DYNAMIC PROGRAMMING BASED COORDINATED RAMP METERING ALGORITHMS

A thesis presented in partial fulfillment of the

requirements for the degree of

PhD of Engineering

in

Mechatronics

at

Massey University,

Auckland, New Zealand

Yu Xuefeng

ABSTRACT

Motorway congestion can be classified into two types, recurrent congestion and non-recurrent congestion. Recurrent congestion happens during peak hours. Non-recurrent congestion occurs due to car accidents, weather conditions or public events. Negative impacts of traffic congestion include wasted fuel, pollution, travel delay and spillover effects caused by slow traffic.

Ramp metering, as an only way to regulate traffic amount accessing to the motorway, is considered as the most cost-effective way to prevent the recurrent congestion. Coordinated ramp metering was developed to control a number of on-ramps simultaneously to improve traffic conditions on busy motorways. The existing coordinated ramp metering algorithms were normally established on macroscopic traffic flow models based on Payne's wok, the performances of which were measured by the employed macroscopic model themselves, and the released metering rates of which tended to be continuous. Implementations in microscopic traffic simulators were few.

This thesis presents DP (Dynamic Programming) based online control approaches for the optimal coordination of ramp metering and evaluates its performances in both macroscopic and microscopic traffic simulation environment. DP decision networks were proposed, where a traffic system can be modeled as a number of discrete traffic states and separated by time stages, and the control problem of coordinated ramp metering was treated as the minimization problem to search the optimal trajectory of discrete decision variables (ramp metering rates) that minimized a cost criterion in terms of TTS (total time spent) along the time horizon.

Experiments conducted in the macroscopic simulation environment demonstrated the full potential of proposed algorithms with precise queue constrains in an ideal deterministic environment, and experiments conducted in the microscopic simulation environment indicated the performances of the proposed algorithms in a stochastic environment and revealed the feasibility in the real world. The implementation of DP ramp metering was proposed under the framework of receding horizon control. A

6.7km stretch of motorway in Auckland, New Zealand, was chosen as a study location and constructed by a microscopic simulator as a simulation scenario and by a macroscopic traffic model as a prediction model. The simulation results indicated that the proposed algorithms were able to eliminate motorway queues under high traffic demands and manage queue lengths at metered on-ramps when queue constrains were not overstrict. The simulation results also revealed that 9 discrete metering rates for each ramp meter were adequate to prevent motorway queues. Such feature not only proved that the optimal trajectory converged very fast in the proposed DP decision networks, but also made on-line control system possible due to less computational load.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

NOMENCLATURE

1. Glossary

Speed (km/h): It, unless otherwise specified, refers to space mean speed in this thesis, and is measured as the average speed of all vehicles crossing a given length of motorway segment over a specified time period.

Density (veh/km/lane): The number of vehicles per unit length is measured along a given length of roadway segment.

Flow (veh/h): The number of vehicles is counted by a fixed detector over a certain time. *In macroscopic traffic models, speed, density, and flow are all related by a basic relation: the flow is equivalent to the product of speed and density.*

Occupancy (percentage): When a fixed detector was occupied by vehicles for a period of time, the percentage of occupied time in one detection interval is counted as occupancy. A rough conversation between density and occupancy can be given by: density is equivalent to occupancy divided by the average vehicle length. This conversion makes it possible that density can be measured at a fixed point.

Time headway (hour): Time headway is defined as, when two vehicles pass a fixed detector successively, the time period between the arrival of the front bumper of the first vehicle and the arrival of the front bumper of the succeeding vehicle.

Distance headway (km): Distance headway is defined as, when two vehicles pass a fixed detector successively, the distance between the arrival of the front bumper of the first vehicle and the arrival of the front bumper of the succeeding vehicle.

Capacity: The maximum possible flow at a point on the highway

Bottleneck: A location on the highway where the road capacity is a local minimum.

Congestion: A traffic condition under which vehicles travel slower than the drivers' desired speeds, restricted by the downstream conditions.

Critical density: The density value at which congestion may form at a point on the motorway.

Jam density: The density at which the traffic comes to a halt, i.e., the maximum density allowed at a point on the highway

Free-flow conditions: A traffic condition under which vehicles are able to travel freely at maximum allowed speeds.

Fundamental diagram: The functional relationship between the flow and density for a point on the highway.

Macroscopic model: A class of traffic models in which traffic is treated as a continuum and modeled by aggregated quantities such as density and flow.

Microscopic model: A class of traffic models in which dynamics of individual vehicle driver units and the interaction between these units and their surroundings are explicitly modeled, and the traffic itself is the collective behavior of all these vehicle driver units.

2. List of Symbols

The following symbols are only used to present glossaries above. More notations and definitions are given in Chapter 3.

v	*(km/hour)*	*Speed*
q	*(veh/hour)*	*Flow*
ρ	*(veh/km/lane)*	*Density (macroscopic model)*
k	*(veh/km/lane)*	*Density (fundamental diagram)*
v_f	*(km/hour)*	*Free flow speed*
k_{jam}	*(veh/km/lane)*	*Jam density (fundamental diagram)*
ρ_{jam}	*(veh/km/lane)*	*Jam density (macroscopic model)*
$k_{critical}$	*(veh/km/lane)*	*Critical density (fundamental diagram)*
$\rho_{critical}$	*(veh/km/lane)*	*Critical density (macroscopic model)*
r	*(veh/hour)*	*Ramp metering rates*

ACKNOWLEDGEMENTS

I would like to gratefully and sincerely thank Prof. Peter Xu, Prof. Fakhrul Alam and Prof. Johan Potgieter, my research supervisors, for the opportunity they offered to develop my own individuality and self-sufficiency by being allowed to work with such independence, for the mentorship they provided to guide a less experienced beginner to became an independent thinker, for rudeness they endured from a total stranger who is unwilling to submit to his fate. I would also like to thank Prof. Clara Fang for her encouragement and kind suggestions of this research work.

Finally, and most importantly, I would like to thank my parents:

谢谢你们身体健康，谢谢你们让我觉得还没长大。

CHAPTER 1 INTRODUCTION

1.1 Motorway Congestion

Congested motorway segments are normally located at upstream and downstream bottleneck locations, which are usually caused by the disturbances from bottleneck locations such as ramp merge, weaving segments or lane drops. Motorway queues[1] could be propagated from the congested road segments to expand to the uncongested motorway segments and cause motorway congestion. The nature of motorway congestion can be observed by either empirical work or theoretical traffic models.

Highway Capacity Manual, HCM (2000), classifies traffic flow within basic motorway segments into three categories (Fig. 1.1), undersaturated flow, queue discharge flow and oversaturated flow. Undersaturated flow represents traffic flow that remains unaffected by the downstream or upstream bottleneck locations within a speed range of 70km/hour to free-flow speed. Queue discharge flow represents traffic flow that has just gone through bottleneck locations and speeds tend to be restored to free-flow speed within a range of 55km/hour to free-flow speed. Oversaturated flow represents traffic flow that has been seriously affected by the downstream bottleneck, and vehicles are passing through motorway queues with stop-and-go movements.

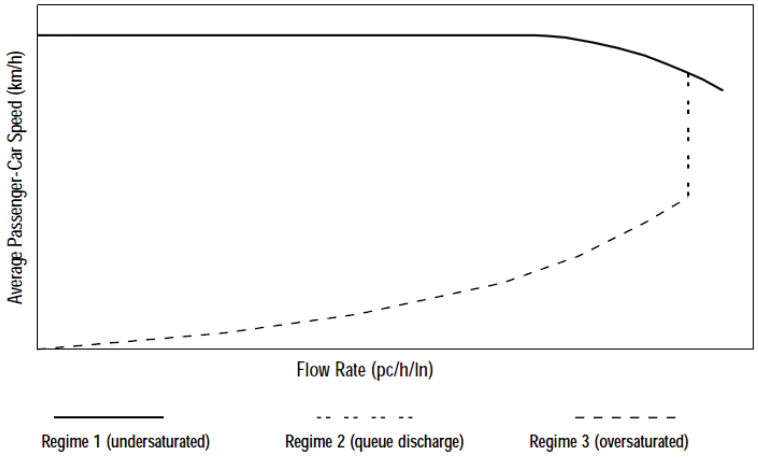


Figure 1.1 The suggested speed-flow relationship from HCM (2000)

[1] Unlike the queues propagated at interchanges, those motorway queues are not static queues but vehicular flows moving as stop-and-go waves.

As can be seen in Fig. 1.1, speeds remain constant when traffic flow is undersaturated, while dropping significantly in regime2 and regime3 where motorway queues appear. In order to stabilize speeds of traffic flow, traffic flow within basic motorway segments is expected to remain as undersaturated flow, while oversaturated flow or congested flow should be avoided.

Traffic stream models can be also employed to reveal more insights of motorway congestion, since motorways operate under the purest form of uninterrupted traffic flow (HCM 2000). Greenshields' model (Greenshields, 1935) assumed a linear speed-density relationship for uninterrupted traffic flow. Based on this assumption, two parabolic curves can be derived for speed-flow relationship and flow-density relationship.

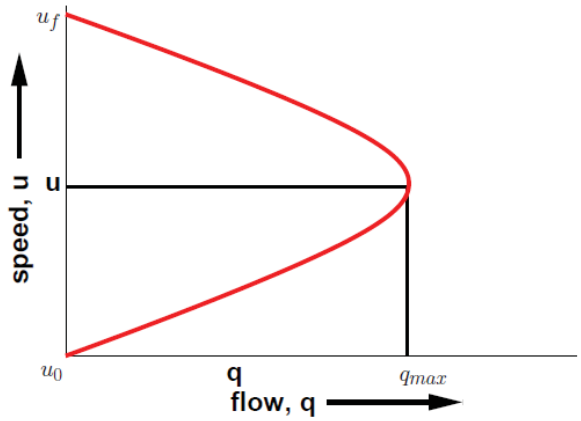


Figure 1.2 The derived speed-flow relationship from Greenshield's linear assumption

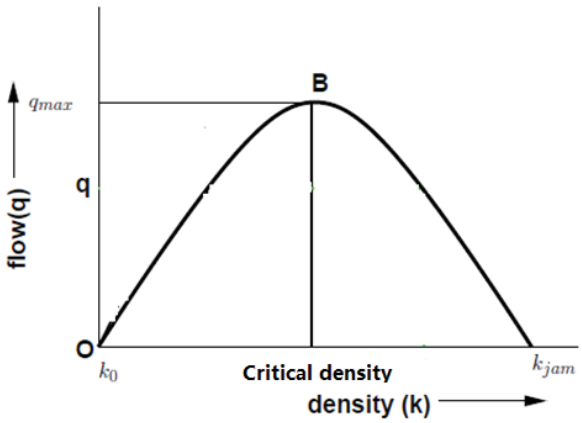


Figure 1.3 The derived flow-density relationship from Greenshield's linear assumption

Fig. 1.2 rephrases the discontinuous empirical curves (Fig. 1.1) by a continuous parabolic curve. Fig. 1.3 reveals an important fact that, to maximize throughput of

motorways, traffic densities should maintain at a certain point called the critical density. In other words, oversaturated flow, or traffic congestion, could be formed within basic motorway segments when traffic density exceeds the critical density.

1.2 Ramp Metering

Ramp metering is a deployment of traffic signals operating on ramps to control the traffic amount entering motorway networks. The potential improvement achieved by ramp metering could be generalized as follows(Papageorgiou & Papamichail, 2007):

- Reduce motorway congestion in space and time
- Increase motorway throughput.
- Reduced congestion spillback to the adjacent urban traffic network or to other merging motorways.
- Significant improvement of traffic safety due to reduced congestion duration.

In practice, ramp control strategies can be classified as fixed time control, local traffic responsive control and coordinated ramp metering control(Bogenberger & May, 1999). The first two control strategies are only used to optimize the local traffic condition by controlling a single on-ramp. The coordinated ramp metering strategies are designed to optimize the traffic conditions over traffic networks by controlling multiple on-ramps simultaneously.

The Greenshield's parabolic assumption (Fig. 1.3) can be employed to explain how a local ramp metering works (Fig. 1.4).

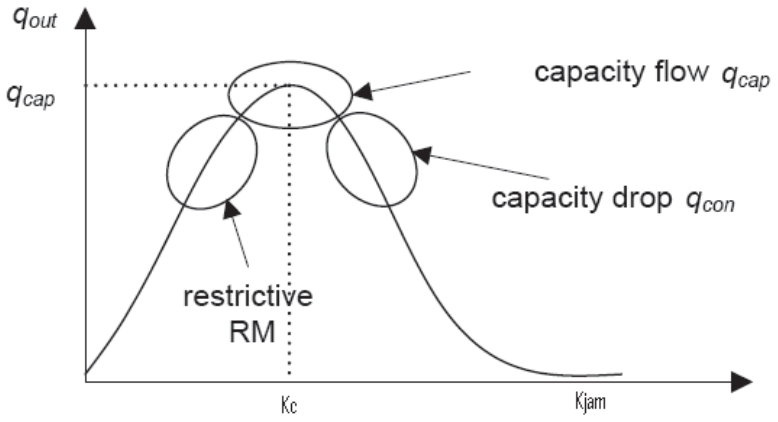


Figure 1.4 How ramp meters work(Papageorgiou & Papamichail, 2007)

If motorway congestion may form when traffic density exceeds the critical density, a successful ramp control strategy should be capable of maintaining traffic density around the critical point and traffic flow around the maximum flow rate or capacity flow (q_{cap}). If ramp metering is too restrictive, the mainstream throughput could not reach the motorway capacity and ramp metering actually causes extra delay for traffic flow. If ramp metering is too permissive, congestion may still happen and cause the capacity drop. From the drivers' point of view, if they want to shorten the duration of reaching their respective destinations along motorway networks, then ramp metering should be capable of decreasing the total time spent (TTS) on travelling and queuing due to the avoidance of capacity drop caused by motorway congestion.

Real-time coordinated ramp metering can be illustrated by a control loop (Fig. 1.5).

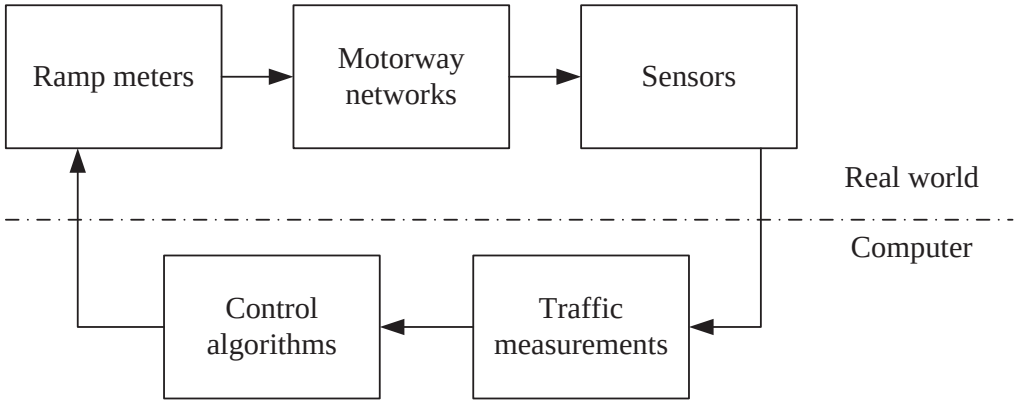


Figure 1.5 The control loop for the coordinated ramp metering

In Fig. 1.5, ramp meters manage traffic volumes from on-ramps to avoid occurrence of congestions in motorway networks. Regulated traffic flow on motorways is expected to steer current traffic conditions towards the optimal traffic conditions, which are monitored by suitable sensors such as inductive loop detectors. Traffic measurements collected by detectors at each control interval are sent to the ramp controllers, where optimal ramp signals are periodically computed by ramp control algorithms based on monitored traffic conditions. Under the control of the optimal signals, ramp meters are coordinated to interact with traffic flow on motorways and meter rates remain constant until new optimal signals are generated after next control interval. The performances of ramp metering can be measured by TTS (Total Time Spent) values in motorway networks.

1.3 Objectives of the Thesis

The aim of this research is to develop a model-based control algorithm for the optimal coordination of ramp metering by Dynamic Programming (DP) and to simulate and evaluate the proposed algorithms in both macroscopic and microscopic simulation environment. The macroscopic traffic simulation was conducted to demonstrate the full potential of the proposed algorithm in an ideal deterministic environment where prediction errors were unnecessary to be considered, while the microscopic traffic simulation was conducted to show a case study where the prediction model had to be calibrated with a stochastic environment. Two major objectives have been identified to attain the purpose.

- To develop a DP control algorithm for the optimal coordination of ramp metering. DP decision networks were proposed, where a traffic system was modeled as a number of discrete traffic states and separated by time stages. The ramp metering problem was treated as the minimization problem along the time stages in term of TTS values and was solved by forward recursion method. Traffic states over the time horizon were estimated by a macroscopic traffic model. The proposed algorithm was implemented under the framework of receding horizon control.
- To simulate and analyze the algorithm in a stochastic and microscopic simulation environment

  A commercial simulation model, Aimsun, was chosen as the microscopic simulation environment for the implementation of the proposed algorithms. A 6.7km stretch of motorway in Auckland was constructed both in Aimsun as a microscopic simulation scenario and in the macroscopic traffic model as a prediction model to estimate the traffic states over a prediction horizon. The prediction model was calibrated with the simulation results from the Aimsun scenario with low traffic demands, under which the mainline traffic on the motorway was under uncongested conditions. The proposed algorithm was implemented in the Aimsun scenario with high traffic demands, under which

heavy congestions could occur on the mainline of the motorway. The performances of the proposed algorithm were compared to those under no-ramp-control case and those under the control of local responsive ramp metering.

The primary output of thesis is expected to develop a new method of real-time optimization for ramp signal coordination. DP decision networks are present to provide a straightforward calculation for ramp control problems. Its simplicity makes the real-world implementation possible. The secondary output is expected to be the implementation procedure of the proposed algorithm in a stochastic microscopic environment, since the implementations of model-based coordinated ramp metering in microscopic simulations are not seen in the current publications.

1.4 Thesis Overview

The remainder of this thesis is organized as follows. In Chapter 2, the previous work related to this research was reviewed, including macroscopic traffic models, the existing ramp metering algorithms and microscopic traffic simulation. Chapter 3 presented the methodology of DP based ramp metering. A macroscopic traffic model employed as the prediction model was briefly introduced and an objective function with regard to TTS was also given. The ramp metering algorithm was depicted in a proposed DP decision network. In Chapter 4, a macroscopic simulation was conducted, where the proposed algorithm was implemented in the macroscopic model introduced in Chapter 3. A 6.7km stretch of Auckland Northern Motorway was chosen as a study location and constructed by the model as a macroscopic simulation scenario. The performance of the algorithm was measured and compared to those under no-ramp-control case and those under the control of local responsive ramp metering. In Chapter 5, the study location was constructed in Aimsun as a microscopic simulation scenario. The proposed algorithms were implemented in an Aimsun API module to communicate with the microscopic simulator. The prediction model was

calibrated with simulation results from the Aimsun scenario when the mainline traffic was under uncongested conditions. The performance of DP ramp metering was then evaluated under different traffic demands and compared to those under no-ramp-control case and those under local responsive ramp metering. Chapter 6 concludes this thesis with some remarks on future research.

CHAPTER 2 LITERATURE REVIEW

This chapter reviews the previous work related to this study. The background of the fundamental diagram was briefly introduced to acquire a general understanding for modeling traffic flow before the development of macroscopic traffic models are presented. The existing ramp metering algorithms were classified and described. The car-following theories and some noticeable microscopic simulators were also reviewed. Traffic variables frequently mentioned in this thesis are defined as follows.

2.1 The Fundamental Diagrams

Traffic flow can be classified as interrupted flow or uninterrupted flow (HCM 2000). The fundamental diagrams are to establish the stationary relationships between traffic variables for uninterrupted traffic flow (May, 2000). The relationships could be based on mathematical assumptions or on empirical knowledge. The following subsections simply introduce major previous work on the relationships of three traffic variables: speed, density and flow.

2.1.1 Speed - Density Relationship

Greenshields (1935) made the linear assumption for speed-density relationship (Fig. 2.1) from an aerial photographic study.

$$v = v_f(1 - \frac{k}{k_j}) \tag{2.1}$$

Where, v_f is the free-flow speed is and k_j is the jam density.

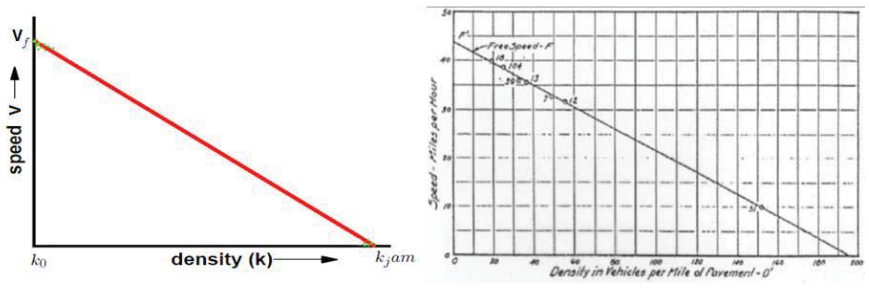


Figure 2.1 Speed – density relationship (Greenshields, 1935)

This assumption significantly underestimated the jam density, so the congested

portion of the graph did not fit the obtained traffic data especially for those from motorways. Greenberg (1959) gave another assumption of a logarithmic curve (Figure 2.2).

$$v = v_o ln(\frac{k}{k_j}) \tag{2.2}$$

Where, v_o is the optimum speed, but that is also very difficult to be observed in the field. A crude estimate of the value of v_o was given by one-half of free-flow speed. Another disadvantage of the logarithmic assumption is that speed goes to infinity as density goes to zero.

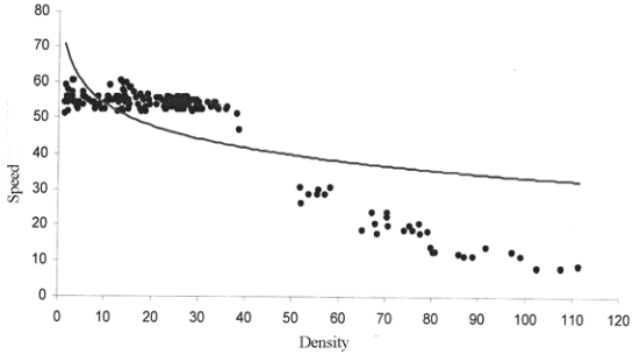


Figure 2.2 Greenberg's speed – density relationship (Ardekani, Ghandehari, & Nepal, 2011)

The third equation for speed-density relationship was proposed by Underwood (1961) as a result of traffic studies on the Merritt Parkway in Connecticut.

$$v = v_f e^{-k/k_o} \tag{2.3}$$

Its jam density goes to infinity as speed goes to zero (Fig. 2.3). Another disadvantage is that the optimum density, k_o, is very hard to be identified and various, depending on the road environment.

