Texas Transportation Institute, Texas A&M University, College Station, TX*.
49. Wallace, Charles E., et al., (1984), *TRANSYT-7F user's manual*. No. UF-TRC-U32 FP-06/07.
50. Waze. (n.d.-a). Retrieved from <https://www.waze.com>
51. Webster, F. V. (1958). Traffic Signal Settings. Road Research Technical Paper HMSO, London.
52. Xu, H., Liu, H., and Tian, Z. (2010). Control delay at signalized diamond interchanges considering internal queue spillback. *Transportation Research Record*, 2173(1), 123-132.

53. Yagar, S. (1974). Capacity of a signalized road junction: critique and extensions. *Transportation Research*, 8, 137-147.
54. White, S. & Satran M. The Component Object Model. *Microsoft*
<https://docs.microsoft.com/en-us/windows/win32/com/the-component-object-model>
55. Kenney, J. B. (2011). Dedicated short-range communications (DSRC) standards in the United States. *Proceedings of the IEEE*, 99(7), 1162-1182.
56. Chaudhary, N. A., and Chu, C. L. (2000). *Guidelines for timing and coordinating diamond interchanges with adjacent traffic signals* (No. TX-00/4913-2,). Texas Transportation Institute, Texas A & M Univ.
57. Newell, G.F., 2002. A simplified car-following theory: a lower order model. *Transportation Research Part B: Methodological*, 36(3), 195-205.
58. Gazis, D.C., Herman, R. and Rothery, R.W., 1961. Nonlinear follow-the-leader models of traffic flow. *Operations research*, 9(4), 545-567.
59. Helly, W., 1959. Simulation of bottlenecks in single-lane traffic flow.
60. Gipps, P.G., 1981. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2), 105-111.
61. Wiedemann, R., 1974. Simulation des Strassenverkehrsflusses.
62. Krauß, S., Wagner, P. and Gawron, C., 1997. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5), 5597.
63. Chu D. Waymo One: A year of firsts. <https://blog.waymo.com/2019/12/waymo-one-year-of-firsts.html> (2019, accessed 20 April 2020).
64. Yurtsever, E., Lambert, J., Carballo, A. and Takeda, K., 2020. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8, 58443-58469.
65. P. Varaiya, "Smart cars on smart roads: problems of control", *Transactions on Automatic Control*, AC-38(2), 195-207, 1993.
66. Brackstone, M. and McDonald, M., 1999. Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour*, 2(4), 81-196.
67. Bellman, R., 1952. On the theory of dynamic programming. *Proceedings of the national Academy of Sciences*, 38(8), 716-719.

68. Treiber, M., Hennecke, A. and Helbing, D., 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2), 1805.
69. Treiber, M. and Kesting, A., 2013. Car-following models based on driving strategies. In *Traffic flow dynamics* (181-204). Springer, Berlin, Heidelberg.
70. Jiang, D. and Delgrossi, L., 2008, May. IEEE 802.11 p: Towards an international standard for wireless access in vehicular environments. In *VTC Spring 2008-IEEE vehicular technology conference* (2036-2040). IEEE.
71. Krajzewicz, D., Hertkorn, G., Rössel, C. and Wagner, P., 2002. SUMO (Simulation of Urban MObility)-an open-source traffic simulation. In *Proceedings of the 4th middle East Symposium on Simulation and Modelling (MESM20002)* (183-187).
72. Sommer, C., German, R. and Dressler, F., 2010. Bidirectionally coupled network and road traffic simulation for improved IVC analysis. *IEEE Transactions on mobile computing*, 10(1), 3-15.
73. Segata, M., Cigno, R.L., Harges, T., Heinovski, J., Schettler, M., Bloessl, B., Sommer, C. and Dressler, F., 2022. Multi-Technology Cooperative Driving: An Analysis Based on PLEXE. *IEEE Transactions on Mobile Computing*.
74. Sichitiu, M.L. and Kihl, M., 2008. Inter-vehicle communication systems: a survey. *IEEE Communications Surveys & Tutorials*, 10(2), 88-105.
75. Al-Sultan, S., Al-Doori, M.M., Al-Bayatti, A.H. and Zedan, H., 2014. A comprehensive survey on vehicular ad hoc network. *Journal of network and computer applications*, 37, 380-392.
76. Sommer, C., German, R. and Dressler, F., 2010. Bidirectionally coupled network and road traffic simulation for improved IVC analysis. *IEEE Transactions on mobile computing*, 10(1), 3-15.
77. Dresner, K. and Stone, P., 2008. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31, 591-656.
78. Fayazi, S.A. and Vahidi, A., 2018. Mixed-integer linear programming for optimal scheduling of autonomous vehicle intersection crossing. *IEEE Transactions on Intelligent Vehicles*, 3(3), 287-299.
79. Mirheli, A., Hajibabai, L. and Hajbabaie, A., 2018. Development of a signal-head-free intersection control logic in a fully connected and autonomous vehicle environment. *Transportation Research Part C: Emerging Technologies*, 92, 412-425.