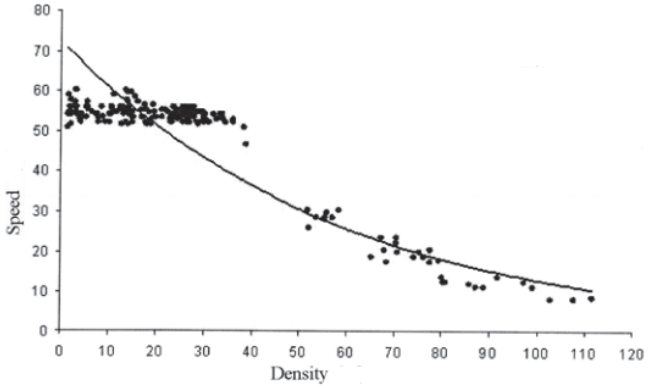


Figure 2.3 Underwood's Speed – density relationship (Ardekani et al., 2011)

Drake, Schofer, and May (1967) proposed the fourth equation based on the observation of the most speed-density curves appearing as S-shaped curves and the knowledge of limitations of the Underwood's equation (Fig. 2.4).

$$v = v_f e^{-1/2(k/k_o)^2} \tag{2.4}$$

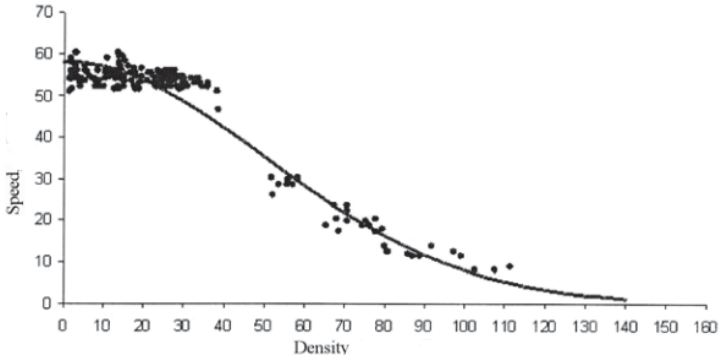


Figure 2.4 Drake's Speed – density relationship (Ardekani et al., 2011)

Drake et al. (1967) compared the performances of the four equations mentioned above and analyzed estimates of the speed and density parameters for each equation based on the field measured data. They found that Underwood's equation overestimated free-flow speed while Greenberg's equation underestimated optimum speed and that Greenshield's equation predicted a much lower jam density than would be expected from field measurements while Underwood's and Drake's equations led to jam densities of infinity when speeds went to zero. They concluded that each equation only fits for some portions of the density range.

The four speed-density relationships above assume one continuous function for the complete range of flow conditions including free-flow and congested flow situations, so they were also categorized as single-regime traffic models (Ceder & May, 1976). Considering that the deficiencies of single-regime models, Edie (1961) proposed the first multi-regime model, in which, the Underwood's equation was used for free-flow regime while the Greenberg's equation was used for the congested-flow regime (Fig. 2.5).

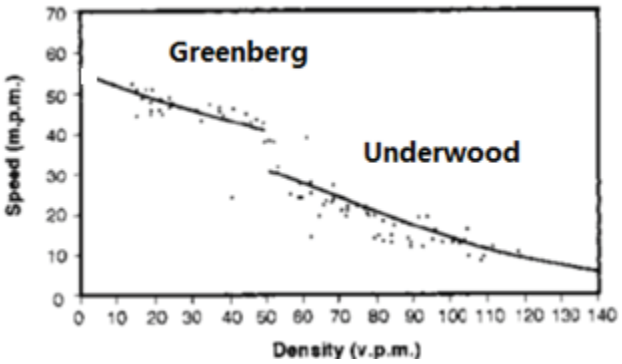


Figure 2.5 Two-regime speed – density relationship (Edie, 1961)

Inspired by Edie's idea, Drake et al. (1967) proposed other three multi-regime models by combining different portions of density ranges of the previous single-regime models. The breaking point was identified by Quandt's work on likelihood functions (Quandt, 1958, 1960).

On the other hand, Pipes (1967) and Drew (1968) modified the Greenshields' equation to generalize a family of single-regime models for different traffic situations by introducing a tunable parameter n to the original equation. Varying n from -1 to $+1$, the speed-density relationship can be exponential ($n = -1$), parabolic ($n = 0$) or linear ($n = +1$).

Another family based on the single-regime model, Eq. (2.5), was proposed and evaluated by researchers at the University of California (Ceder & May, 1976; May & Keller, 1968).

$$v^{1-m} = {v_f}^{1-m}[1 - (\frac{k}{k_j})^{l-1}] \tag{2.5}$$

Where, $0 \leq m < 1$ and $l > 1$. Since the identification of m and l was highly related to the site's characteristics, Easa and May (1980) developed a nomograph procedure based on the field measured traffic data, such as jam density and free-flow speed, to select m and l. In addition, the efforts was also made to develop a family of two-regime models by May and Keller (1967). Current researches involving the speed-density relationship normally employed a modified Drake's equation,

$$v = v_f e^{-1/m(k/k_o)^m} \tag{2.6}$$

It seems inspired by the early studies of May and his colleagues (1976; 1968).

2.1.2 Speed - Flow Relationship

Most previous work of speed-density relationship was conducted by early researchers in the 60's and 70's. They believed that once speed-density equations are determined, speed-flow relationship or density-flow relationship can be derived from it (Gartner, Messer, & Rathi, 2005; Gerlough & Huber, 1976). Take Greenshields' equation as an example.

Substituting $q = k \cdot v$ to Eq. (2.1), we obtain

$$q = k_j(v - \frac{v^2}{v_f}) \tag{2.7}$$

Where, q represents flow that is equal to the product of speed and density. Eq. (2.7) reveals a parabolic relationship between flow and speed (Fig. 2.6).

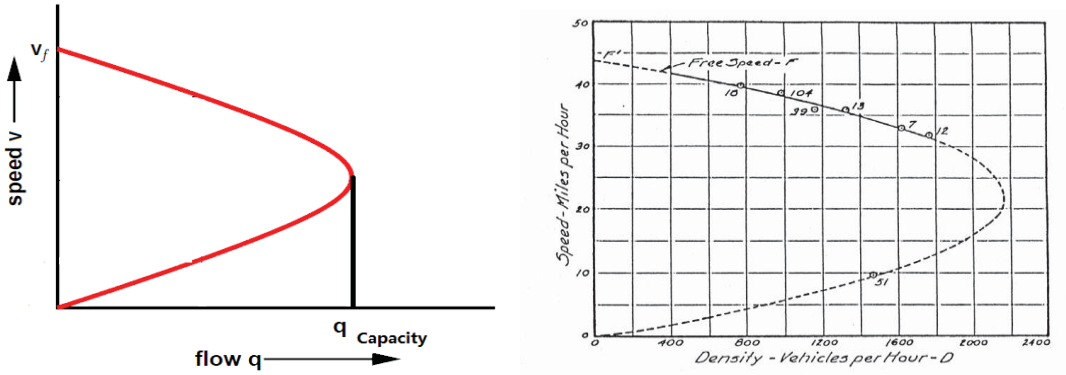


Figure 2.6 Greenshields' speed – density relationship (Greenshields, 1935)

Note that the term "Density – vehicles per hour" defined by Greenshields in Fig. 2.6 means traffic flow (q), since the term "flow" was not defined by traffic engineers in his time (Kuhne, 2008).

Later researchers conducted a great deal of empirical work to identify the speed-density relationship by field measurements (Banks, 1989, 1990; Chin & May, 1991; Hall & Hall, 1990). They all discovered a fact that speeds remain almost constant when traffic volumes are high. Banks (1991) reported there was 3% drop from flow capacity at bottleneck locations under high traffic volumes. Hall, Hurdle, and Banks (1992) summarized previous work and gave an empirical shape of speed-flow curve (Fig. 2.7).

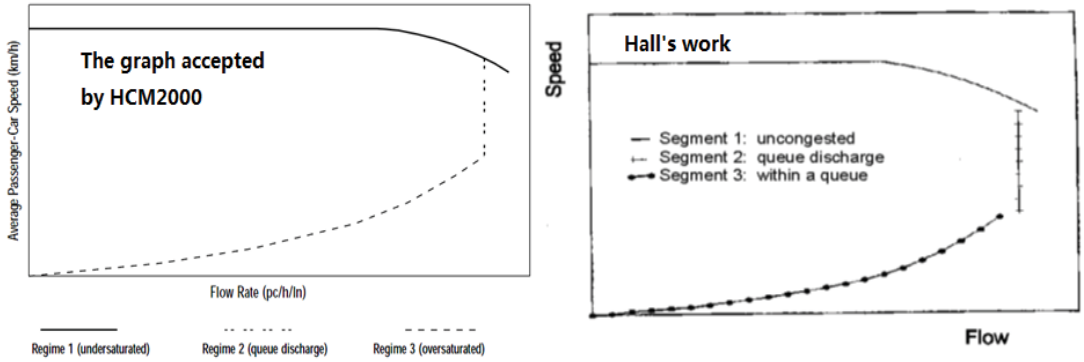


Figure 2.7 An empirical speed – density relationship (Hall et al., 1992)

Segment 1 and segment 2 of the curve can be easily supported by the results of most mentioned field studies above, even from those of earlier studies (Ceder & May, 1976; May & Keller, 1967). For segment 3 (the oversaturated flow), no empirical work can demonstrate the entire lower part of the curve in Fig. 2.7.

2.1.3 Flow - Density Relationship

As other extensive study of speed-density relationship, flow-density relationship can also be derived from it. Substituting $v = q/k$ to Eq. (2.1), we obtain

$$q = v_f k - (\frac{v_f}{k_j})k^2 \tag{2.8}$$

Eq. (2.8) is a parabolic curve derived from Greenshields' equation (Fig. 2.8).

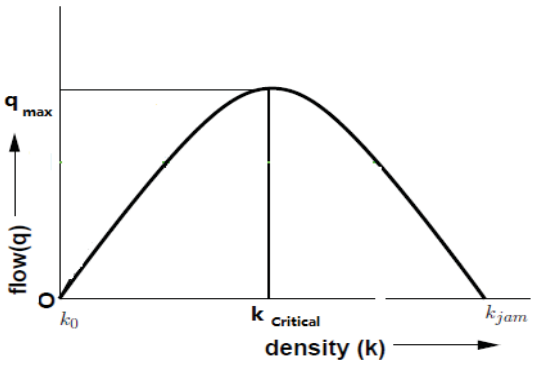


Figure 2.8 Greenshields' speed – density relationship

Where, $k_{critical}$ represents the critical density at which flows are maximum and k_{jam} represents the jam density at which flows tend to be zero. Edie (1961) pointed out that a gap occurred in flow-density data and in high-flow ranges. Easa (1982), Koshi, Iwasaki, and Ohkura (1983) suggested, on the basis of such a gap, that discontinuous functions are more appropriate to describe speed-density relationship

(Fig. 2.9).

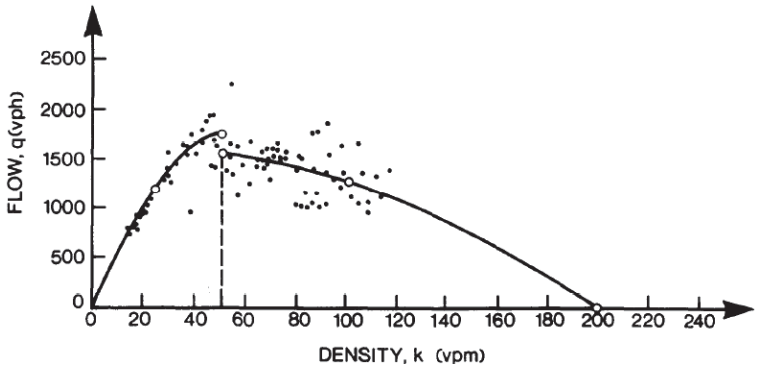


Figure 2.9 The assumed discontinuous function (Easa, 1982). Redrawn by Hall, Allen, and Gunter (1986)

Hall et al. (1986) conducted a very important empirical work to examine those early assumptions. Instead of converting occupancy data to more commonly used density data as most researchers did in their time (Koshi et al., 1983), they directly utilized raw occupancy for their study. They believed that such conversion may cause unnecessary uncertainty. Their work proposed four feasible configurations to fit the field data and define the flow-occupancy (or flow-density) relationship (Fig. 2.10).

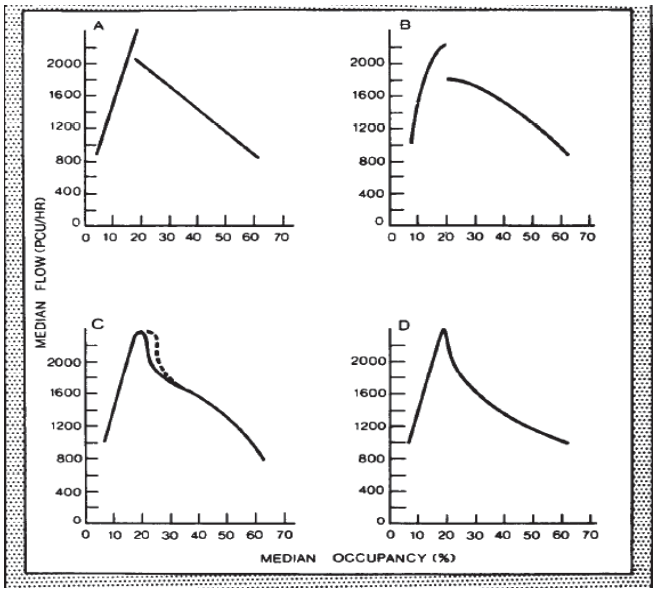


Figure 2.10 The feasible flow-occupancy relationship (Hall et al., 1986)

In Figure 2.10, the graph A is the reverse lambda shape. The graph B is a discontinuous function. The graph C is a continuously differentiable function. The graph D is a continuous but not continuously differentiable function. They concluded that the arguments of a discontinuous flow-occupancy (or flow-density) are not

convincing and suggested an inverted V curve for the flow-occupancy relationship. Banks (1989) confirmed the curve they proposed by using data from the San Diego area.

2.2 Continuum Traffic Flow Models

The fundamental relationships mentioned above reveal traffic stream characteristics in terms of the stationary relationships of aggregated quantities of flow, density and speed, so they are also named traffic stream models. The drawback of traffic stream models is that they cannot model traffic flow temporally and spatially (Gartner et al., 2005). Consequently, continuum traffic flow models are developed to describe the temporal-spatial evolution of those macroscopic flow quantities. The first order continuum models normally consist of two equations, conservation law and flow-density relationship.

2.2.1 The Conservation Law in Traffic Flow Theory

The continuum models deal with traffic flow as compressible fluid due to the analogy between vehicular traffic flow (Gartner et al., 2005). The traffic flow is presented by the macroscopic flow quantities: flow, density and speed, similar to fluid dynamics (Haberman, 1977). Those fluid-like quantities are regarded as the point quantities that are defined at all position points x along the roadway and evolved with time t. In this thesis, flow, density and speed are given by $q(x,t)$, $\rho(x,t)$ and $v(x,t)$ respectively, and they are related by

$$q(x,t) = \rho(x,t) \times v(x,t) \tag{2.9}$$

Assuming that no cars can be created or destroyed on a roadway segment and no entrances and exits are on this road too, as shown in Fig. 2.11.

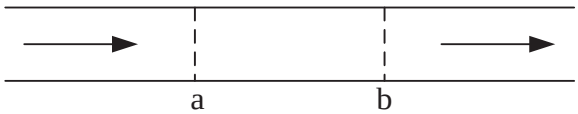


Figure 2.11 A segment of roadway

Then the number of cars in the region between a and b at time t on the roadway

segment can be given by

$$N(t) = \int_a^b \rho(x,t)dx \tag{2.10}$$

And the change of the numbers of cars on the roadway segment from $t = t_0$ to $t = t_1$ can be given by,

$$N(t_1) - N(t_0) = \int_{t_0}^{t_1} q(a,t)dt - \int_{t_0}^{t_1} q(b,t)dt \tag{2.11}$$

Take derivative on both sides with respect to t_1, we obtain

$$\frac{dN(t_1)}{dt_1} = \frac{d}{dt_1}\int_{t_0}^{t_1} q(a,t)dt - \frac{d}{dt_1}\int_{t_0}^{t_1} q(b,t)dt = q(a,\ t_1) - q(b,t_1) \tag{2.12}$$

Since t_1 represents any arbitrary time, it can be replaced by t. Then substituting Eq. (2.10) into Eq. (2.12), yields

$$\frac{d}{dt}\int_a^b \rho(x,t)dx = q(a,t) - q(b,t) \tag{2.13}$$

Considering a and b as independent variables, then Eq. (2.12) is rewritten as

$$\frac{\partial}{\partial t}\int_a^b \rho(x,t)dx = q(a,t) - q(b,t) \tag{2.14}$$

Take partial derivative on both sides with respect to b, we obtain

$$\frac{\partial}{\partial t}\frac{\partial}{\partial b}\int_a^b \rho(x,t)dx = \frac{\partial}{\partial b}q(a,t) - \frac{\partial}{\partial b}q(b,t)$$

$$\frac{\partial \rho(b,t)}{\partial t} = 0 - \frac{\partial}{\partial b}q(b,t)$$

$$\frac{\partial \rho(b,t)}{\partial t} + \frac{\partial q(b,t)}{\partial b} = 0 \tag{2.15}$$

Since b represents an arbitrary position on the road, it can be replaced by x. Thus, yields

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = 0 \tag{2.16}$$

Eq. (2.14) expresses an integral form of the conservation law and Eq. (2.16) represents a differential form of the conservation law (LeVeque, 1992). The second form is more often employed to formulate a macroscopic traffic flow model.

2.2.2 LWR Traffic Flow Model

Richards (1956), Lighthill and Whitham (1955) first combined Eq. (2.16) with the Greenshields' flow-density relationship, Eq. (2.8), to complete a simple traffic

continuum model. The model is also named as LWR model.

Eq. (2.16) can be rewritten as

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial q(x,t)}{\partial x} = \frac{\partial \rho}{\partial t} + \frac{\partial q}{\partial \rho} \times \frac{\partial \rho}{\partial x} = \frac{\partial \rho}{\partial t} + \frac{dq(\rho)}{d\rho} \times \frac{\partial \rho}{\partial x} = 0 \qquad (2.17)$$

In Eq. (2.17), $q(\rho)$ is the stationary flow-density relationship. It can be any flow-density relationship mentioned in Section 2.1.3, but it has to possess the following characteristics (Sun, 2005):

1. $q(0) = q(\rho_{jam}) = 0$, where, ρ_{jam} is the jam density.
2. $q(\rho)$ must be a differentiable concave function.
3. $q_{capacity} = q(\rho_{critical})$, where, $q_{capacity}$ is the maximum flow and $\rho_{critical}$ is the critical density.

Substituting the Greenshields' model (Eq. (2.8)) into Eq. (2.17), yields

$$\frac{\partial \rho}{\partial t} + \left(v_f - 2v_f \frac{\rho}{\rho_j}\right) \times \frac{\partial \rho}{\partial x} = 0 \qquad (2.18)$$

As can be seen in Eq. (2.18), LWR model is a fairly simple, though nonlinear, first order partial differential equation (PDE).

One of common ways to solve PDEs is to simplify the PDEs to be ODEs (ordinary differential equations) along the characteristics, which is called the characteristic method (LeVeque, 1992). This method can be used to solve LWR model by determining the characteristic curves in space-time plane (Gartner et al., 2005).

Let us consider the traffic density can be measured by a moving observer in a vehicle, whose position can be measured by $x = x(t)$. Then the traffic density can be given as $\rho = \rho(x(t), t)$. By applying the chain rule, we obtain

$$\frac{d\rho(x,t)}{dt} = \frac{\partial \rho}{\partial t} + \frac{dx}{dt} \times \frac{\partial \rho}{\partial x} \qquad (2.19)$$

Comparing Eq. (2.19) to Eq. (2.17), we find that

if $$\frac{dx}{dt} = \frac{dq(\rho)}{d\rho}, \qquad (2.20)$$

then $$\frac{d\rho(x,t)}{dt} = 0. \qquad (2.21)$$

Assuming all vehicles tend to remain same speed and keep same distance headway without any interference, and then we obtain $v = \frac{dx}{dt} = Constant = c$. The Eq. (2.20)

becomes

$$\frac{dx}{dt} = \frac{dq(\rho)}{d\rho} = c_1 \tag{2.22}$$

Integrating Eq. (2.22), yields

$$x = \frac{dq(\rho)}{d\rho} \times t + c_2 = c_1 \times t + c_2 \tag{2.23}$$

Since Eq. (2.22) and Eq. (2.21) together express Eq. (2.17) in ordinary differential forms, the curves represented by Eq. (2.23) in $x - t$ plane are characteristics of Eq. (2.17). Eq. (2.23) shows that the characteristics are straight lines with different x-intercepts, c_2. Eq. (2.21) implies ρ remain constant along those straight lines. Consequently, the density, $\rho = (x(t), t)$, can be calculated at any point of $x - t$ plane by looking for the initial densities, $\rho = (x(0), 0) = \rho_0$. Since the characteristics dominated by Eq. (2.23) only depend on traffic densities, ρ, the characteristics can also be interpreted as density waves propagating in $x - t$ plane with wave velocities, $\frac{dq(\rho_0)}{d\rho} = c_1$. If initial densities are assumed to be uniform along the roadway, the characteristics should be paralleled straight lines in $x - t$ plane. In other words, density waves are propagating in the same wave velocity.

When initial densities or initial distance headways along the roadway are not uniform, the density waves with different wave velocities may intersect in $x - t$ plane, and discontinuities occur, along which the crossing density waves terminate. Those mathematical discontinuities are also named as shock waves. The physical phenomenon of shock waves can be illustrated by Fig. 2.12.

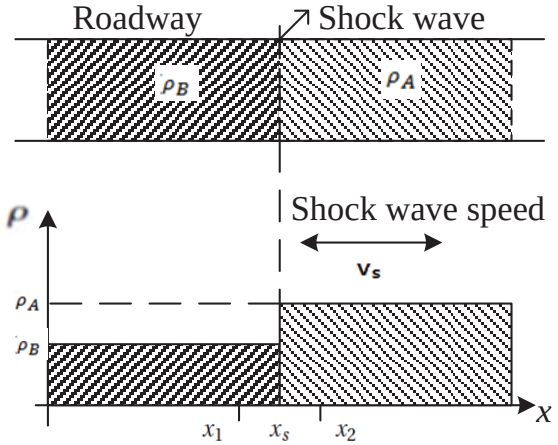


Figure 2.12 A shock wave on a roadway

In Fig. 2.12, traffic densities along a roadway segment are given by ρ_A and ρ_B, $\rho_B < \rho_A$, so a shock wave generate at a location (x_s) where ρ_B abruptly becomes ρ_A. The speed of the wave can be derived in the following manner.

Imagine two fixed observers stand at each side of x_s, x_1 and x_2. Their positions are infinitesimally near the shock. Applying an integral form of the conservation law, Eq. (2.13), to the portion of the roadway between two observers, yields,

$$\frac{d}{dt}\int_{x_1}^{x_2}\rho(x,t)dx = \frac{d}{dt}\int_{x_1}^{x_s}\rho(x,t)dx + \frac{d}{dt}\int_{x_s}^{x_2}\rho(x,t)dx = q(x_1,t) - q(x_2,t) \qquad (2.24)$$

Where,

$$\frac{d}{dt}\int_{x_1}^{x_s}\rho(x(t),t)dx = \frac{d}{dt}F(x_s,t) = \frac{d\,x_s}{dt}\frac{\partial F}{\partial\, x_s} + \frac{\partial F}{\partial t} \quad \left(\text{if } \int_{x_1}^{x_s}\rho(x(t),t)dx = F(x_s,t)\right)$$

$$= \frac{d\,x_s}{dt}\frac{\partial}{\partial\, x_s}\int_{x_1}^{x_s}\rho(x,t)dx + \frac{\partial}{\partial t}\int_{x_1}^{x_s}\rho(x,t)dx$$

$$= \frac{d\,x_s}{dt}\rho(x_s^-,t) + \int_{x_1}^{x_s}\frac{\partial}{\partial t}\rho(x,t)dx = \frac{d\,x_s}{dt}\rho(x_s^-,t) \qquad (2.25)$$

$$\frac{d}{dt}\int_{x_s}^{x_2}\rho(x,t)dx = -\frac{d\,x_s}{dt}\frac{\partial}{\partial\, x_s}\int_{x_2}^{x_s}\rho(x,t)dx + \frac{\partial}{\partial t}\int_{x_s}^{x_2}\rho(x,t)dx$$

$$= -\frac{d\,x_s}{dt}\rho(x_s^+,t) + \int_{x_s}^{x_2}\frac{\partial}{\partial t}\rho(x,t)dx = -\frac{d\,x_s}{dt}\rho(x_s^+,t) \qquad (2.26)$$

Both integrals vanish in Eq. (2.25) and Eq. (2.26), because x_1 and x_2 are infinitesimally near the shock. Combining Eq. (2.25) and Eq. (2.26) with Eq. (2.24), we obtain,

$$\frac{d\,x_s}{dt}\left(\rho(x_s^-,t) - \rho(x_s^+,t)\right) = q(x_1,t) - q(x_2,t) \qquad (2.27)$$

Rearranging Eq. (2.26), yields,

$$v_s = \frac{d\,x_s}{dt} = \frac{q(x_1,t)-q(x_2,t)}{\rho(x_s^-,t)-\rho(x_s^+,t)} = \frac{q(\rho_B)-q(\rho_A)}{\rho_B-\rho_A} \qquad (2.28)$$

Where, v_s is the speed of the shock wave in the LWR model (Haberman, 1977). Thus, the traffic situation in Fig. 2.12 can also be illustrated by characteristics and their shock wave in Fig. 2.13, in which the traffic densities along the roadway emanate from initial densities of ρ_B and ρ_A, propagating at the velocity of $\frac{dq(\rho_B)}{d\rho}$ and $\frac{dq(\rho_A)}{d\rho}$, terminating at the shock.

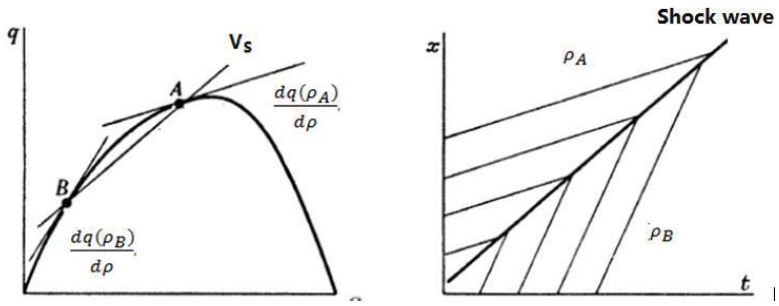


Figure 2.13 The shock wave in the fundamental diagram and t-x plane

The LWR model was implemented at signalized interchanges to analyze traffic queue dynamics by Michalopoulos and Stephanopoulos (1979, 1981). Their works demonstrate the formation and dissipation of queues under traffic signal control by using the method of characteristics (Fig. 2.14). Applications of shock waves are very instructive in understanding the formation of traffic congestion.

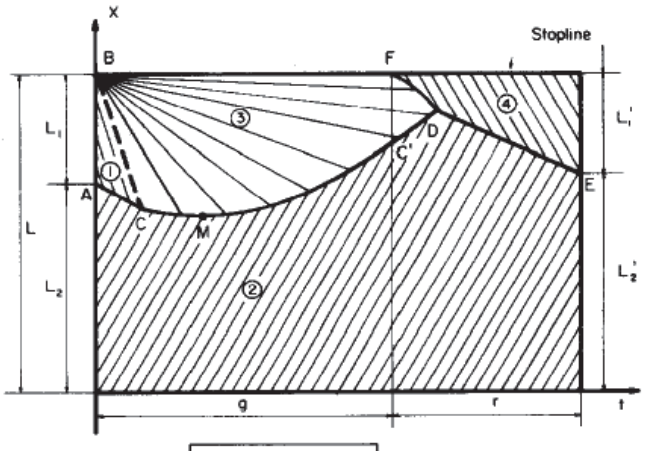


Figure 2.14 Shock wave formations(Michalopoulos & Stephanopoulos, 1979)

Because the characteristic method had difficulties to deal with complicated traffic conditions such as on ramp, off ramp or lane changing (Gartner et al., 2005), numerical methods were also employed to solve this simple continuum model. The differential form of the conservation law is discretized in time and space by finite difference methods, which still combines with the stationary flow-density relationship and leads to a discrete first-order traffic model.

Efforts on the applications of the discrete traffic model were mainly made by Michalopoulos, Beskos, and Lin (1984a, 1984b).

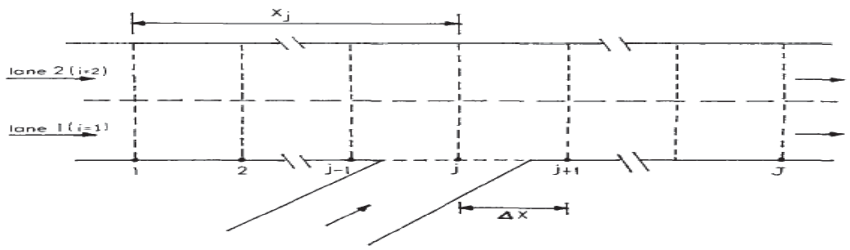


Figure 2.15 Space discretization for a two-lane freeway section (Michalopoulos et al., 1984a)

2.2.3 Second-order Traffic Flow Models

The LWR model is deficient for the description of some traffic features including (Daganzo, 1997):

- LWR model assumes that all drivers tend to remain the same speeds with respect to densities and ignores that driver behaviors are different.
- The zero width of shock wave implicates that drivers change their speed instantaneously without reaction time.
- LWR model cannot explain some traffic dynamic effects (also called traffic instability) such as stop-go waves (Ferrari, 1989) and the hysteresis phenomenon (Treiterer & Myers, 1974), which states that vehicles emerging from a bottleneck location have a retarded behavior compared to the behavior of those approaching the location.

All but the first flaw of the LWR model could be addressed by introducing a dynamic speed-density relationship to replace the stationary flow-density relationship in the LWR model. It together with the conservation law completes second-order systems of PDEs to describe traffic phenomenon, which are also named as second-order traffic flow models.

The dynamic speed-density relationship can be derived in the following manner. Payne (1971) and Whitham (1974) proposed a dynamic speed-density hypothesis by introducing a reaction time for drivers τ and an anticipated position Δx to the momentum equation, Eq. (2.29).

$$v(x, t+\tau) = V[\rho(x+\Delta x, t)] \tag{2.29}$$

Where, $V[\rho]$ is the stationary speed-density relationship. Eq. (2.29) implicates that drivers need reaction time to reach anticipated positions. Expanding the both sides of Eq. (2.29) in a Taylor series with respect to τ and Δx and neglecting the high order

terms, yields,

$$v(x,t) + \tau \frac{dv(x,t)}{dt} = V[\rho(x,t)] + \Delta x \frac{dV[\rho(x,t)]}{dx} \tag{2.30}$$

Where,

$$\frac{dv(x(t),t)}{dt} = \frac{dx}{dt}\frac{\partial v}{\partial x} + \frac{\partial v}{\partial t} = v\frac{\partial v}{\partial x} + \frac{\partial v}{\partial t} \tag{2.31}$$

$$\frac{dV[\rho(x,t)]}{dx} = \frac{d\rho(x(t),t)}{dx}\frac{dV}{d\rho} = \left(\frac{dt}{dx}\frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial x}\right)\frac{dV}{d\rho} = \left(0 \times \frac{\partial \rho}{\partial t} + \frac{\partial \rho}{\partial x}\right)\frac{dV}{d\rho} = \frac{\partial \rho}{\partial x}\frac{dV}{d\rho} \tag{2.32}$$

Substituting Eq. (2.31) and Eq. (2.32) into Eq. (2.30) and rearranging the combined equation, yields,

$$\frac{\partial v}{\partial t} = \frac{1}{\tau}(V[\rho] - v) - v\frac{\partial v}{\partial x} + \frac{\Delta x}{\tau}\frac{\partial \rho}{\partial x}\frac{dV}{d\rho} \tag{2.33}$$

Where, the space increment, Δx, can be given by $\Delta x = \frac{0.5}{\rho}$ (Payne, 1971); $\frac{dV(\rho)}{d\rho}$ is assumed to be an negative approximate constant (referring to section 2.1.1) and can be given by $\frac{dV(\rho)}{d\rho} = -\frac{c}{0.5} < 0$ (Papageorgiou, Blosseville, & Hadj-Salem, 1989). Then Eq. (2.33) can be rewritten as,

$$\frac{\partial v}{\partial t} = \frac{1}{\tau}(V[\rho] - v) - v\frac{\partial v}{\partial x} - \frac{c}{\tau\rho}\frac{\partial \rho}{\partial x} \tag{2.34}$$

On the left hand side of Eq. (2.34), the first term accounts for the relaxation effects in which drivers force the perturbed speed towards steady-state speeds, $V[\rho]$. The second term is the change of speed due to convection, so it implicates that drivers adapt their speeds to suit traffic conditions at different locations. The third term is propositional to the spatial change of densities, $\frac{\partial \rho}{\partial x}$, so it represents that drivers adjust their speeds according to anticipated downstream densities. Consequently, Eq. (2.34) expresses a dynamic speed-density relationship, which along with the conservation law, Eq. (2.16), leads a second-order traffic model named Paney's model.

Papageorgiou et al. (1989) compared three second-order models with LWR model. The one based on the Paney's model shows significantly higher accuracy than other models for the prediction of traffic conditions. Messmer and Papageorgiou (1990) presented the discretized form of this model, in which they added two extra terms to Eq. (2.34) for different traffic situations, including merging traffic near on-ramps and weaving traffic at lane drops. The numerical simulation program based on this discretized

model was named METANET (Messmer, 2000).

Daganzo (1997) pointed out that the second-order model could be flawed for two reasons. The first flaw he mentioned is that it could lead to negative speeds that is also known as the 'wrong way travel' problem. The second critique is that the speed of characteristics of the Paney's model are sometimes faster than actual traffic speeds, which implicates that drivers are affected by the traffic conditions behind them. Papageorgiou (1998) explained that the negative speeds can be practically solved by setting a positive minimum value for speeds, and that the speeds in the macroscopic models are average speeds rather than individual speeds so there is no physical contradiction even if the Panye's model gives the higher speed of characteristics.

Liu, Lyrintzis, and Michalopoulos (1998) fixed negative speeds by introducing density-dependent reaction time $\tau(\rho)$ and Zhang (2002) proposed a new model whose speed dynamics is derived from a revised momentum equation, to remove both of the deficiencies. However, their models have no published discretized forms and no available simulation programs, so implementations of them are few.

2.3 Ramp Metering

Ramp metering or on-ramp control has been considered as the most cost-effective way to alleviate motorway congestion (Papageorgiou & Kotsialos, 2002). By limiting vehicles to access motorways ramp metering can potentially achieve reduced motorway congestion in space and time, increased motorway throughput, reduced congestion spillback to the adjacent urban traffic network or to other merging motorways, and improved traffic safety due to reduced congestion duration (Chaudhary, Tian, Messer, & Chu, 2004). Ramp control strategies can be classified as fixed time control, local traffic responsive control and coordinated ramp metering control (Bogenberger & May, 1999).

The first two strategies are only used to optimize the local traffic condition by controlling a single on-ramp. The coordinated control strategy is used to optimize the traffic conditions over traffic networks by controlling a number of on-ramps

simultaneously.

2.3.1 Fixed Time and Local Responsive Ramp Metering Algorithms

Fixed-timed ramp control normally generates a ramp signal that operates at constant time cycle for a specific time of day, usually rush hour. The ramp metering rates are preset based upon historical data that could be years, months, or days old.

The local responsive ramp metering generates ramp metering rates to adapt to the local traffic condition in the vicinity of the on-ramps. The control strategies can deliver appropriate metering rates every few minutes based on the changes of densities or flow data detected from loop detectors.

The fundemental digrams play a very essential role in the design of local responsive ramp metering. The terms like ‘crtical denstiy’ and ‘flow capacity’ in traffic stream models are often involved in the ramp metering algorithms. Three most noticeable local ramp metering strageties are intoduced in this section, incuding demand-capacity control, ALINEA (Papageogiou, Hadj-Salem, & Blosseville, 1991) and fuzzy logic control (Bogenberger, 2000; Taylor, Meldrum, & Jacobson, 1998)

Demand-capacity control is the most straightforward local ramp control strategy (Fig. 2.16). Metering rates are generated based on the comparison between the measured upstream flow data and the downstream flow capacity. The idea of this strategy is trying to maintain the downstream flow below the flow capacity in the fundamental diagram (Fig. 2.8).

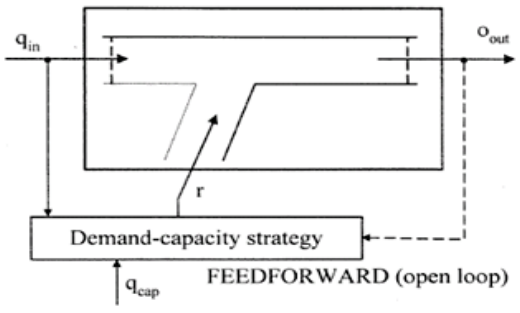


Figure 2.16 Demand-capacity control (Papageorgiou & Kotsialos, 2002)

The on-ramp inflow is caculated by Eq. (2.35):

$$R(t) = C - q_{in}(t-1) \tag{2.35}$$

Where, t is the time interval to update the metering rate and normally set to one minute. $R(t)$ is the number of vehicles allowed entering motorway in the period t. C is the downstream flow capacity, which has to be identified by the flow-density relationship (Fig. 2.8). $q_{in}(t-1)$ is the upstream flow in the previous time interval $t-1$. The main disadvantage of this strategy is that the generated metering rates are very unstable and oscillate with the change of the upstream flow rates that is very sensitive to the system disturbances, such as unexpected slow vehicles or merging difficulties.

Instead of maintaining the downstream flow below the flow capacity, ALINEA (Asservissement Linéaire d'Entrée Auotroutière) is designed to stabilize the downstream occupancy (density) blow the critical occupancy (density) in Fig.2.8. Different with the demand-capacity control, ALINEA is a feedback control algorithm (Fig. 2.17), so the previous time metering rate has to be used as the input for the next iteration, which prevents the generated metering rates from large-amplitude oscillations.

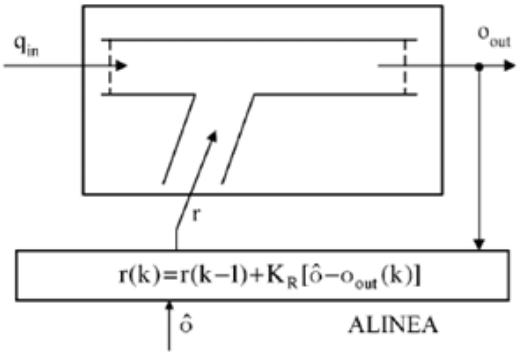


Figure 2.17 ALINEA algorithm (Papageorgiou & Kotsialos, 2002)

The metering rate is calculated by the Eq. (2.36):

$$R(t) = R(t-1) + k[o_{critical} - o_{out}(t)] \tag{2.36}$$

Where, k is a tuneable parameter. $o_{critical}$ is the downstream critical occupancy (density) identified by the flow-occupancy (density) relationship. o_{out} is the detected downstream occupancy. Since ALINEA only need the downstream occupancy to compute the metering rates, it lacks consideration of upstream traffic conditions and

ramp queues, so it might generate too strict metering rates.

Fuzzy logic control (FLC) has been introduced to the design of ramp metering algorithms due to its expandable rule-based systems, which ensures that traffic conditions in the vicinity of on-ramps can be all be involved in the computation of metering rates. A typical layout of FLC ramp metering proposed by Bogenberger (2000) is shown as flows.

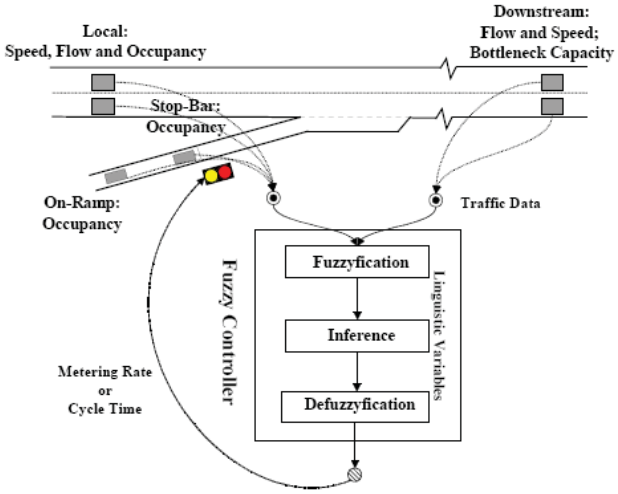


Figure 2.18 FLC ramp metering (Bogenberger, 2000)

In Fig. 2.18, local traffic conditions could be monitored by detectors installed at four different locations, including upstream detectors, downstream detectors, queue detectors at the end of on-ramps and check-in detectors in front of stop lines. All detected information is fuzzified as inputs. Fuzzy relation between inputs and outputs (or metering rates) is defined by rules, a list of if-then pairs. The fundamental diagrams are very instructive in making such a knowledge-based system (Taylor & Meldrum, 2000).

2.3.2 Coordinated Ramp Metering

For the coordinated ramp metering, the ramp metering algorithms can be further classified as model-free control strategies or model-based control strategies. The former strategies rely on empirical assumptions from output data of traffic systems, such as capacity estimations from flow measurements (Geroliminis, Srivastava, &

Michalopoulos, 2011) or the observed repeatable historical data (Hou, Xu, Yan, Xu, & Xiong, 2011; Hou, Yan, Xu, & Li, 2012).

For the later strategies, macroscopic traffic flow models are involved to formuate the on-ramp control problems. A model-predictive hierarchical control approach for ramp metering can be shown in Fig. 2.19 (Papamichail, Kotsialos, Margonis, & Papageorgiou, 2010).

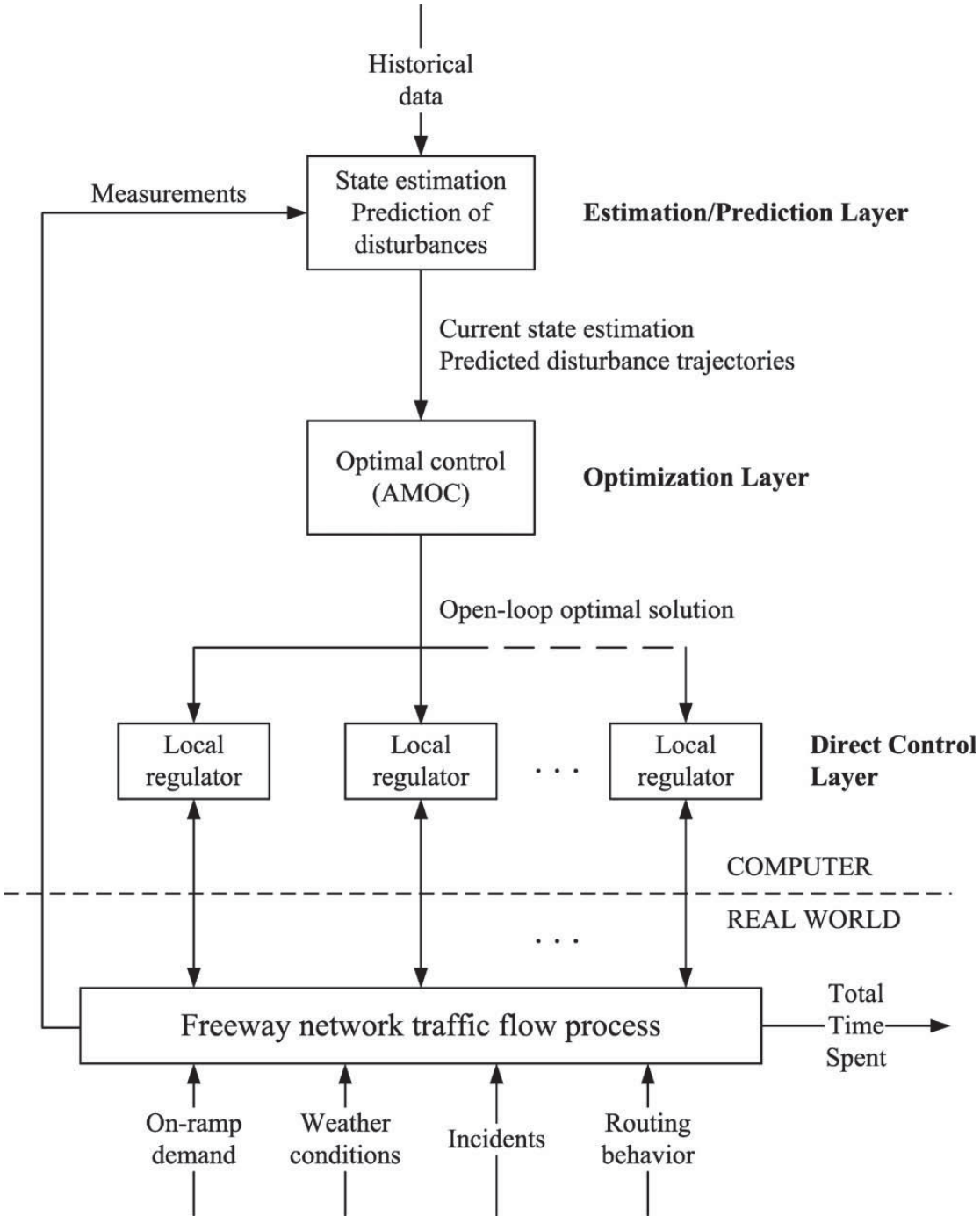


Figure 2.19 The hierarchical control strategy (Papamichail et al., 2010)

In Fig. 2.19, historical data are used to calibrate the macroscopic traffic model and real-time measurements are processed to provide the initial traffic states. An optimization method for Advanced Motorway Optimal Control (AMOC) is employed to find the optimal traffic states in terms of Total Time Spent (TTS) along the time-horizon and the corresponding optimal control trajactory (or the optimal inflow of on-ramps). The optimizaiton procedure is placed under the framework of receding horizon control whereby each time the procedure starts with new measurements and the previous prediciton errors are rejected.