80. Müller, E.R., Carlson, R.C. and Junior, W.K., 2016. Intersection control for automated vehicles with MILP. *IFAC-PapersOnLine*, 49(3), 37-42.
81. Levin, M.W. and Rey, D., 2017. Conflict-point formulation of intersection control for autonomous vehicles. *Transportation Research Part C: Emerging Technologies*, 85, 528-547.
82. Milanés, V., Shladover, S.E., 2014. Modeling cooperative and autonomous adaptive cruise control dynamic responses using experimental data. *Transportation Research Part C: Emerging Technologies* 48, 285-300.
83. Milanés, V., Shladover, S.E., Spring, J., Nowakowski, C., Kawazoe, H., Nakamura, M., 2014. Cooperative adaptive cruise control in real traffic situations. *IEEE Transactions on Intelligent Transportation Systems* 15 (1), 296–305.
84. Zwaneveld, P.J. and Van Arem, B., 1997. Traffic effects of automated vehicle guidance systems: a literature survey.
85. Vander Werf, J., Shladover, S.E., Miller, M.A. and Kourjanskaia, N., 2002. Effects of adaptive cruise control systems on highway traffic flow capacity. *Transportation Research Record*, 1800(1), 78-84.
86. Davis, L.C., 2004. Effect of adaptive cruise control systems on traffic flow. *Physical Review E*, 69(6), 066110.
87. Xu, Q., Mak, T., Ko, J. and Sengupta, R., 2004, October. Vehicle-to-vehicle safety messaging in DSRC. In *Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks* (19-28).
88. Yang, X., Liu, L., Vaidya, N.H. and Zhao, F., 2004, August. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.* (114-123). IEEE.
89. Sheikholeslam, S. and Desoer, C.A., 1990, May. Longitudinal control of a platoon of vehicles. In *1990 American control conference* (291-296). IEEE.
90. Rajamani, R., Tan, H.S., Law, B.K. and Zhang, W.B., 2000. Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons. *IEEE Transactions on Control Systems Technology*, 8(4), 695-708.
91. VanderWerf, J., Shladover, S., Kourjanskaia, N., Miller, M. and Krishnan, H., 2001. Modeling effects of driver control assistance systems on traffic. *Transportation Research Record*, 1748(1), 167-174.

92. Van Arem, B., Van Driel, C.J. and Visser, R., 2006. The impact of cooperative adaptive cruise control on traffic-flow characteristics. *IEEE Transactions on intelligent transportation systems*, 7(4), 429-436.
93. Schakel, W.J., Van Arem, B. and Netten, B.D., 2010, September. Effects of cooperative adaptive cruise control on traffic flow stability. In *13th International IEEE Conference on Intelligent Transportation Systems (759-764)*. IEEE.
94. Shladover, S.E., Su, D. and Lu, X.Y., 2012. Impacts of cooperative adaptive cruise control on freeway traffic flow. *Transportation Research Record*, 2324(1), 63-70.
95. Naus, G.J., Vugts, R.P., Ploeg, J., van De Molengraft, M.J. and Steinbuch, M., 2010. String-stable CACC design and experimental validation: A frequency-domain approach. *IEEE Transactions on vehicular technology*, 59(9), 4268-4279.
96. Ploeg, J., Scheepers, B.T., Van Nunen, E., Van de Wouw, N. and Nijmeijer, H., 2011, October. Design and experimental evaluation of cooperative adaptive cruise control. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC) (260-265)*. IEEE.
97. Rajamani, R., 2011. *Vehicle dynamics and control*. Springer Science & Business Media.
98. Thormundsson, B. Artificial intelligence software market revenue worldwide 2018-2025. *Statista* <https://www.statista.com/statistics/607716/worldwide-artificial-intelligence-market-revenues/>
99. Nirale, S. What makes emerging technologies future customer experience. *Servion* <https://servion.com/blog/what-emerging-technologies-future-customer-experience/>
100. Thorpe, T.L. and Anderson, C.W., 1996. Traffic light control using sarsa with three state representations. *Technical report, Citeseer*.
101. Wiering, M.A., 2000. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000) (1151-1158)*.
102. Abdulhai, B., Pringle, R. and Karakoulas, G.J., 2003. Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3).
103. Teodorović, D., Varadarajan, V., Popović, J., Chinnaswamy, M.R. and Ramaraj, S., 2006. Dynamic programming—neural network real-time traffic adaptive signal control algorithm. *Annals of Operations Research*, 143(1), 123-131.