Among many macroscopic traffic models, a discreted second-order model (Papageorgiou et al., 1989) derived from the Payne's model (Payne, 1971), was found to be computationally efficient and accurate for modelling a large scale motorway network (Kotsialos, Papageorgiou, Diakaki, & Middleham, 2002). In many on-line ramp control methods (Bogenberger & Keller, 2001; Kotsialos & Papageorgiou, 2004; Kotsialos et al., 2002; Papamichail & Papageorgiou, 2008), the model was employed to formulate the optimal ramp control problems and used as simulation envirament. The numerical program of the model named METANET is released by Messmer and Papageorgiou (1990). The manual of METANET (Messmer, 2000) provode the practical methods to model the multi-origin and multi-destination motorway networks with different geometric characteristics including bifurcations, junctions, on-ramps and off-ramps. An objective function in terms of TTS, Eq. (2.37), was also proposed to compare the efficiency of various control strategies, which is widely used in model-based coordinate ramp control strategies today.

$$J(k_0) = TTS_{Motorway} + TTS_{Queue} + PenaltyFunctions \qquad (2.37)$$

In which, $TTS_{Motorway}$ is the total time spent by all vehicles travelling on motorway; TTS_{Queue} is the total time spent by all vehicles waiting in the queue; the penalty functions are normally for constrains of ramp queue lengths and speed limits.

Maximization of the throughput of the motorway network is equivalent to minimization of the total time spent on the motorway network (Papageorgiou, 1983), so the optimal ramp control problem can be formulated as a nonlinear constrained minimization problem of Eq. (2.37) for model-based ramp control strategies. This

problem had been solved by computational intelligence (CI) methods, e.g., genetic algorithms, neuro-fuzzy logic algorithms (Bogenberger & Keller, 2001; Bogenberger, Vukanovic, & Keller, 2001; Ghods, Kian, & Tabibi, 2009) or game-theoretic approach (Ghods, Fu, & Rahimi-Kian, 2010); the problem was also solved by gradient-based methods, e.g., feasible-direction algorithm (Kotsialos & Papageorgiou, 2004; Papageorgiou & Marinaki, 1995), sequential quadratic programming (Bellemans, 2003) and approximate dynamic programming (Zhao, Xu, & Yu, 2011). The proposed algorithms above were only implemented and evaluated in the Paney-based macroscopic models. Implementations in microscopic simulators were not seen. The optimal ramp signals released by those algorithms are continuous, which needs a significant amount of computational efforts for real-time control.

In this thesis, DP decision networks was proposed for the minimization problem and solved by the forward recurrence method (Fang & Elefteriadou, 2006; Luus, 2002). Up to 37 discreated metering rates for each ramp meter can be obtained by minimizing the cost function of TTS values. The optimal metering policy was applied under the receding horizon framework (Baskar, Schutter, & Hellendoorn, 2012; Kouvaritakis, Cannon, & Rossiter, 1999; Zegeye, Schutter, Hellendoorn, Breunesse, & Hegyi, 2012). The proposed algorithm was implemented and evaluated in a microscopic simulator.

2.4 Microscopic Traffic Models

In contrast to macroscopic traffic models that treat traffic dynamics as aggregated macroscopic quantities of flow, density and speed, microscopic traffic models simulate individual vehicle dynamics that represent microscopic properties such as speeds and positions, based on the models of driver behaviors like car-following models (Fritzsche, 1994; Gipps, 1981; Wiedemann & Reiter, 1992; Q. Yang & Koutsopoulos, 1996) and lane changing models (Moridpour, Sarvi, & Rose, 2010).

2.4.1 Car-Following Models

Car-following models describe the fundamental interaction between leading vehicles and

following vehicles in the same lane, which are the most fundamental microscopic models. Notations and definitions are summarized in Fig. 2.20.

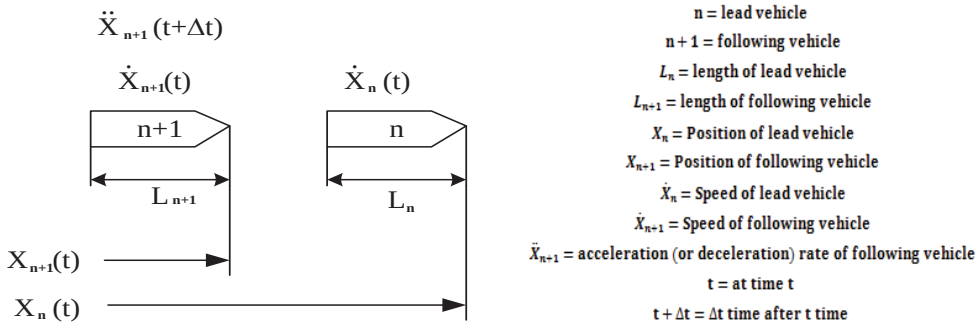


Figure 2.20 Notations and definitions for car-following theory

The models often used in today's microscopic traffic simulators are commonly classified into three classes, General Motors' theories or GHR (Gazis-Herman-Rothery) models, safety-distance models and psycho-physical models.

- GHR models

  GHR models developed by the researchers associated with General Motors group (Chandler, Herman, & Montroll, 1958; Gazis, Herman, & Rothery, 1961) are properly the most well-known car-following models. The researchers proposed five generations of car-following models, all of which took the form

  Response = function (sensitivity, stimuli)

  This form implies that the acceleration or deceleration (response) of the following vehicle is proportional to the relative speed (stimuli) of lead and following vehicle (May, 2000). The significant contributions of the GHR models are not only the discovered stimulus-response mechanism but the mathematical bridge between microscopic and macroscopic traffic models. May and Keller (1967) demonstrated that all proposed traffic stream models were related to almost all GHR models. Take the fifth GHR model, Eq. (2.38), as an example.

$$\ddot{X}_{n+1}(t+\Delta t) = \frac{a_{l,m}[\dot{X}_{n+1}(t+\Delta t)]^m}{[X_n(t)-X_{n+1}(t)]^l}[\dot{X}_n(t) - \dot{X}_{n+1}(t)] \qquad (2.38)$$

  In which, $a_{l,m}$ is the sensitivity term. The work of May and Keller (2000) showed that when $l\ and\ m$ values were 0 and 1, 0 and 2, and 1 and 2, the GHR model was equivalent to Greenberg, Greenshields and Underwood traffic stream models,

respectively.

- Safety-distance models

Pipes (1967) characterized the motion of vehicles by stating the following rules, namely: " *A good rule for following another vehicle at a safe distance is to allow yourself at least the length of a car between your vehicle and vehicle ahead for every tem miles per hour of speed at which you are travelling.*" The resulting equation can be shown as follows,

$$\text{Minium distance headway} = [X_n(t) - X_{n+1}(t)] = \mathrm{L}_n \left[\frac{\dot{X}_{n+1}(t)}{(1.47)(10)}\right] + \mathrm{L}_n$$

Pipe's equation is properly the most simple safety-distance model, which is different from the stimulus-response type of car-following models. Kometani and Sasaki (1959) proposed the earliest safety-distance model. The model describes the basic relationship between lead and following vehicles by specifying a safe following distance to avoid a collision when the front vehicle decelerates heavily. Major development of this model was made by works of Gipps (1981). In Gipps model, vehicles have two statuses, free or constrained by the front vehicles. When constrained, the following vehicle tends to adjust the speed to reach the safe distance headway to response any reasonable actions from the lead vehicle. When free, the following vehicle tries to reach the maximum acceleration. It has been found that Gipps model produces the realistic behaviors of propagations of queues for a platoon of vehicles (Brackstone & McDonald, 1999).

- Psycho-physical models

The major deficiency of stimulus-response type of models is that the following vehicle has to response to any actions of the lead vehicle as long as the relative speed is no zero even if the distance headway is large. Psycho-physical models can remove such a deficiency by introducing the threshold where drivers tend to change their behaviors. Michaels (1963) proposed the first psycho-physical model, in which two types of thresholds, the thresholds of the perception of the relative speed and the thresholds of the distance headway, were given to define the driver behaviors into different regimes, such as following, free driving and closing-in. However, it is

difficult to confirm the validation of this model since very little work has been undertook on this model since the sixties (Brackstone & McDonald, 1999).

The development of car-follow models, focusing on the microscopic properties of interactions between vehicles, leads to a new perspective of modeling traffic flow, microscopic traffic simulation.

2.4.2 Microscopic Simulators

Microscopic traffic models also named traffic micro-simulation models are primarily the integrations of car-following models with many other driver-behavior models, such as lane changing models, gap-acceptance models and overtaking models (Olstam & Tapani, 2004). The numerical programs of the microscopic traffic models are named microscopic traffic simulators. The microscopic simulators can produce more detailed and realistic traffic features, comparing to macroscopic traffic models, at the expense of increasing the computational effort enormously. The popular software packages of microscopic simulators include Aimsun (TSS, 2008), Paramics (PARAMICS, 2000) and VISSIM (VISSIM, 2000), in which, Aimsun is established on a safe-distance model, Paramics and VISSIM on GHR and Psycho-physical models (Olstam & Tapani, 2004)

Microscopic simulators continuously model the interactions between each vehicle and its surrounding traffic as well as road environments throughout the duration of the traffic simulation, based on the driver-behavior models. States of vehicles and detectors in the simulators are continuously updated over the simulated time which has been split into fixed time intervals named simulation steps, while the states of traffic signals are discretely updated at a preset time by users. The latest simulators can distinguish the different types of vehicles whose driver behaviors are also defined differently. Most traffic devices present in a real traffic network can also be modeled in the microscopic simulator, including traffic lights, traffic detectors, Variable Message Signs, ramp meters, etc.

The required inputs to microscopic simulators are the geometric information for road

networks, traffic demands at origins as well as turning proportions at destinations. Other input data may include for example control plans for signalized intersections or ramp metering rates at on-ramps. Common outputs are macroscopic measures such as speed, density, flow and occupancy, which are defined in the beginning of this chapter.

Microscopic simulators such as Aimsun or Paramics have been used often as traffic simulation environments to test innovative algorithms for traffic management in many research areas (Abdelfatah, Ramadan, & Darwish, 2012; Chu, Liu, & Recker, 2001; Fang & Elefteriadou, 2008; Fehon & Klim, 2010). For the study of ramp metering, although local traffic responsive ramp metering algorithms had been implemented and evaluated in some previous research (Abdelfatah et al., 2012; Chu, Liu, Recker, & Zhang, 2004), the implementations of model-based coordinated ramp metering in microscopic simulators were not seen.

2.5 Summary

The early studies of the speed-density relationship pioneered in the development of traffic stream models. The single-regime models defined the speed-density curve with one continuous function over the complete range of flow conditions, while multi-regime models described it with discontinuous functions on the basis of the observation of the gap from the obtained field data. Although speed-flow relationships and flow-density relationships can be derived from the speed-density relationships, a great deal of empirical work have indicated that the derived relationships were incompatible with the field measured data. Meanwhile, the proposed empirical curves of speed-flow relationships and flow-density relationships have proved that multi-regime models were not convincing and appropriate single-regime models were called for. Unfortunately, there is not such a mathematical model that is compatible with all speed-flow-density curves obtained from field measurements.

The continuum traffic flow models were developed to describe the temporal-spatial evolution of macroscopic flow quantities studied in traffic stream models. The

analytical solutions of the first-order continuum model (LWR model), consisting of conservation law and a stationary flow-density relationship, revealed the formation and dissipation of queues under traffic signal control by shock waves but failed to explain some important traffic features for uninterrupted traffic flow. The second-order models, consist of conservation law and a dynamic speed-density relationship, were developed to remove the deficiencies of the first-order model. The criticism on second-order models (Payne's model) mainly pointed to 'wrong way travel' problem and the physical contradiction caused by faster speed of characteristics. Although Papageorgiou had countered the critiques in his article, later researchers still made a great effort to improve models. Latest models corrected the two problems mentioned above, but not much work had been undertook for implementations of the models, so Payne's model still dominated the field of macroscopic traffic flow modeling even after it had been proposed for more than thirty years.

Three types of ramp-metering schemes were proposed in the most publications: fixed time, local traffic responsive and coordinated traffic responsive ramp metering. Fixed-timed ramp metering was developed to generate constant ramp signals that operated at a specific time of day, usually rush hour, while local traffic responsive control was designed to adapt the metering rates to the traffic condition in the vicinity of the ramp. Coordinated ramp metering was designed to optimize traffic flow over a section of the motorway by controlling a number of ramp meters.

Except fixed-timed control, most current ramp strategies were established on the implementations of fundamental diagrams and macroscopic traffic flow models. For local responsive control, the fundamental diagrams addressed its importance in a practical way. For example, the downstream flow capacity in Demand-Capacity control and the preset critical occupancy in ALINEA can actually be identified by the flow-density relationship. For the coordinated ramp metering, macroscopic traffic models were playing an essential role in the formulation of optimal ramp control problems. Many noticeable works based on Payne's model can be found from recent publications. The evaluation and implementation of the proposed algorithms were all

based on Payne-based models.

As the parallel efforts of macroscopically modeling traffic flow (from traffic stream models to traffic continuum models), the development of car-following models led to microscopic traffic simulators, collective models of driver behaviors. Since they produced more detailed and realistic traffic behaviors than macroscopic models did, they have replaced macroscopic models as traffic simulation environments to test innovative algorithms in many recent publications.

In this thesis, we proposed a DP based method for the optimal coordination of ramp metering. The objective function was formulated by a macroscopic model based on the manual of METANET. Its performance had been measured in both the macroscopic and microscopic environments (Aimsun), where a 6.7km stretch of Auckland Northern Motorway had been chosen as the study location.

CHAPTER 3 OPTIMIZATION METHODOLOGY

In this chapter, a new model-based ramp control strategy was proposed. The employed macroscopic traffic model was briefly described with an objective function for the optimization criterion. The ramp control problem was depicted by a Dynamic Programming (DP) decision network, whose optimal metering rates can be found along the optimal trajectory acquired by forward recurrence method (Luus, 2002) .

3.1 Motorway Traffic Flow Model

The employed traffic flow model is a second-order macroscopic model based on Paney's model (Payne, 1971). It has been employed extensively in traffic engineering tasks to predict the short-time traffic condition and evaluate traffic control strategies (Messmer, 2000). The simulation program of this model is known as METANET (Messmer & Papageorgiou, 1990).

3.1.1 Non-destination Oriented Model

There are two versions of METANET programs, namely the destination oriented model and the non-destination oriented model. The former considers the traffic assignments of different destinations, while the latter has no such consideration and the traffic flows leaving the network are simply treated by splitting rates (the percentage of exiting traffic). In this thesis, the latter will be used as the traffic prediction model for the proposed algorithm and introduced in this subsection.

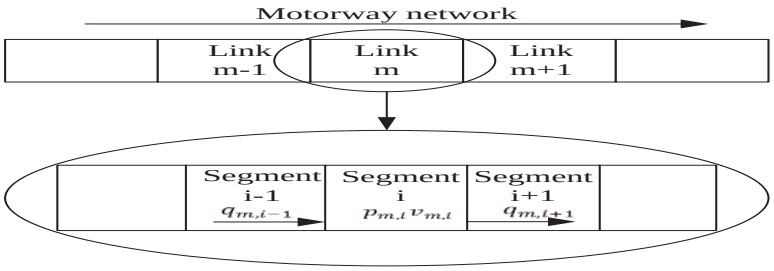


Figure 3.1 Links and segments in the macroscopic model

The motorway network in this model is divided by links, and each link has different

road characteristics from the adjacent links and can be subdivided into small segments that share the same road characteristics (Fig. 3.1)

The traffic flow is discretized in time and space, with the length of the discretized space being represented by $L_{m,i}$ and the interval of the discretized time by T_S, where *m* and *i* represent the link index and the segment index. The following equation must be hold for every segment in the whole network to avoid the numerical instability (Messmer & Papageorgiou, 1990).

$$L_{m,i} > v_{free} T_s \tag{3.1}$$

Where, v_{free} is the free speed of the link *m*. More notations and definitions are given in Table 3.1.

Table 3.1 Notations and definitions for the employed macroscopic model

Symbol	Unit	Description
m		*link index*
i		*segment index*
l		*simulation step counter*
$L_{m.i}$		*the length of the i-th segment of link m*
T_s	***second***	*the duration of one simulation step*
$v_{free,m}$	***km/h***	*the free flow speed in link m*
$\rho_{m,i}(l)$	***veh/km/lane***	*traffic densities at i-th segment of link m at simulation step l*
$v_{m,i}(l)$	***km/h***	*space mean speeds at i-th segment of link m at simulation step l*
$q_{m,i}(l)$	***veh/h***	*traffic flow rates at i-th segment of link m at simulation step l*
n_m		*the number of lanes in link m*
n_o		*the number of lanes in the o-th on-ramp*
$q_{in,m,i}(l)$	***veh/h***	*traffic inflow into the i-th segment of link m at simulation step l*
$q_{out,m,i}(l)$	***veh/h***	*traffic outflow out of the i-th segment of link m at simulation step l*
$V(\rho_{m,i}(l))$	***km/h***	*steady-state speeds at the i-th segment of link m at simulation step l, which is given by the function of the stationary speed-density relationship*
a_m		*the fitting parameter of the speed-density relationship for link m*
t_m	***hour***	*the tuning parameter (a time constant) of the speed relaxation term*
c_m	***km²/h***	*the tuning parameter of the anticipation term*

k_m	***veh/h***	*the tuning parameter of the anticipation term*
δ_m		*the tuning parameter of the merging term*
$\emptyset_m$		*the tuning parameter of the weaving term*
$\rho_{crit,m}$	***veh/km/lane***	*the critical density of link m*
$\rho_{jam,m}$	***veh/km/lane***	*the jam density of link m*
$q_{on,o}(l)$	***veh/h***	*traffic volumes allowed to access the motorway from the o-th on-ramp at simulation step l*
$Q_{cap,o}$	***veh/h***	*on-ramp capacity of the o-th on-ramp*
$D_o(l)$	***veh/h***	*traffic demands of the o-th on-ramp at simulation step l*
$w_o(l)$	***veh***	*the queue length of the o-th on-ramp at simulation step l*
$r_o(l)$	***veh/h***	*ramp metering rates of the o-th on-ramp at simulation step l, which normalizes a value between 0 and $Q_{cap,o}$* no to a range from 0 to 1*

The traffic states, including speed, density and flow, in every segment are calculated by Eq. (3.2)-(3.5).

$$\rho_{m,i}(l+1) = \rho_{m,i}(l) + \frac{T_s}{n_m L_{m,i}}[q_{in,m,i}(l) - q_{out,m,i}(l)] \tag{3.2}$$

$$\begin{aligned} v_{m,i}(l+1) = v_{m,i}(l) & \\ + \frac{T_s}{L_{m,i}} v_{m,i}(l)[v_{m,i-1}(l) - v_{m,i}(l)] & \quad \text{Convection} \\ + \frac{T_s}{t_m}[V[\rho_{m,i}(l)] - v_{m,i}(l)] & \quad \text{Relaxation} \\ - \frac{c_m T_s[\rho_{m,i+1}(l) - \rho_{m,i}(l)}{t_m L_{m,i}[\rho_{m,i}(l) + k_m]} & \quad \text{Anticipation} \end{aligned} \tag{3.3}$$

$$q_{m,i}(l+1) = \rho_{m,i}(l+1) v_{m,i}(l+1) \tag{3.4}$$

$$V[\rho_{m,i}(l)] = v_{free,m} exp(-\frac{1}{a_m}\left(\frac{\rho_{m,i}(l)}{\rho_{crit,m}}\right)^{a_m}) \tag{3.5}$$

Where, Eq. (3.2) is the discretized form of conservation law (referring to Eq. (2.16)). Eq. (3.4) is the fundamental relationship of flow-density-speed that is held in all continuum models. Eq. (3.5) is the stationary speed-density relationship (referring to Eq. (2.6)) of link *m*, and it gives steady-state speeds also named as equilibrium speeds, at which drivers tend to restore their perturbed speeds.

Eq. (3.2) is the discretized form of the dynamic speed-density relationship (referring to Eq. (2.34)) which consists of the current traffic speed $(v_{m,i}(l))$, and three other terms as follows:

- *Convection term* describe a traffic phenomenon where the vehicles coming from the upstream segment i-1 will gradually adjust their speed to fit the current speed of the segment i.
- *Relaxation term* takes into consideration the fact that every vehicle in the segment is trying to reach its equilibrium speed of the segment.
- *Anticipation term* states a fact that drivers on segment i will adjust their speed by observing the downstream density (the density of segment i+1)

t_m ,c_m and k_m are the adjustable parameters that need to be identified by field data for the link *m*, and a_m and $v_{free,m}$ are the constants to be identified by the speed-density diagram for the link *m*.

For Eq. (3.3), there are two optional terms that could be added in according to the different traffic situations (Messmer & Papageorgiou, 1990). The first term is used when lanes drop between two links (Fig. 3.2), called the weaving term (Eq. (3.6)).

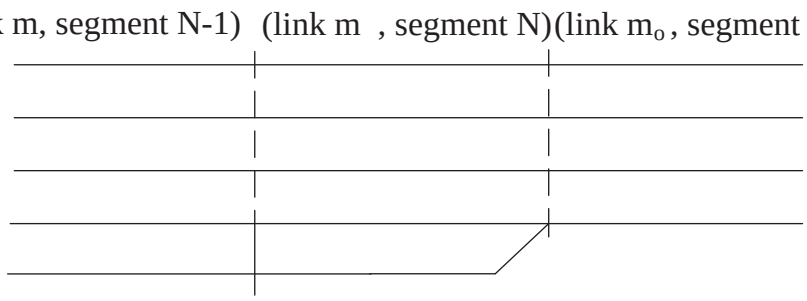


Figure 3.2 Weaving traffic

$$\Delta v_{weav,m,N}(l) = -\frac{\emptyset_m T_s}{l_{m,N}\ n_m}\left[\frac{(n_m - n_{m_0})\rho_{m,N}(l)}{\rho_{crit,m}}\right] v_{m,N}^2(l) \quad (3.6)$$

Eq. (3.6) is only used for lane drops so it gives negative value to express the fact that vehicles slow down once weaving occurs. The higher the upstream density ($\rho_{m,N}$) is, the stronger the effect of negative impact of weaving traffic is. The term being proportional to the square of the upstream speed ($v_{m,N}^2$) implicates that speed may decrease dramatically at weaving phenomenon when vehicles travelling at high

speeds.

The second term is called the merging term (Eq. (3.7)), which is added to Eq. (3.3) when an on-ramp is included in the segment (Fig. 3.3).

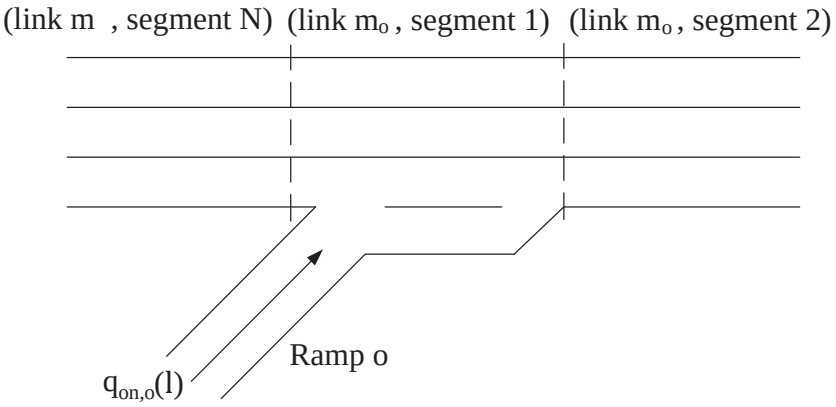


Figure 3.3 Merging traffic

$$\Delta v_{merge,m_o,1} = -\left(\frac{\delta_{m_o} T_s}{l_{m_o} n_{m_o}}\right) \frac{q_{on,o}(l) v_{m_o,1}(l)}{\rho_{m_o}(l) + k_{m_o}} \tag{3.7}$$

That the merging term from Eq. (3.7) is proportional to the on-ramp inflow ($q_{on,o}(l)$) and the mainline speed ($v_{m_o,1}(l)$) can be intuitively understood. The higher amount of traffic approaches the motorway from the on-ramp and higher the current mainline speed is, the more obviously the mainline speed decreases at the next simulation step. That the merging term is inversely proportional to the mainline density (ρ_{m_o}) is because higher (lower) the mainline density is, lower (higher) the mainline speed is and less (more) obvious the change of the mainline speed is at the next simulation step.

$q_{on,o}(l)$ in Eq. (3.7) represents traffic volumes allowed to access the motorway from the o-th on-ramp at simulation step l, which is calculated by:

$$q_{on,o}(l) = Min[D_o(l) + \frac{w_o(l)}{T_s}, Q_{cap,o} Min\left(1, \frac{\rho_{jam,m_o} - \rho_{m_o,1}(l)}{\rho_{jam,m_o} - \rho_{crit,m_o}}\right)] \tag{3.8}$$

Two terms are involved in the calculation of $q_{on,o}(l)$. The first term, including the ramp demands ($D_o(l)$) and the queue length ($w_o(l)$), represents the amount of traffic that are trying to access the motorway. The second term represents the amount of traffic able to access the motorway that is assumed to be monotonically decreasing with the mainline density. Eq. (3.8) can also be expressed by Fig. 3.4.

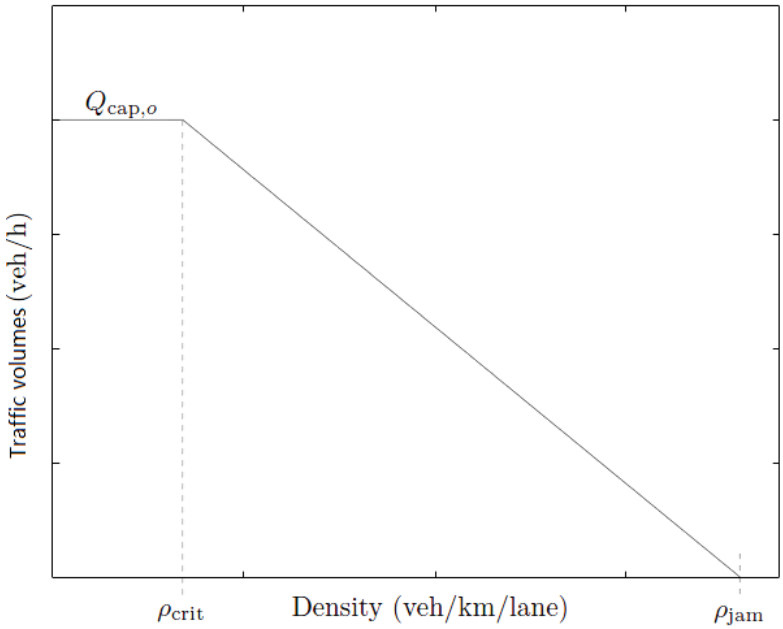


Figure 3.4 Traffic volumes allowed accessing the motorway from an on-ramp

$w_o(l)$ in Eq. (3.7) is given by:

$$w_o(l+1) = w_0(l) + T_s(D_o(l) - q_{on,o}(l)) \tag{3.9}$$

Note that Eq.(3.7) does not take into account ramp meters, so when ramp o is under the control of any ramp strategy, it is given by:

$$q_{on,o}(l) = Min[D_o(l) + \frac{w_o(l)}{T_s}, Q_{cap,o} Min\left(r_o(l), \frac{\rho_{jam,m_o} - \rho_{m_o,1}(l)}{\rho_{jam,m_o} - \rho_{crit,m_o}}\right)] \tag{3.10}$$

Where, $r_o(l)$ is the ramp meter rate that is defined by an interval [0, 1]. The actual ramp meter rate for the implementation can be calculated by the on-ramp capacity (Q_{cap}).

$$\text{ramp meter rate} = r_o(l) Q_{cap} n_o \tag{3.11}$$

Eq. (3.8) and (3.9) are also used to compute the origin queue length at the first segment of network when the traffic inflow from the origin exceeds the road capacity of the first link and a virtual queue forms from outside the network.

For the last segment of traffic network, the anticipation term of Eq. (3.3) is replaced by Eq. (3.12), when the density of that segment exceeds the critical density of the link, to ensure that the downstream of the traffic network is always under free flow conditions.

$$-\frac{c_m T_s[\rho_{critical}(l) - \rho_{m,i}(l)]}{t_m L_{m,i}[\rho_{m,i}(l) + k_m]} \tag{3.12}$$

And the anticipation term is removed when the density of that segment is less than the

critical density of the link, which assumes that the downstream density remains the same.

3.1.2 Objective Function

In this thesis, the coordinated ramp metering control problem is formulated as a minimization problem of the total time spent (TTS) by all vehicles in the motorway network. The objective function is given by:

$$J(l_0) = T_s \sum_{l=1}^{L_p-1} \sum_{m} \sum_{i} \rho_{m,i}(l) L_m n_m + T_s \sum_{l=1}^{L_p-1} \sum_{o} w_o(l) + T_s \sum_{l=1}^{L_p-1} \sum_{o} [r_o(l) - r_o(l-1)]^2 + T_s \sum_{l=1}^{L_p-1} \sum_{o} a_o \left[max\{0, w_o(l) - w_{max,o}\right]^2 \quad (3.13)$$

Here the first term is the total time vehicles spent on the motorway ($TTS_{Motorway}$). Where, L_p is the number of total simulation steps; l is the simulation counter; $w_o(l)$ is the queue length on the ramp o, and the ramp meter rates are included in the equation of $w_o(l)$. The second term is the total time vehicles spent in the queue at the upstream road segment of the origin and on-ramps (TTS_{Queue}). The third term is to ensure that decision variables will not have wide swings between short time intervals. The fourth term is a penalty term that restricts the maximum queue lengths of each on-ramp ($w_{max,o}$), where a_o is a weighting factor for each ramp that determines the balance between TTS performance and queue constrain violation as well as the level of importance of each on-ramp. The decision variable vector, $r_o(l)$ (ramp meter rates), is constrained by $r_o(l)_{min} < r_o(l) < 1$.

3.2 A DP based Ramp Metering Strategy

Dynamic programming (Bellman & Dreyfus, 1966) is a very powerful method for optimization of an engineering system that can be described as a finite set of discrete

states and separated into stages along the time horizon. One state is associated with a set of feasible actions (or decisions) linked to the corresponding states of next stage. The changes of system states over the time horizon can be found by the states linked by the trajectory of decisions along the time stages. For an optimal control problem, optimizing such a system is to identify the optimal trajectory of decisions along the time stages according to the certain optimization criterion.

For ramp control problems, a DP decision network can be formed when a traffic system can be considered as a number of discrete traffic states and separated by time stages. The state transitions between two neighboring stages are determined by feasible actions (ramp metering rates) at the corresponding states. Optimal metering rates can be obtained by searching the optimal trajectory of actions (or decisions), along which TTS values returned from each stage are minimized.

3.2.1 A Generic DP Decision Network

Take a motorway stretch with 3 on-ramps as example. If ramp meter rates are defined by an interval [0, 1] and the control resolution of ramp metering rates are defined by 0.05, for a motorway stretch with 3 on-ramps, the possible ramp metering rates are,

$$Ramp_1 = [0.05, 0.1, \dots, 0.95]^T$$

$$Ramp_2 = [0.05, 0.1, \dots, 0.95]^T$$

$$Ramp_3 = [0.05, 0.1, \dots, 0.95]^T$$

According to Eq. (3.11), those ramp metering rates actually represent

$$Ramp \quad = [45, 90, \dots, 855]^T$$

Where, the on-ramp capacity is given by 900vehicle /km/lane. Each ramp meter may have a total of 19 values (released metering rates).

The coordinated ramp control problem can now be depicted by a forward recurrence DP decision network as shown in Fig. 3.5.

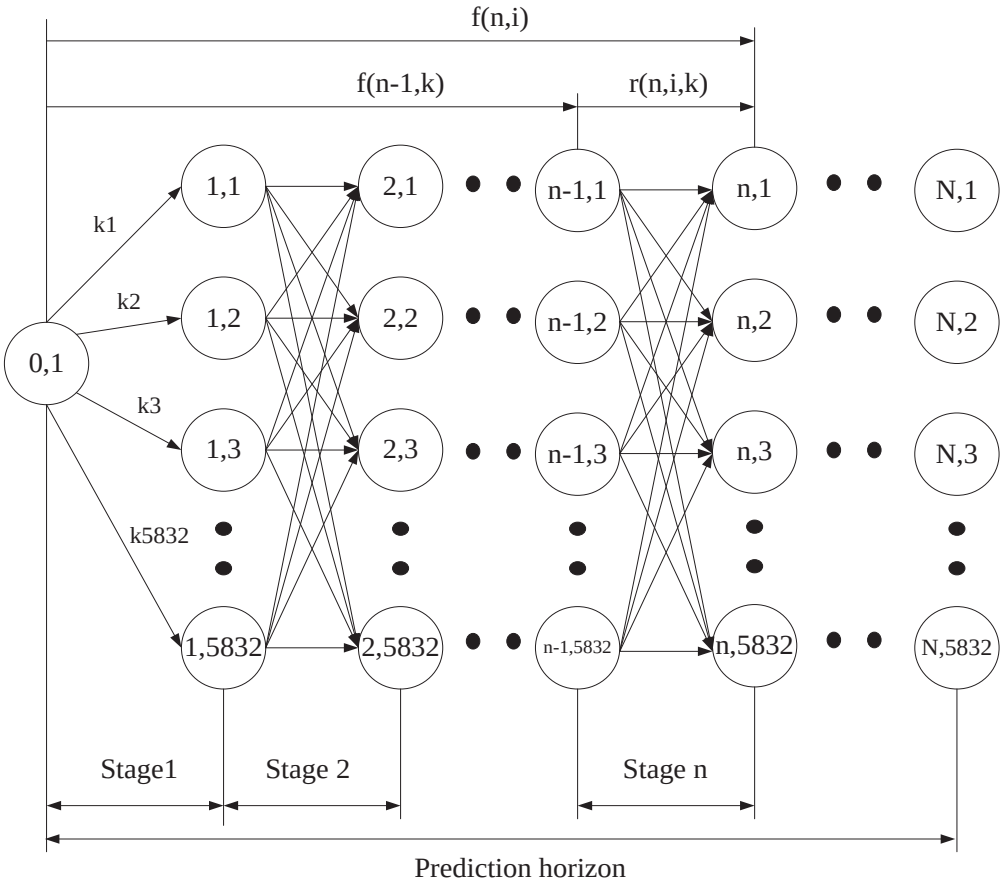


Figure 3.5 A generic DP decision network for ramp control problems

The network nodes stand for traffic states of the motorway stretch, which consists of:

- Speed, density and flow rates of each segment on the motorway stretch, Eq. (3.2)-(3.4).
- The queue length for each on-ramp, Eq. (3.9).
- The virtual queue length outside the motorway from the origin, Eq. (3.9).
- The ramp metering rates for all on-ramps $[Ramp_1, Ramp_2, Ramp_3]$.

The two numbers in the node represent the index of states and the index of stages respectively. For example, the node (0, 1) represents the initial traffic states, where 0 being stage 0 and 1 being state 1. Each stage represents one control step (a time interval to update metering rates), so the total number of stages represents the entire time horizon (or prediction horizon). Except stage 0, each stage includes $18^{the\ number\ of\ on-ramps} (= 18^3 = 5832)$ traffic states associated with all combinations of the ramp metering rates. One state transits to any state of the next stage

due to a decision k (ramp metering rates) represented by an arrowed line. A decision k is composed of three ramp meter rates for three on-ramps, denoted by $Ramp_1$, $Ramp_2$, $Ramp_3$.

The forward recurrence relation (Hillier, 2001) is given by,

$$f(n,i) = \min_{k \in K}[\, r(n,i,k) + f(n-1,k)] \tag{10}$$

Here, *f (n,i)* represents the minimum TTS value at a given state *i* from stage 0 to stage *n* . $r(n,i,k)$ is the return function which returns the TTS values from the last stage (*n-1*) to the *i* –th state of the current stage *n* via decision *k*. The TTS value during the transition of two neighbouring states is calculated by Eq. (3.13).

The notations and symbols for the forward recurrence DP in Fig. 3.5 are listed in Table 3.2.

Table 3.2 Notations and definitions for DP decision network

Terminology	Symbol	Description
Stage	***n***	*n represents the index of a stage. One stage means one control step (2mins).*
State	***(n,i)***	*n and i are the indexes of traffic states and time stages, and (n,i) represents the i-th traffic states at the n-th stage*
Decision variable	***k € K***	*k corresponds to ramp metering rates, by which a traffic state of next stage can be approached, composed by* $[ramp_1, ramp_2, .., ramp_m]$. *K are all feasible decision variables associated with states.*
Immediate return	***r(n,i,k)***	*The TTS value calculated from last stage (n-1) to the i-th state of current stage n via decision k, which is calculated by Eq. (3.13).*
Optimal value of a state	***f(n,i)***	*Minimum TTS value at a given state i, from stage 0 to stage n.*

This generic decision network ensures that each traffic state may transit to any state of the next stage. However, it also implicates some unrealistic trajectory might be found resulting in unstable metering rates between two neighbouring stages. For example, if the state transition between one state with a ramp meter rate of 0.1 and other state of next stage with the ramp meter rate is allowed to be 0.9, the ramp meter rates would

change from 90vehicles/hour/lane to an implausible 810 vehicles/hour/lane in one control step.

In order to overcome this drawback and save excessive computational load, the optimal decision policy can be found by employing the concept of iterative dynamic programming (Luus, 2002). The solution procedure is broken into two phases of search. During the first phase, a simplified decision network is proposed to search a near-optimal trajectory with a resolution of meter rate of 0.1 and each ramp meter may have 9 released metering rates, where one state can only move to neighbouring states (not any state) at the next stage to avoid excessive oscillations. At the second phase, the generic decision network is constructed around the results from the first phase of search to find a finer solution with a resolution of meter rate of 0.025. After two phases of searching, each ramp meter can eventually have 37 released metering rates with minimum control resolution, 0.025.

For a local ramp meter, when the numbers of released metering rates are no less than ten, the effect of the discretization can be negligible (Kotsialos et al., 2006), so a discretization with 37 released rates is considered sufficient for a ramp meter. Also, the second phase of search is optional, if the first phase of search is adequate.

3.2.2 The First Phase of Search

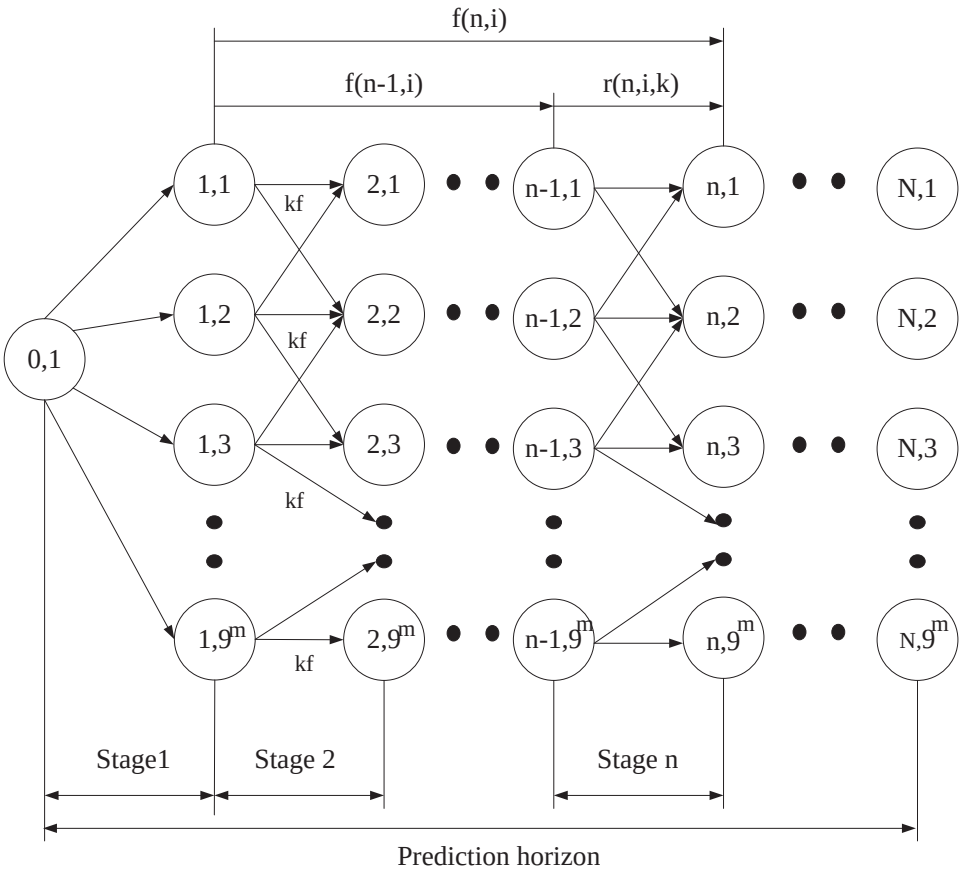


Figure 3.6 A simplified DP decision network for approximate optimal trajectories

Provided that a motorway stretch has *m* on-ramps and ramp metering rates for each on-ramp are represented by $Ramp_m$, which is defined within a range of 0.8 equally divided by 9 released metering rates (control resolution of 0.1) and given by

$$\mathrm{Ramp_m} = [0.1, 0.2, \ldots, 0.9]^{\mathrm{T}} \tag{11}$$

The coordinated ramp control problem can now be depicted by a simplified forward recurrence DP decision network (Fig. 3.6).

Each time stage represents one control step, and the horizon is made up of N stages. Except initial stage that only includes the initial traffic state, each of the remaining stages includes 9^{m} traffic states associated with all combinations of the ramp metering rates. One state can approach other states of the next stage by the feasible decision variables (k_f), which are composed of $[Ramp_1, Ramp_2, .., Ramp_m]$.

Instead of approaching all states of next stage, the decision variables from current states to the states of next stage are restricted by ensuring that the change of ramp meter rates between neighboring stages is no more than 0.1 (Fig. 3.7).

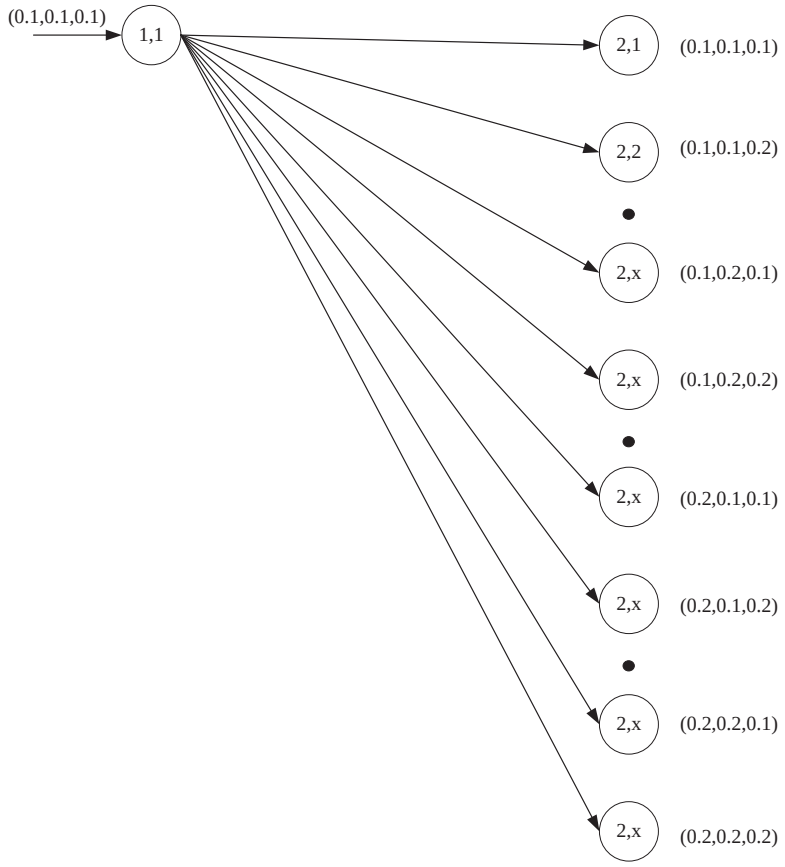


Figure 3.7 An example for feasible decision variables

Take Fig 3.7 as an example , if m=3 and State (1,1) are linked by decision variable, [0.1, 0.1, 0.1], then decision variables linked to the adjacent states of next stage are defined by[0.1, 0.1, 0.1], [0.1, 0.1, 0.2], [0.1, 0.2, 0.1], [0.1, 0.2,0.2], [0.2, 0.1, 0.1], [0.2, 0.1, 0.2], [0.2, 0.2, 0.1] and [0.2, 0.2, 0.2].

This decision network can be read as a shortest path problem with N stages and 9^m states over a region of [0.1, 0.9] per stage. By solving this decision network, an approximate optimal trajectory can be achieved, along which, a series of optimal decision variables with control resolution of 0.1 can be found.

3.2.3 The Second Phase of Search

The second phase of search attempts to search for a finer optimal trajectory along a

series of contracted regions based on the first approximate optimal trajectory. Released metering rates for each ramp meter are defined within a range of 0.1 with 5 released rates by using finer control resolutions of 0.025. The generic DP decision network has been employed in this search to ensure that each traffic state may transit to any state of the next stage, since the range for this search is small (0.1) and it is unnecessary to concern the oscillation between two neighboring stages.

After the approximate optimal trajectory from the first phase of search has been achieved, then decision variables along this trajectory can be considered as the center value of the new regions for the second search.

For example, assuming that decision variables at the first stage, k_{19} = [0.1, 0.3, 0.1], are found in the first optimal trajectory after the first phase of search is completed. By using [0.1, 0.3, 0.1] as center values of a range of 0.1, the decision variables at the first stage for the second phase of search have to be redefined by

$$\mathrm{Ramp}_1 = [0.05, 0.075, 0.1, 0.125, 0.15]^{\mathrm{T}}$$

$$\mathrm{Ramp}_2 = [0.25, 0.275, 0.3, 0.325, 0.35]^{\mathrm{T}}$$

$$\mathrm{Ramp}_3 = [0.05, 0.75, 0.1, 0.125, 0.15]^{\mathrm{T}}.$$

The new regions for the second search at each stage are constructed based on decision variables along the first optimal trajectory at each corresponding stage. Subsequently the decision network can be reconstructed for the second phase of search, in which, the control resolution is given by 0.025, and each stage includes 5^m states. One state can approach any state of next stage by decision variable k. Within further contracted regions, the refined optimal trajectory can be obtained quickly with sufficient accuracy.