104. Srinivasan, D., Choy, M.C. and Cheu, R.L., 2006. Neural networks for real-time traffic signal control. *IEEE Transactions on intelligent transportation systems*, 7(3), 261-272.
105. Mannion, P., Duggan, J. and Howley, E., 2016. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. *Autonomic road transport support systems*, 47-66.
106. Wang, Y., Yang, X., Liang, H. and Liu, Y., 2018. A review of the self-adaptive traffic signal control system based on future traffic environment. *Journal of Advanced Transportation*, 2018.
107. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G. and Petersen, S., 2015. Human-level control through deep reinforcement learning. *nature*, 518(7540), 529-533.
108. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D. and Riedmiller, M., 2013. Playing atari with deep reinforcement learning. *NIPS Deep Learning Workshop 2013*.
109. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D. and Wierstra, D., 2015. Continuous control with deep reinforcement learning. *4th International Conference on Learning Representations, ICLR 2016*.
110. Genders, W. and Razavi, S., 2016. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*.
111. Gao, J., Shen, Y., Liu, J., Ito, M. and Shiratori, N., 2017. Adaptive traffic signal control: Deep reinforcement learning algorithm with experience replay and target network. *arXiv preprint arXiv:1705.02755*.
112. Vidali, A., Crociani, L., Vizzari, G. and Bandini, S., 2019, June. A Deep Reinforcement Learning Approach to Adaptive Traffic Lights Management. In *WOA* (42-50).
113. Wei, H., Zheng, G., Yao, H. and Li, Z., 2018, July. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2496-2505).
114. Van der Pol, E. and Oliehoek, F.A., 2016. Coordinated deep reinforcement learners for traffic light control. *Proceedings of learning, inference and control of multi-agent systems (at NIPS 2016)*, 1.
115. Bellman, R., 1957. A Markovian decision process. *Journal of mathematics and mechanics*, 679-684.

116. T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, 2012
117. Sutton, R.S., 1995. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8.
118. Watkins, C.J. and Dayan, P., 1992. Q-learning. *Machine learning*, 8(3), 279-292.