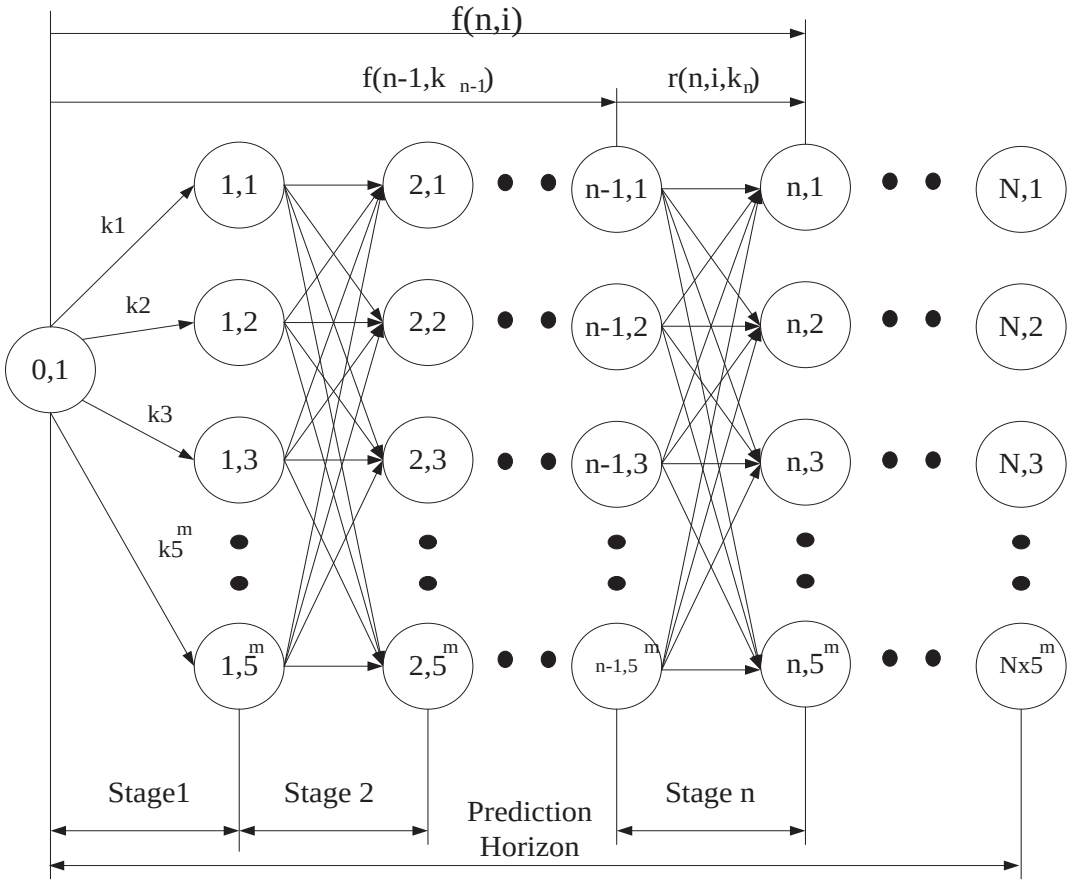


Figure 3.8 A generic DP decision network for a finer optimal trajectory

3.3 Summary

This chapter proposed a model-based coordinate ramp metering strategy. The employed model was a macroscopic model based on the non-destination oriented model of METANET, where the motorway had been broke into links subdivided into segments. Traffic states of each segment were described by aggregate variables: speed, flow and density. The queues from on-ramp and the virtual queue from the origin were calculated based on the mainline density. An objective function in term of TTS was also given for the optimization criterion.

A generic DP decision network was proposed for the optimal coordination of ramp metering, where the traffic states of a motorway stretch was depicted by a finite set of state nods linked by decision variables (ramp metering rates) and separated along time stages. Optimizing the decision network by the forward recurrence method to

minimize the TTS values at each stage can find the optimal trajectory of decision variables, along which optimal metering rates can be acquired for the entire time horizon. The limitation of this method was to find some unrealistic trajectories that caused unstable metering rates in a short period of time. In order to overcome the drawback, a method of two phases of search was proposed. The first search was trying to find an approximate optimal trajectory by a simplified DP decision network where each state nod can only reach neighboring states at next stage by feasible decisions. The second search was optional and expecting to find finer trajectory based on the near-optimal trajectory from the first search. The concept of the two phases of search was actually the original idea of IDP (iterative dynamic programming).

CHAPTER 4 MACROSCOPIC TRAFFIC SIMULATION

This chapter implements the DP based ramp metering algorithm in a macroscopic traffic simulation environment, where the traffic prediction model and the simulation environment were the same and no prediction errors were considered. The performance of the proposed approach algorithm was measured in terms of TTS and compared to the no-ramp-control case and a local responsive ramp metering algorithm, ALINEA. A 6.7km motorway stretch with 3 on-ramps in Auckland Northern Motorway was chosen as the study location and constructed as the simulation scenario by the same macroscopic model in Section 3.1.1, where the proposed ramp metering algorithm was implemented under the framework of receding horizon control.

4.1 Framework of the Simulation Study

The proposed algorithm was implemented as shown in Fig. 4.1.

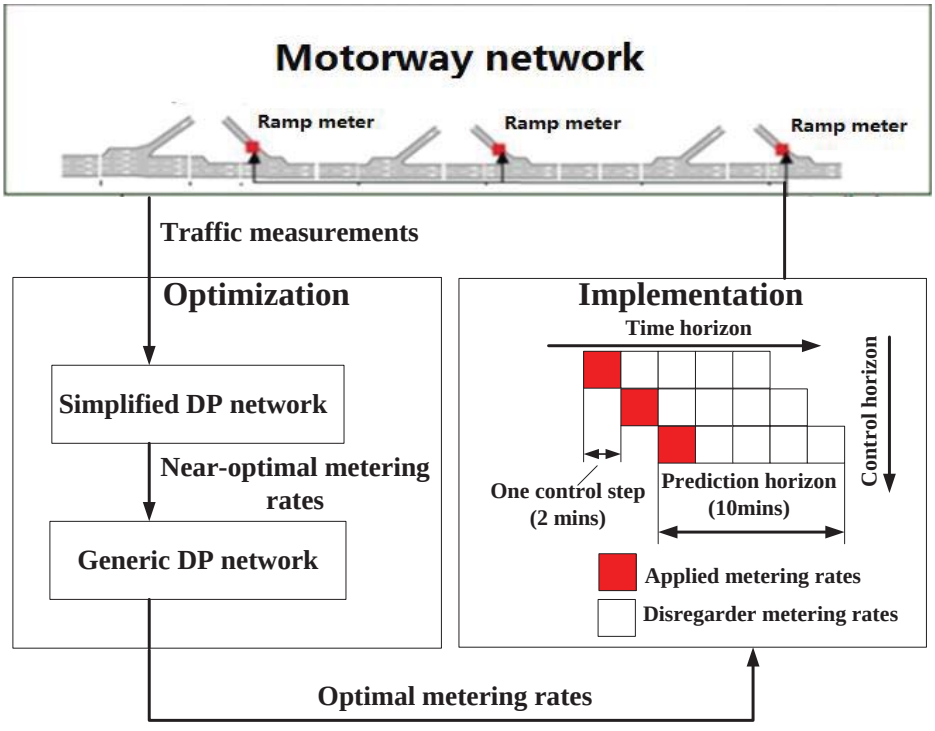


Figure 4.1 The framework of the implementation in a macroscopic traffic model

In Fig. 4.1, in order to acquire the optimal metering rates, DP decision networks are constructed to search (two phases of search) decision variables that minimizes TTS

values over a time horizon (prediction horizon). A simplified DP decision network is constructed first (Fig. 4.2). The traffic system is described by 9^3 discrete states that are separated along a time horizon of 10 minutes and one control step is given by 2 minutes (in other words, 9 released metering rates for each ramp meter are generated every 2 minutes), so the time horizon is divided into five stages. Initial traffic states are obtained from the real-time measurements of the motorway. Once the near-optimal decision trajectory is found, a generic decision network is constructed to search finer optimal trajectory with 37 released metering rates for each ramp meter (Fig. 4.3).

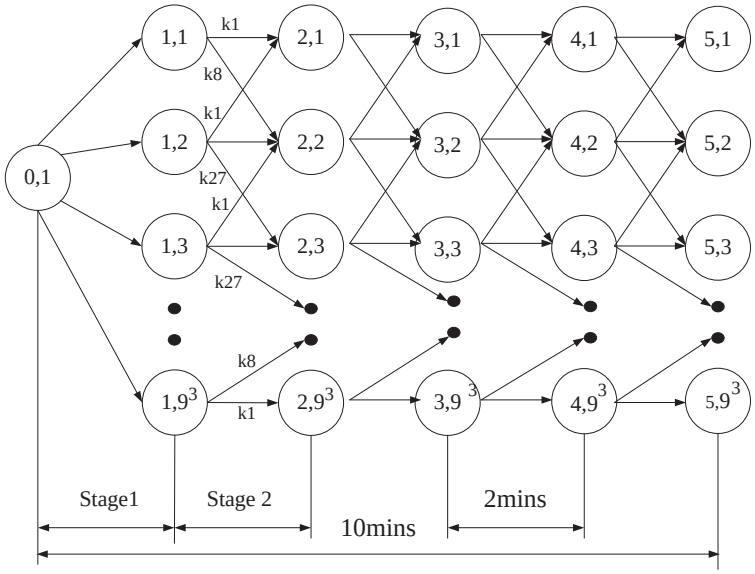


Figure 4.2 The simplified DP decision network for the simulation study

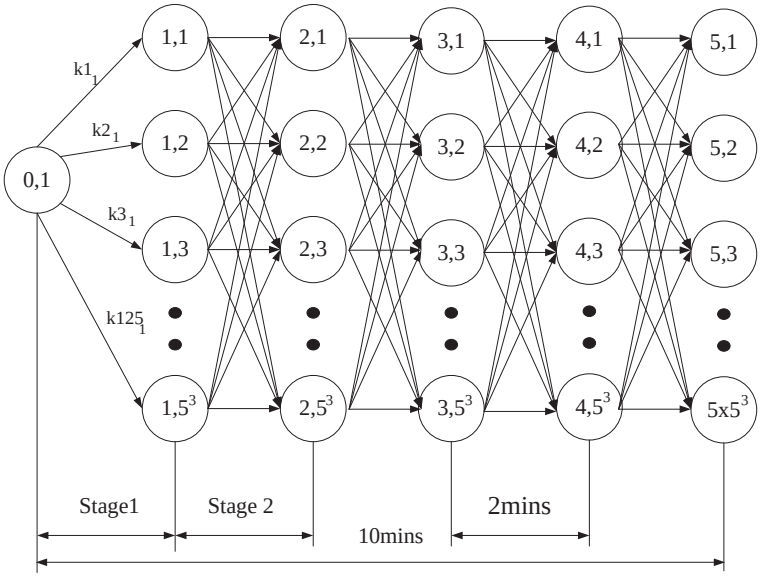


Figure 4.3 The generic DP decision network for the simulation study

The traffic states along the optimal trajectory are resulting from the previous decision and previous traffic states, calculated by the macroscopic traffic model (Section 3.1.1). The forward recurrence method is employed to ensure that the optimal policy is constituted with regard to TTS values minimized at each stage.

The entire prediction horizon includes five stages, but only decisions in the first stage are actually implemented to the motorway. After one control step, the horizon rolls forward for a stage, as indicated in Fig. 4.1 and the entire process starts over with the new traffic measurements so the previous prediciton errors would not be accumlated. This rolling horizon concept is known as the receding horizon control.

In order to study the performance differences between the near-optimal trajectory and optimal trajectory, the proposed algorithm was implemented twice. For the first time, only the first search with a simplified DP network (Fig. 4.2) was implemented and the second search with a generic DP decision network had been ignored, so each ramp meter can only have 9 released metering rates. For the second time, DP conducted two phases of search so each ramp meter can have 37 released metering rates.

4.2 Simulation Environment

4.2.1 Study Location

The study area to implement the proposed algorithm is a 6.7km motorway stretch ranging from the upstream southbound off-ramp of Grenville interchange to the downstream southbound on-ramp of Tristram interchange in the Auckland Northern Motorway (Fig. 4.4). The study area includes three interchanges, which are Greville interchange (1), Constellation interchange (2) and Trisram interchange (3). Each interchange includes one two-lane on-ramp and one two-lane off-ramp.

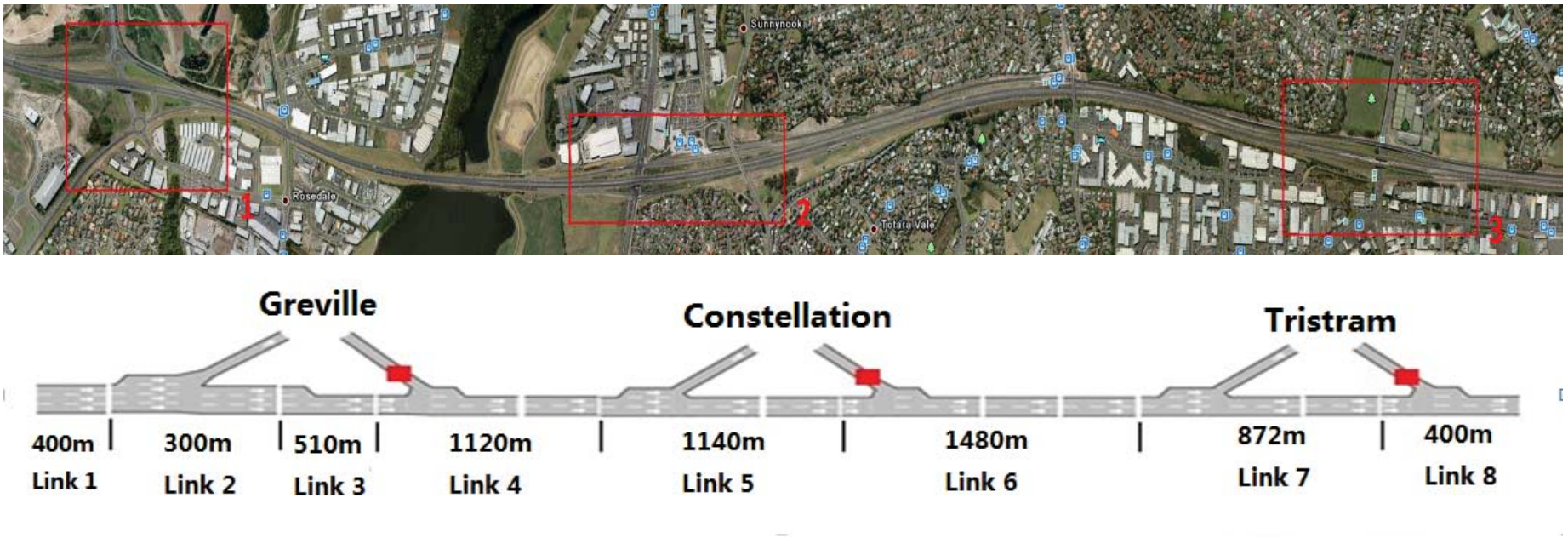


Figure 4.4 Study area constructed in a macroscopic traffic model

4.2.2 Simulation Setup

The macroscopic model in Section 3.1 was employed to construct the simulation environment. The motorway network was divided into 8 links subdivided into 13 segments as shown in Fig. 4.4. The length of the segments ranged between 310m and 550m, and the simulation step (T_s) was set to 5 seconds, which along with v_{free} =100 km/h enables Eq. (3.1) hold for every segment of the entire motorway network. Also note that traffic states calculated in DP decision networks was also based on the same macroscopic model with the same simulation step of 5 seconds. In other word, the traffic prediction model and the simulation model were the same one.

The geometric and traffic details of the network links were measured from Google Maps and given in Table 4.1. The global network parameters were recommended values from documentation of METANET (Messmer, 2000) and given in Table 4.2. All initial ramp queues and the queue from origin were set to 0. The duration of simulation was three hours from 6am to 9am.

Table 4.1 The information of network links in the macroscopic model

Name	Lane number	Capacity (Vehicles/lane)	Critical Density (Vehicles/km/lane)	Length (km)	Segment number
Link1	*3*	*2100*	*31.4*	*0.4*	*1*
Link2	*3*	*2100*	*31.4*	*0.3*	*1*
Link3	*2*	*2100*	*31.4*	*0.5*	*1*
Link4	*2*	*2100*	*31.4*	*1.1*	*2*
Link5	*2*	*2100*	*31.4*	*1.1*	*2*
Link6	*2*	*2100*	*31.4*	*1.4*	*3*
Link7	*2*	*2100*	*31.4*	*0.8*	*2*
Link8	*2*	*2100*	*31.4*	*0.4*	*1*

Table 4.2 Global network parameters

Symbol		Unit	Empirical Value
t_m	*tuning parameter for relaxation term*	*hour*	*0.0056*
c_m	*tuning parameter for anticipation term*	*km^2/h*	*35*
k_m	*tuning parameter for anticipation term*	*veh/h*	*13*
δ_m	*merging parameter*		*0.8*
$\emptyset_m$	*weaving parameter*		*2*
ρ_{jam}	*Jam density*	*veh/km/lane*	*180*
a_m	*fitting parameter for Eq. (3.5)*		*2*
$v_{free,m}$	*free speed*	*veh/h*	*100*
$\rho_{critical}$	*critical density*	*veh/km/lane*	*31.4*

The critical density in every link of the network were assumed to be 31.4 veh/km/lane, which couples with a_m= 2 to ensure that the stationary speed-density relationship (Fig. 4.5) may hold in every link due to the relaxation term in Eq. (3.3).

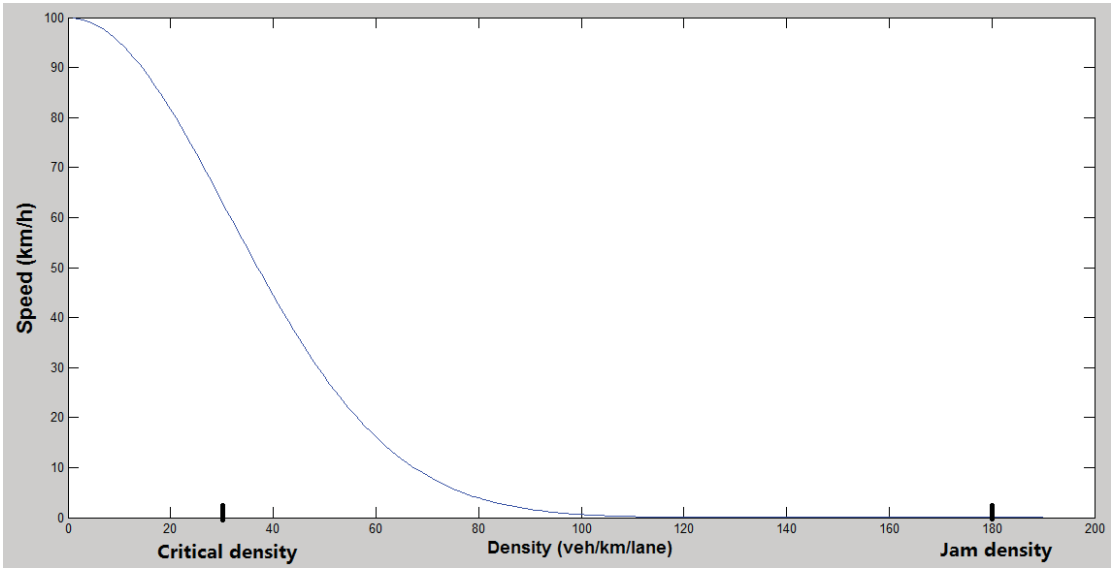


Figure 4.5 Stationary speed-density relationship for the relaxation effect

Traffic demands in Table 4.3 were adapted from the field data provided by NZTA (New Zealand Traffic Agency) collected by SCATS (Sydney Coordinated Adaptive Traffic System) to reveal the performances of the proposed algorithm under congested situations. All performances were measured by the percentage improvement in term of

TTS comparing to no-ramp-control situation.

The ramp demands in Table 4.3 were given for three two-lane on-ramps, whose capacities were set to 900veh/lane; and turning proportions at off-ramps (the percentages of exiting traffic from three off-ramps) were set to 10%, 15% and 15%. Also, it was assumed that downstream of the study location stays free-flow conditions during the simulation period.

Table 4.3 Traffic demands (veh/h)

Time	Origin	Greville (Ramp 1)	Constellation (Ramp2)	Trisram (Ramp 3)
6:00am	*2000*	*1000*	*1100*	*1000*
6:15am	*3600*	*1200*	*1200*	*1600*
6:30am	*3900*	*1300*	*1400*	*1800*
6:45am	*3600*	*1400*	*1500*	*1500*
7:00am	*3500*	*1600*	*1600*	*1400*
7:15am	*3500*	*1500*	*1700*	*1400*
7:30am	*3300*	*1500*	*1600*	*1400*
7:45am	*3100*	*1400*	*1500*	*1300*
8:00am	*2800*	*1300*	*1400*	*1300*
8:15am	*2600*	*1300*	*1200*	*1200*
8:30am	*2500*	*1100*	*1100*	*1200*
8:45am	*2500*	*1000*	*1000*	*1000*

Maximum lengths of ramp queues were set to values as shown in Table 4.4 for the penalty term of the objective function, Eq. (3.13).

Table 4.4 Constrains of ramp queues (veh/lane)

Test	Greville (Ramp 1)	Constellation (Ramp2)	Trisram (Ramp 3)
1	*90*	*165*	*50*
2	*90*	*150*	*50*
3	*75*	*150*	*50*
4	*75*	*125*	*50*

Four tests based on different constrains of queue lengths was conducted to study the influences of queue restrictions in congested traffic conditions. Weighting factors in the penalty term of Eq. (3.13), a_o, were given by $a_o = [1,100,1]$, so queue constrains of ramp 2 (Constellation) took priority over others.

4.3 Simulation Results and Discussion

The simulation results from the first phase of search and from two-phase of search were separately given to show the performance differences between the near-optimal trajectory and the optimal trajectory. ALINEA was compared to the DP based ramp metering algorithms by means of the percentage improvement of TTS based on no-ramp-control case.