APPENDIX A
COMMON SIGNAL PHASES USED BY TRAFFIC ENGINEERS AT DIAMOND
INTERCHANGES

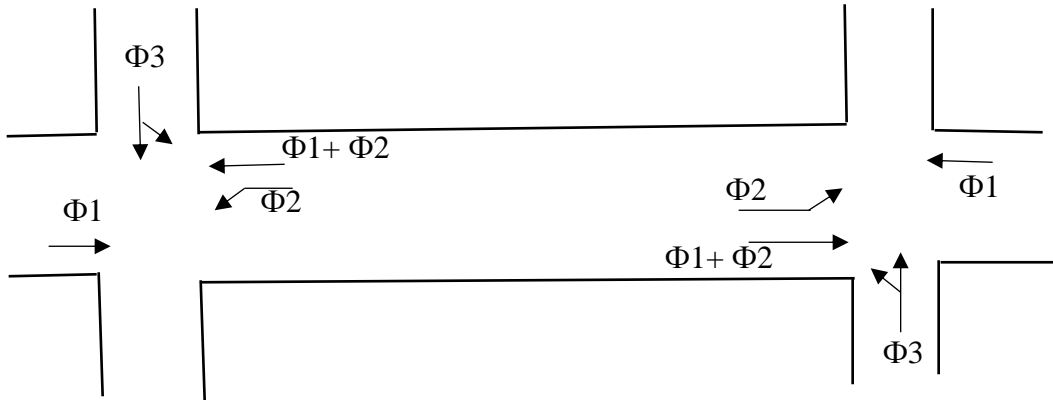


Figure A.1. "3-Phase Lag-Lag" Phase Movements

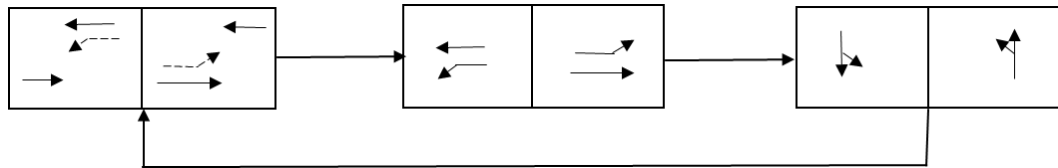


Figure A.2. "3-Phase Lag-Lag" Signal Plan

Table A.1. "3-Phase Lag-Lag" Signal Stages Used in DP.

Stage	Green Phases
1	$\Phi 1$ $\Phi 2$
2	$\Phi 2$
3	$\Phi 3$

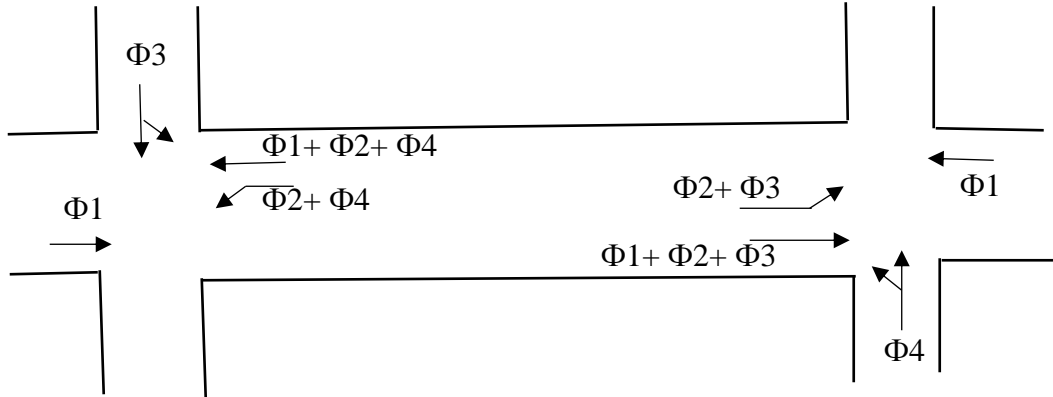


Figure A.3. "4-Phase Lead/Lag - Lag" Phase Movement

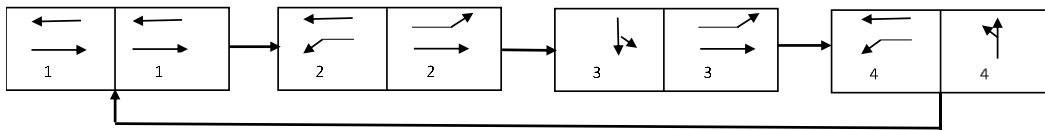


Figure A.4. "4-Phase Lead/Lag - Lag" Signal Scheme

Table A.2. "4-phase Lead/Lag - Lag" Signal Stages Used in DP

Stage	Green Phases
1	Φ1 →
2	Φ2 →
3	Φ3 →
4	Φ4 →

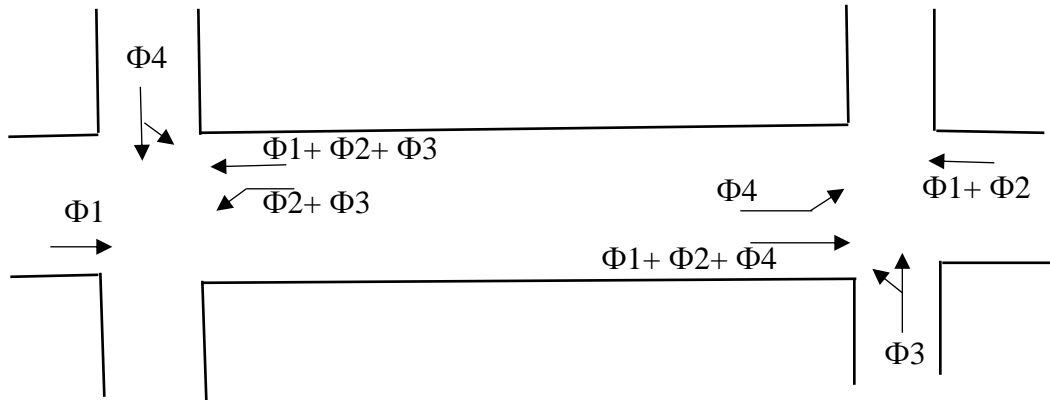


Figure A.5. "4-Phase Lead/Lag" Phase Movement

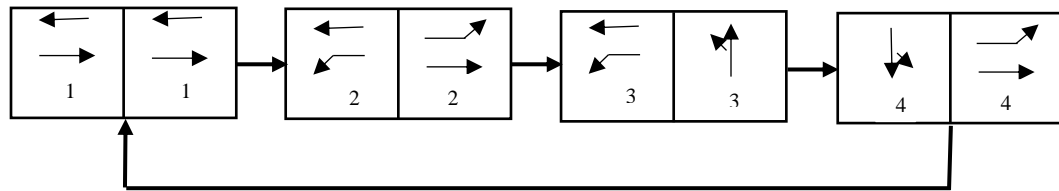


Figure A.6. "4-Phase Lead/Lag" Signal Scheme

Table A.3. "4-Phase Lead/Lag – Lag" Signal Stages Used in DP.