ALINEA employed in this experiment was with a queue override strategy, which is given in Eq. (4.1)

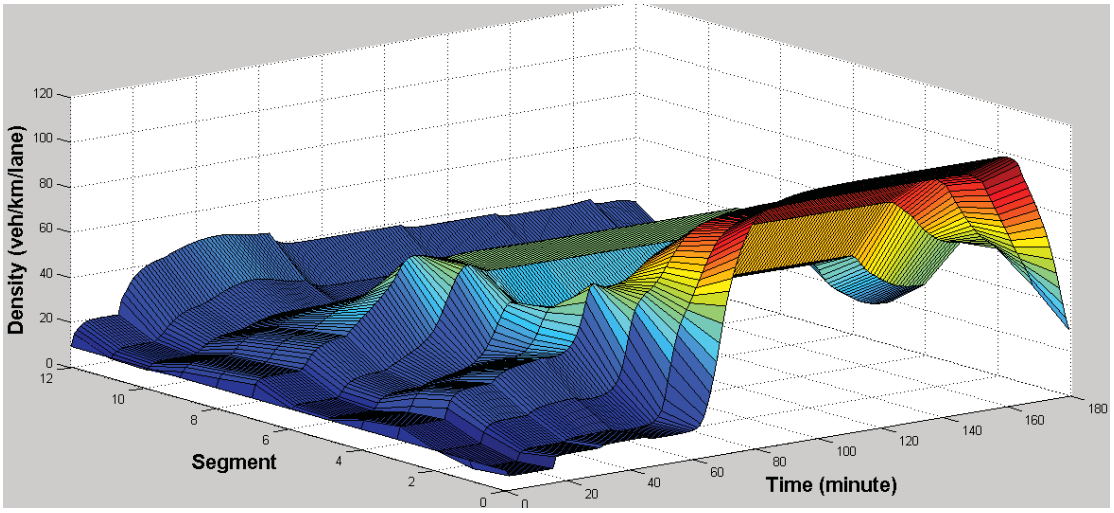
$$\begin{cases} R(l) = R(l-1) + k[\rho_{critical,m} - \rho_{m,i}(l)] & w_o \le w_{max} \\ R(l) = q_{ramp,o}(l) & w_o > w_{max} \end{cases} \quad (4.1)$$

Where, $R(l)$ is the ramp inflow at simulation step l; q_{ramp} is the traffic demand for the o-th on-ramp at simulation step l, $\rho_{critical,m}$is the critical density of link m and $\rho_{m,i}(l)$ is the density of the downstream segment i; w_o is the queue length of the o-th on-ramp and w_{max} is the queue constrain of the o-th on-ramp. k is a tunable parameter and specified as 70km/h (Hadj-Salem, Blosseville, & Papageorgiou, 1990).

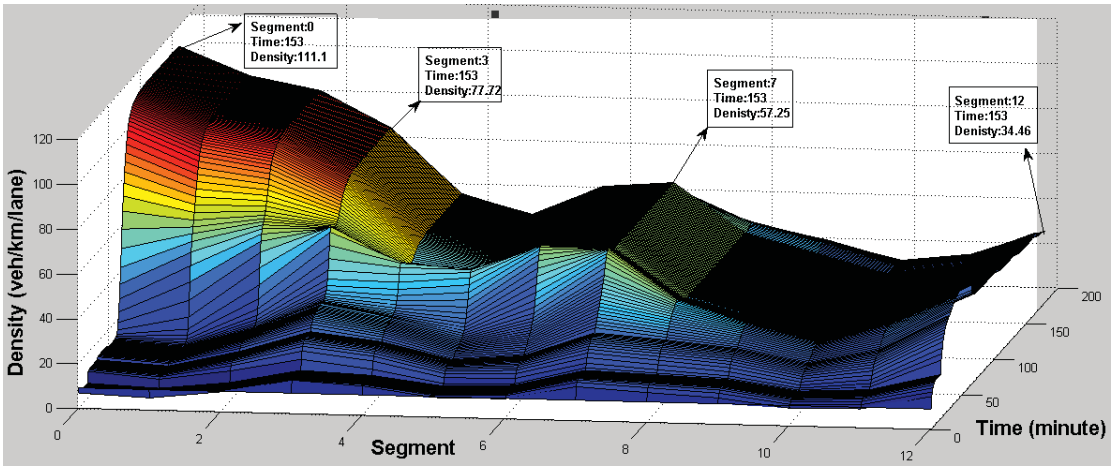
Simulation results were presented by minute-based macroscopic quantities (speeds and densities) along the motorway stretch during the simulation, and plotted in three-dimensional graphs shown as speed profiles and density profiles. Queue lengths were also plotted to observe the effects of queue constrains.

4.3.1 Simulation Results of No-ramp-control Case

The density profile and the speed profile were both plotted in 3D graphs (Fig. 4.6-4.7) to reveal motorway queues and the changes of ramp queues were plotted in 2D graphs.

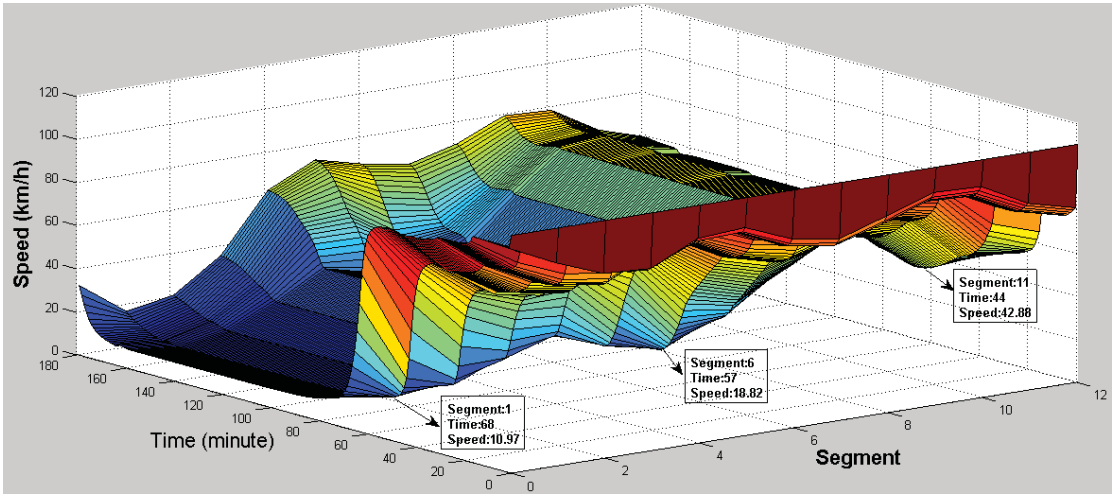


a) the density profile (front view)

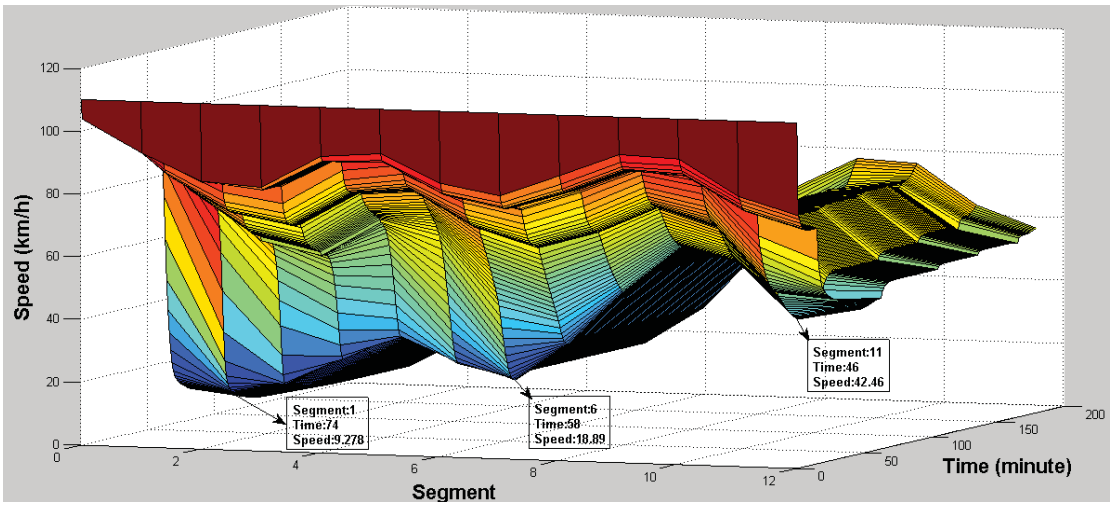


b) the density profile (side view)

Figure 4.6 The density profile under no-ramp-control case



a) the speed profile (front view)



b) the speed profile (side view)

Figure 4.7 The speed profile under no-ramp-control case

As can be seen in Fig. 4.6a, heavy traffic congestion occurred from about 7:00am to about 8:40am (from 60 to 160 in the time horizon), where motorway queues[2] emerged at ramp 2 (segment 7), being propagated to the origin (segment 0) with densities of about 110km/veh/lane (Fig. 4.6b), dissipating in front of ramp 3 (segment 12). Three bottleneck locations can be identified at the downstream of ramp 1 (segment 3), the downstream of ramp 2 (segment 7) and the downstream of ramp 3 (segment 12), by observing positions of abrupt changes of the density profile.

During the congestion, traffic flow was oversaturated from the origin to the downstream of ramp 2 (segment 7), whose densities decreased from 111veh/km/lane to 57veh/km/lane and whose speeds increased from 10km/h to 19km/h (Fig. 4.7a). From the downstream of ramp 2 (segment 7) to segment 10, speeds tended to restore within a range of 30km/h to 65km/h (Fig. 4.7b), where motorway queues dissipated with densities decreasing from 57veh/km/lane to about 15veh/km/lane (Fig. 4.6b). Traffic flow remained oversaturated before segment 10. Local congestions can be found at ramp 3 (segment 12), with densities increasing to 34veh/km/lane and speed decreasing to 42km/h due to the disturbances of ramp inflows. Disturbances from the bottleneck location can be observed from the downstream segments (Fig. 4.6b) and upstream segments (Fig. 4.7b) of all on-ramps. The changes of ramp queues are

[2] Motorway queues are normally the density profiles above the critical density between on-ramps, which can also be observed from the speed profiles below 55km/h between on-ramps (referring to section 1.1).

plotted in Fig. 4.8.

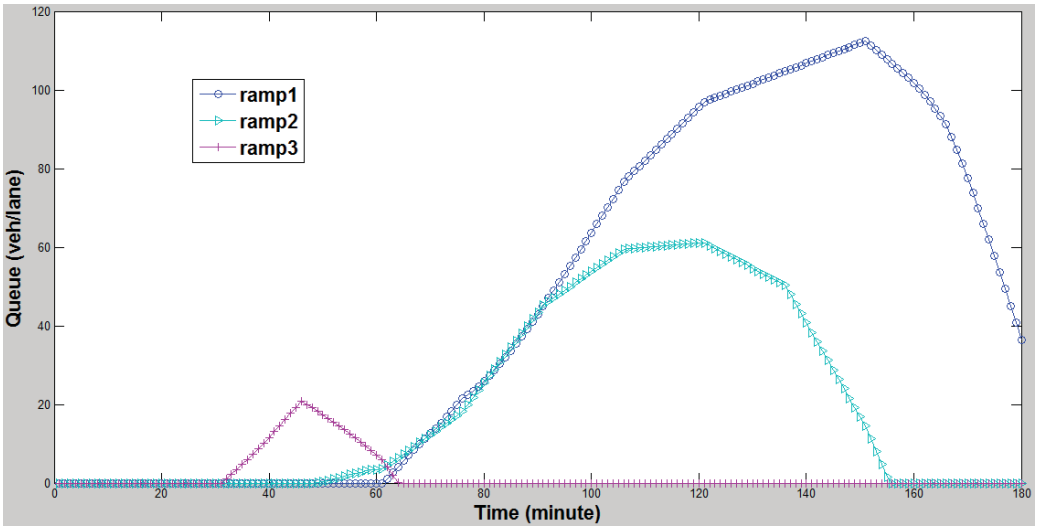
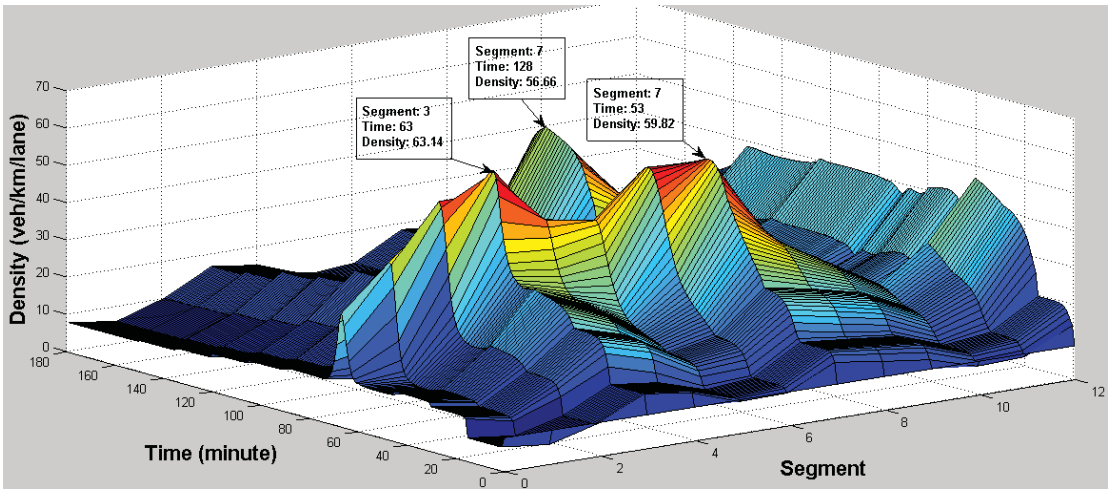


Figure 4.8 On-ramp queues under no-ramp-control case

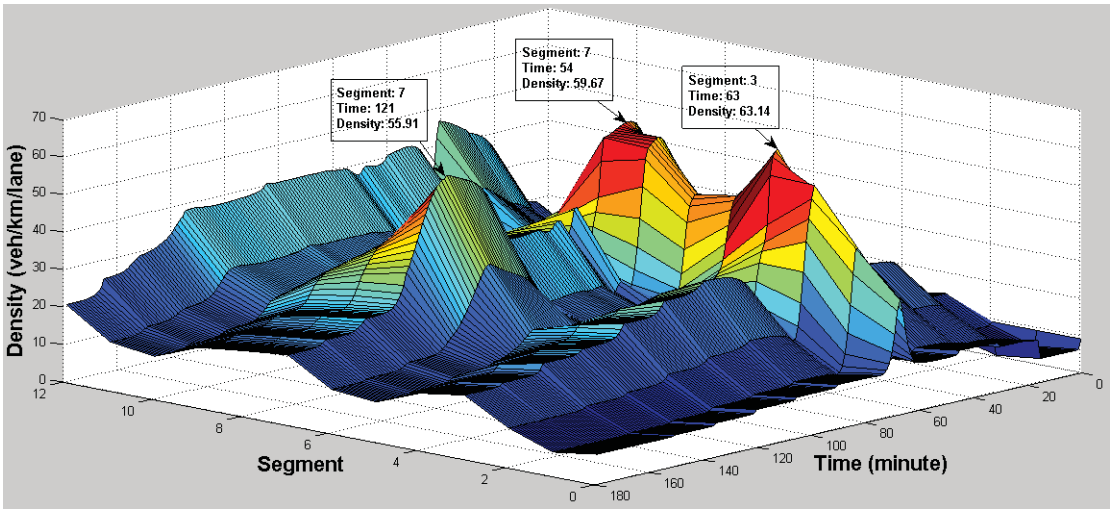
As can be seen in Fig. 4.8, the longest queue length formed at ramp 1 since ramp 1 had the highest downstream density (segment 3 in Fig 4.6), while ramp 3 had the shortest queue length because the downstream motorway stretch was under free-flow conditions. The changes of queue lengths at each on-ramp also reflect the changes of densities of its downstream segment during the simulation.

4.3.2 Simulation Results of ALINEA/Q

Four tests were conducted with different queue constrains for on-ramps (Table 4.5). The density profile and the speed profile were both plotted in 3D graphs to reveal motorway queues and ramp queues were directly plotted in 2D graphs. Simulation results from the first test were shown in Fig. 4.9~4.10, and the simulation results from the remaining tests were given in Appendix A.

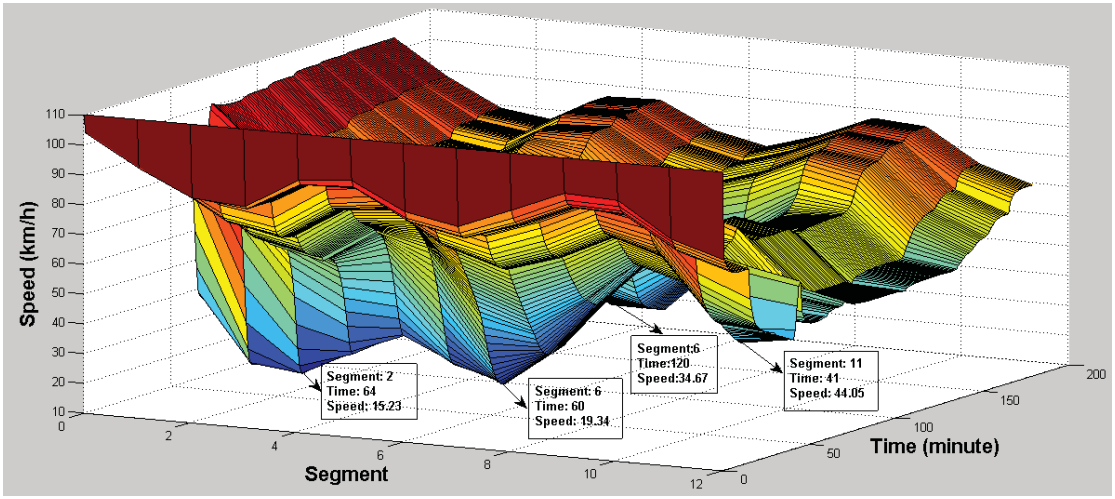


a) the density profile (front view 1)

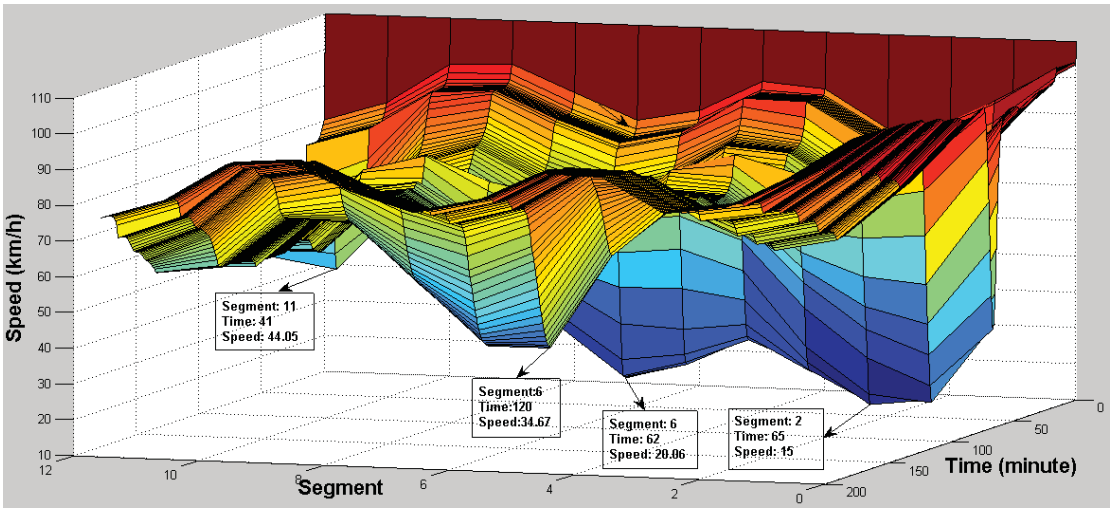


b) the density profile (front view 2)

Figure 4.9 The density profile under the control of ALINEA/Q



a) the speed profile (side view 1)



b) the speed profile (side view 2)

Figure 4.10 The speed profile under the control of ALINEA/Q

The density profile in Fig. 4.9 ranges from 10veh/km/lane to about 65veh/km/lane. The bottleneck locations can be identified at the downstream of ramp 1 (segment 3), the downstream of ramp 2 (segment 7) and the downstream of ramp 3 (segment 12). The majority of the density profile along bottleneck locations is within a range between 10veh/km/lane and 40veh/km/lane except three abrupt changes, which can be observed along the downstream of ramp 1 (segment 3) at 7:03am (time: 63) and along the downstream of ramp 2 (segment 7) at 6:54am (time:54) and at 8:01am(time: 121). It appears in Fig 4.9a that the first congestion occurred at around 7:00am (time: 60), where motorway queues emerging from ramp 2 (segment 7) with densities of about 60veh/km/lane were propagated to ramp 1 (segment 3) and peaked at densities of about 63veh/km/lane, dissipating in front of ramp 3 (segment 12). The reason of the formation of motorway queues could be explained by Table 4.3, where both traffic demands for ramp1 and ramp2 reached 1600veh/h at 7:00am while uncoordinated ramp signals failed to restrict ramp inflows at ramp 2 and caused the propagation of the motorway queues. The second congestion happened at about 8:00am (time: 120) and at ramp 2 (segment 7) alone, where no motorway queue propagated between on-ramps (Fig 4.9b) since the density profile stayed below the critical density (31.4veh/km/lane) in front of and behind this congestion.

The speed profile in Fig. 4.10 reveals the further details of two congestions. As shown

in Fig. 4.10a, the speeds, during the first congestion, dropped to 15.2km/h at the upstream segment of ramp 1 (segment 2) and restored to 40km/h before dropping to 19.3km/h (segment 6) due to the disturbances from ramp 2. Clearly, motorway queues were formed between ramp 1 and ramp2. From segment 6 to segment 10, speeds restored to around 70km/h before reaching ramp 3 (segment 12) so no motorway queue was formed between ramp 2 and ramp 3. For the second congestion (Fig. 4.10b), speeds dropped significantly at the upstream segment (segment 6) of ramp 2 (34.6km/h). The speeds stayed above 55km/h in front of and behind this drop so there was no motorway queue formed between on-ramps.

The remaining tests also show similar 3D profiles and also reveal that similar motorway queues can be found during traffic congestions (Appendix A).

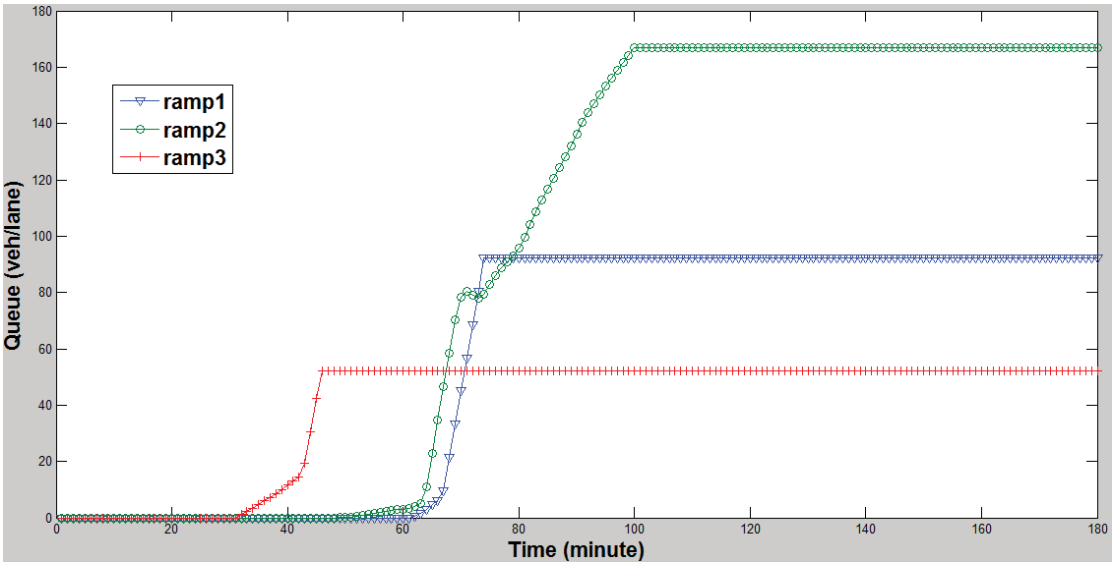


Figure 4.11On-ramp queues under the control of ALINEA/Q (Test 1)

As can be seen in Fig. 4.11, all ramp queues were constrained by the preset values in Table 4.4. The rest simulation results also indicate that ramp queues can be managed under the control of ALINEA, which can be found in Appendix A.