Stage	Green Phases
1	Φ1 →
2	Φ2 →
3	Φ3 →
4	Φ4 →

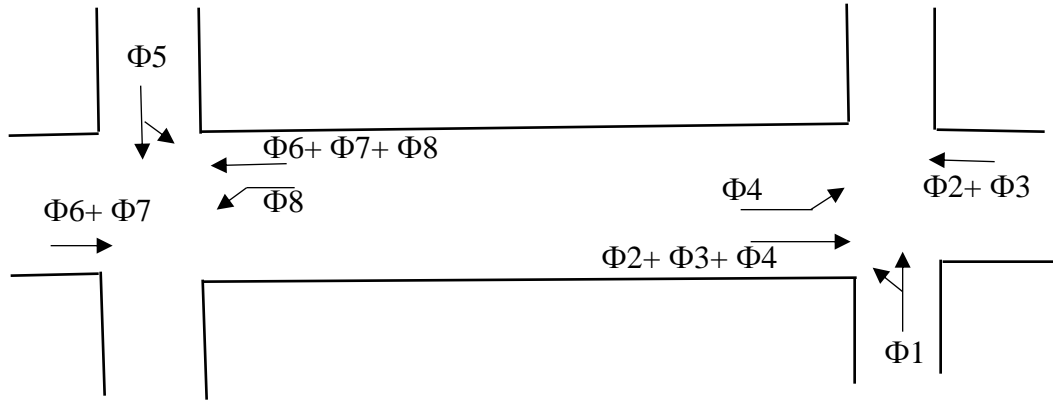


Figure A.7. "3-Phase with Overlaps" Phase Movement

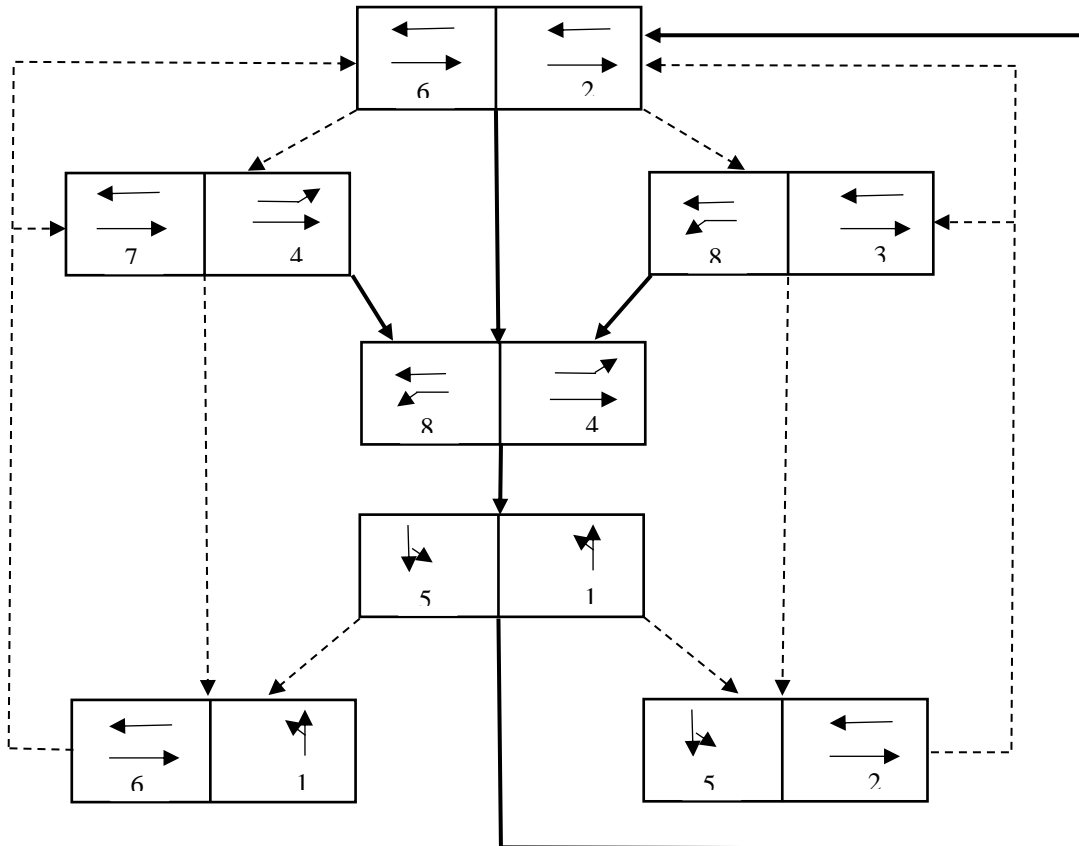
















Figure A.8. "3-Phase with Overlaps" Signal Scheme

Table A.4. “3-Phase with Overlaps” Signal Stages

Stage	Green Phases
1	$\Phi 1$  $\Phi 5$ 
2	$\Phi 1$  $\Phi 6$ 
3	$\Phi 2$  $\Phi 6$ 
4	$\Phi 2$  $\Phi 5$ 
5	$\Phi 3$  $\Phi 8$ 
6	$\Phi 4$  $\Phi 7$ 
7	$\Phi 4$  $\Phi 8$ 

APPENDIX B
ROAD DESIGNS TO CONSIDER FOR HETEROGENOUS TRAFFIC
ENVIRONMENTS

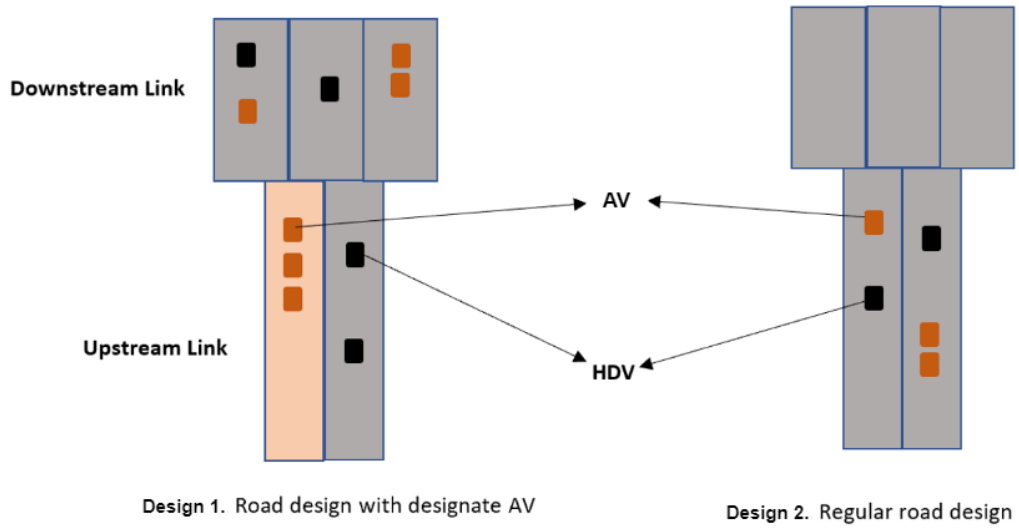


Figure B.1 Road Designs for Heterogenous Vehicular Traffic Environments