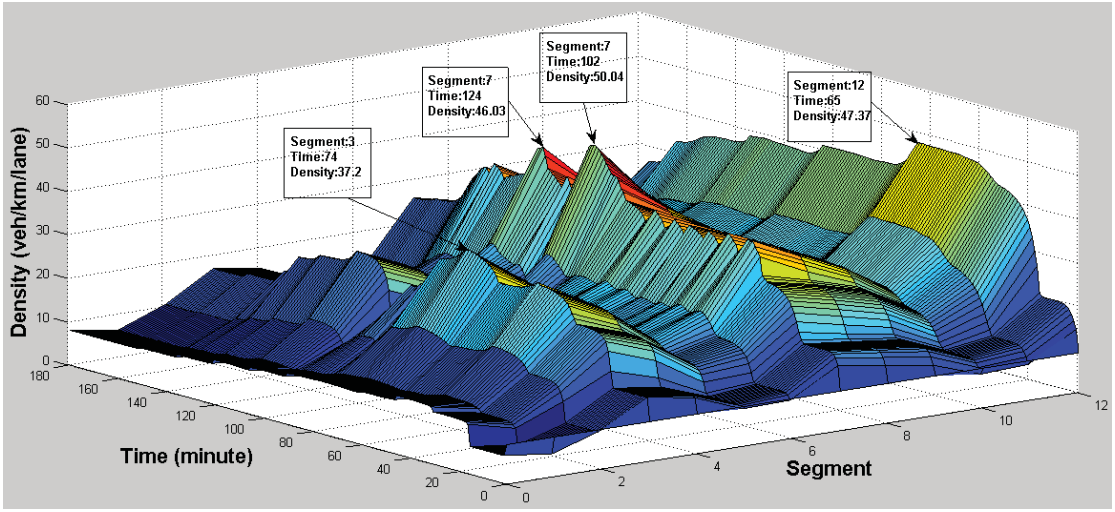
4.3.3 Simulation Results of DP based Ramp Metering

This subsection presents the simulation results of two experiments with regard to DP based ramp metering algorithms. The first experiment only implemented a simplified DP network (the first phase of search) to find near-optimal trajectories, where each

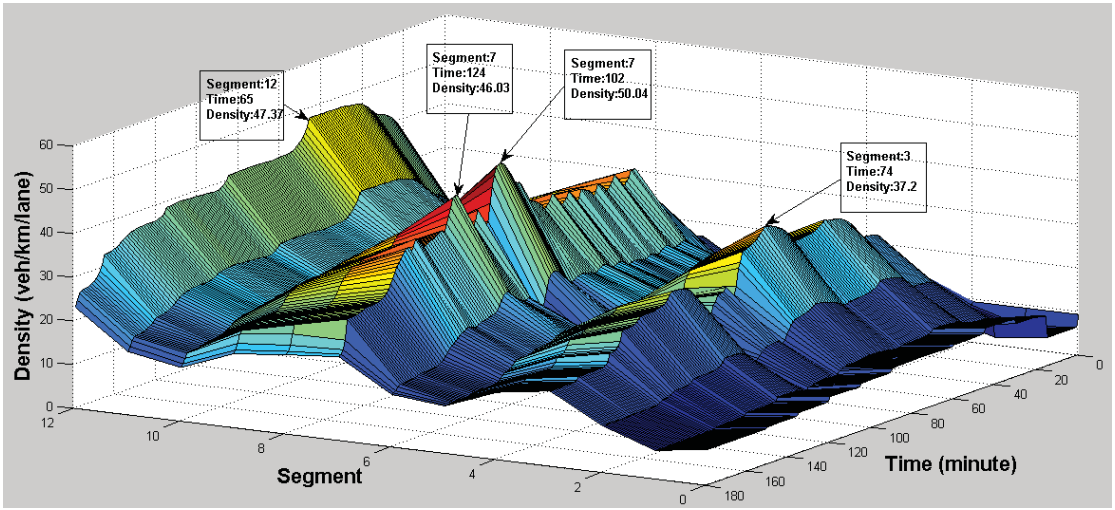
ramp meter only had 9 released metering rates. The second one applied two phases of search to identify the finer optimal trajectories, where each ramp meter had 37 released metering rates.

4.3.3.1 Simulation results from the first phase of search

Four tests had been conducted with different constrains of ramp queues (Table 4.5). The density profiles and the speed profiles were both plotted in 3D graphs to reveal traffic conditions on motorway and ramp queues were plotted in 2D graphs. Simulation results from the first test were shown in Fig. 4.12-4.13.



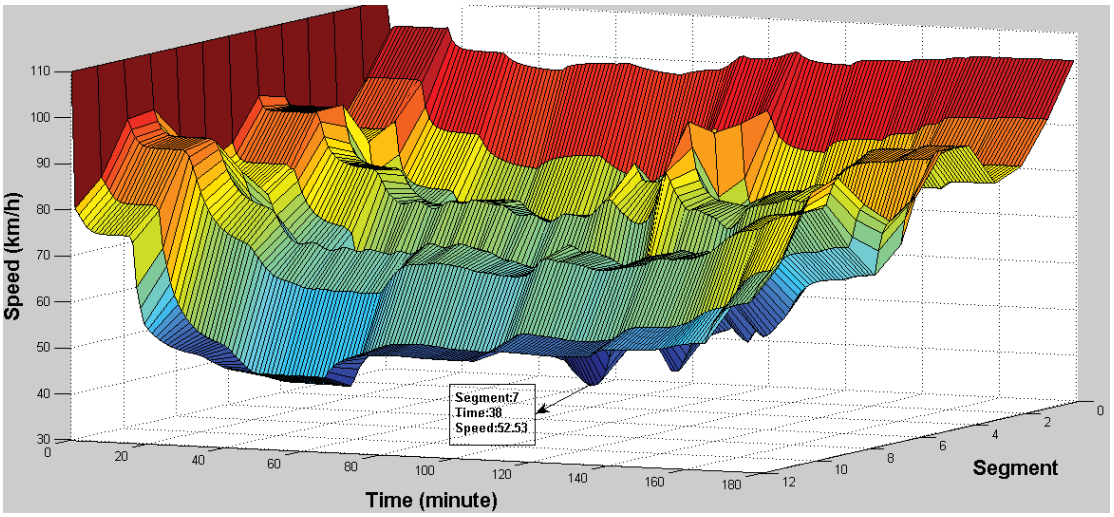
a) the density profile (front view 1)



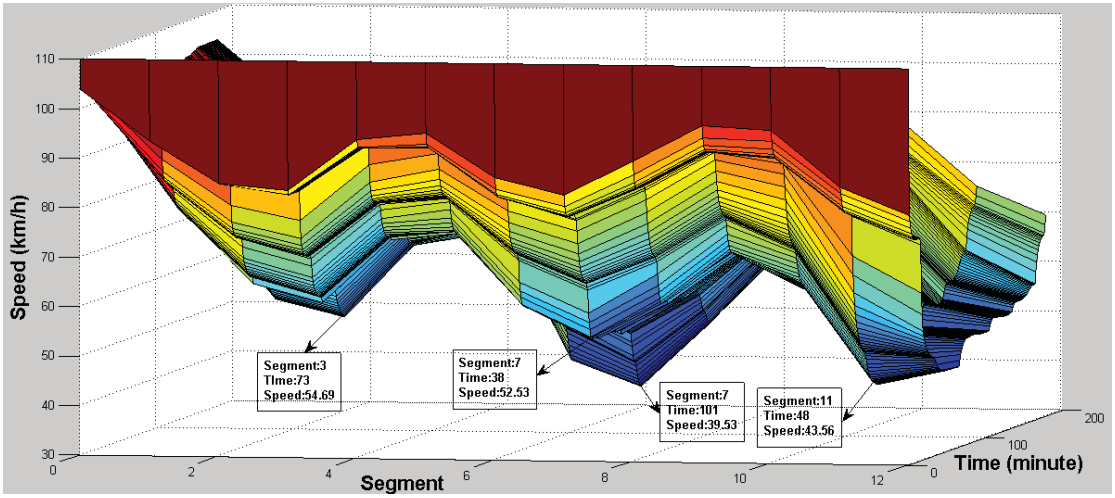
b) the density profile (front view 2)

Figure 4.12 The density profile under the control of DP (the first phase of search)

The density profile in Fig. 4.12 ranges from 10veh/km/lane to about 50veh/km/lane. The bottleneck locations can be identified at the downstream of ramp 1 (segment 3), the downstream of ramp 2 (segment 7) and the downstream of ramp 3 (segment 12). The density profiles between the bottleneck locations were below the critical density (31.4veh/km/lane), so no motorway queue was propagated between on-ramps. Four places with higher densities can be identified along bottleneck locations, where congestions may happen locally due to disturbances of ramp inflows (Fig. 4.12b).



a) the speed profile (side view 1)



b) the speed profile (side view 2)

Figure 4.13 The speed profile under the control of DP (the first phase of search)

The further details can be observed by the speed profile. In Fig. 4.13, the speed profile between on-ramps remains above 55km/h (Fig. 4.13a), so no motorway queue was formed between on-ramps during the simulation. Two local congestions can be

identified at ramp 2 (segment 7) and at the downstream segment of ramp 3 (segment 11) (Fig. 4.13b), where speeds dropped to 39.53km/h and 43.56km/h at 7.41am (time: 101) and at 6:48am (time: 48), respectively.

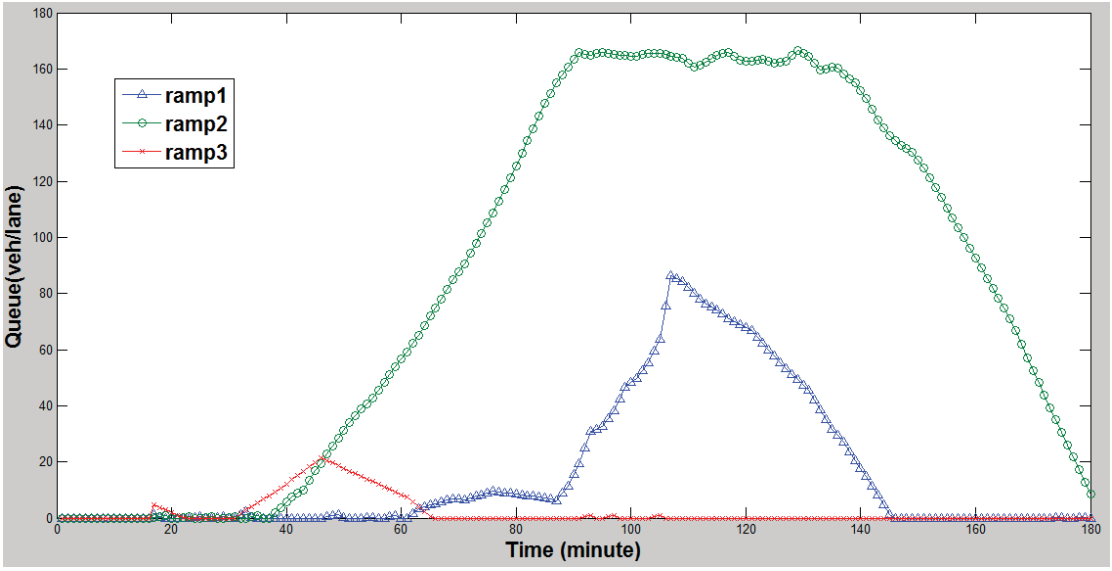


Figure 4.14 On-ramp queues under the control of DP- the first phase of search (Test 1)

Fig. 4.14 shows that all ramp queues were constrained by the preset values in Table 4.4. Unlike the queues stabilizing at the preset values under the control of ALINEA/Q (Fig. 4.11), ramp queues disappeared at the end of simulation due to the coordination of ramp signals.

The simulation results from Test 2 and Test 3 also show similar 3D profiles and no motorway queue can be found during these simulations (Appendix A). Ramp queues are also constrained within the range of the preset values. It is also noticeable that the stricter the queue constrains are getting, the higher the densities along bottleneck locations are becoming. In other words, local congestions occur more often when queue contains are getting stricter.

The simulation results from Test 4 reveal the performances of DP ramp metering under overstrict queue constrains. The density profile (Fig 4.15) shows that the motorway queues were formed between ramp 1 (segment 3) and ramp 2 (segment 7) at around 8:10am (time: 130). The speed profile further reveals the congestion, where the speed decreased from 40.75 km/h to 19.50 km/h between two on-ramps at 8:10am without the recovery in the middle segments (Fig. 4.16).

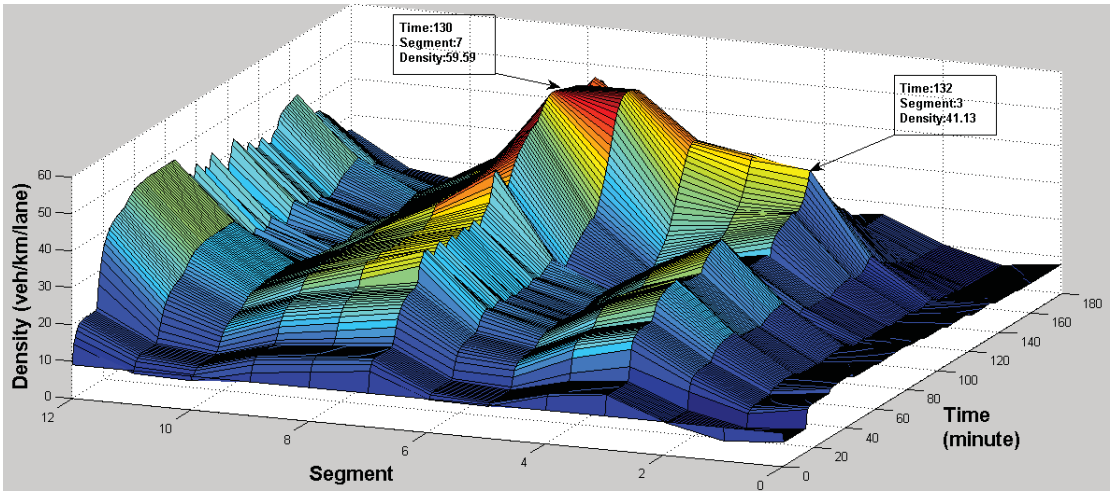


Figure 4.15 The density profile under the control of DP (the first phase of search)

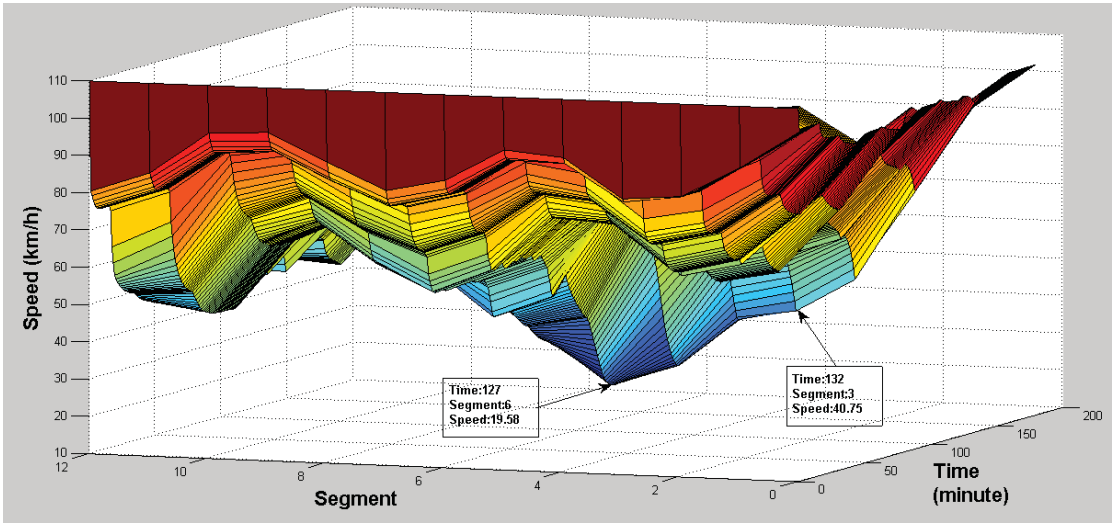


Figure 4.16 The speed profile under the control of DP (the first phase of search)

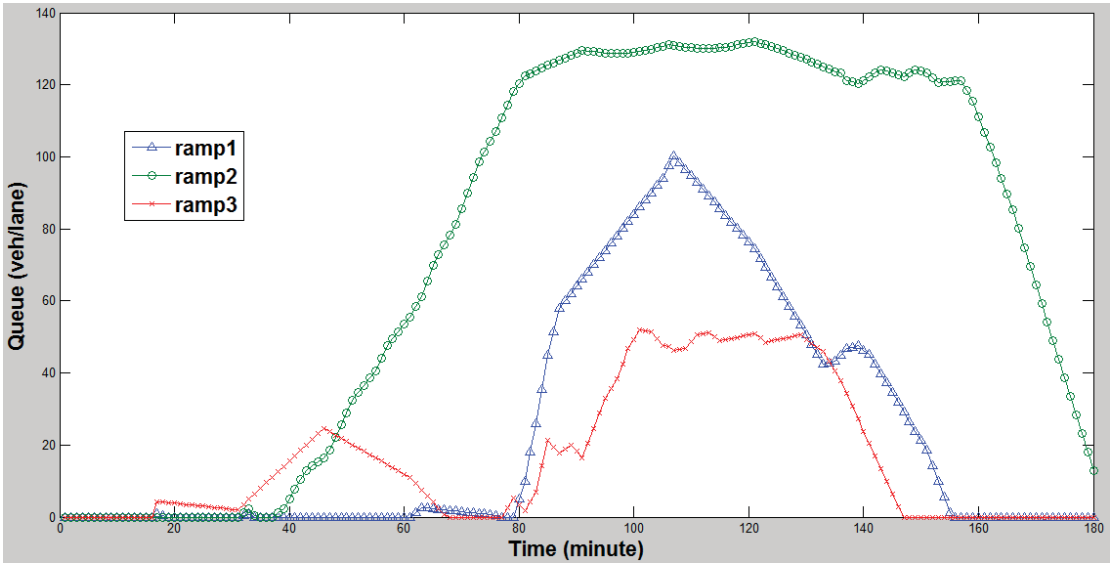


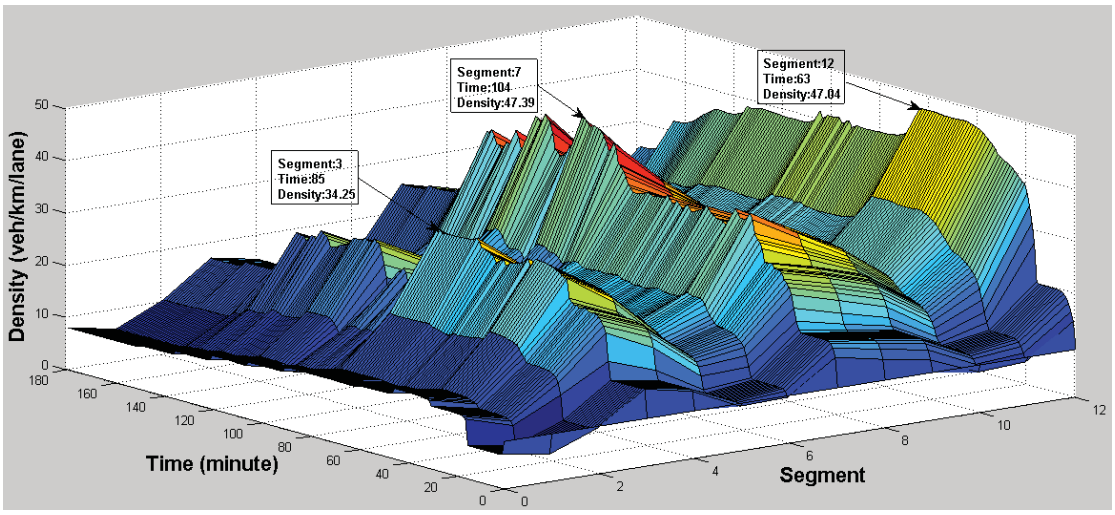
Figure 4.17 On-ramp queues under the control of DP - the first phase of search (Test 4)

The changes of ramp queues for Test 4 can be observed in Fig. 4.17, where queue

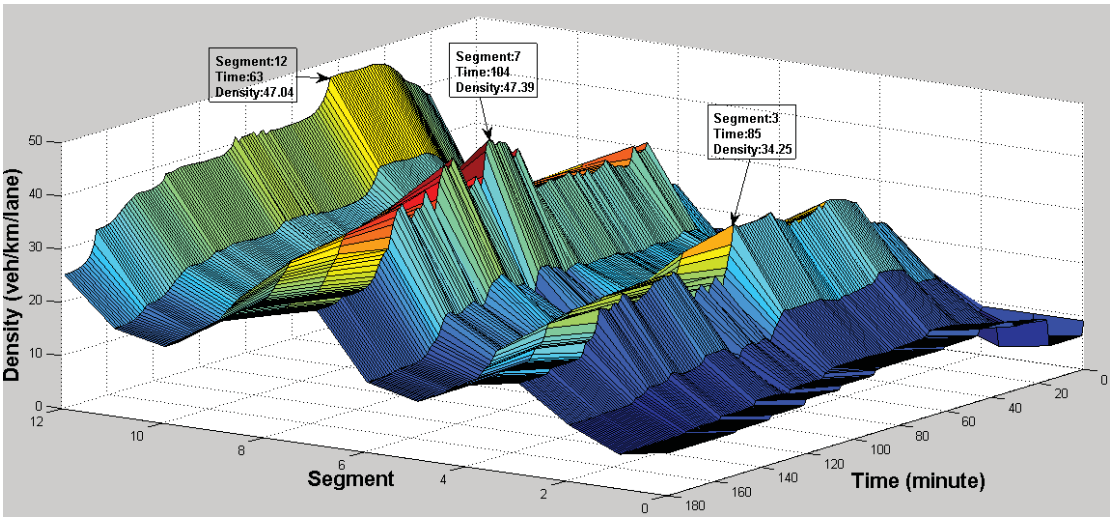
lengths at ramp 2 and ramp 3 were constrained by the preset values, but the queue length at ramp 1 failed to be constrained. The reason for that is because queue constrains of ramp 2 take priority over others ($a_o = [1,100,1]$ in Eq. (3.13)). When DP metering generated more permissive metering rates at ramp 2, it also gave stricter metering rates for the rest on-ramps to balance TTS values, which also explains why the maximum queue lengths at ramp 1 and ramp 3 in Test 4 (Fig. 4.17) are actually higher than those in Test 1 (Fig. 4.14). In addition, the permissive metering rates at ramp2 may cause serious local congestions due to the disturbances of large ramp inflows, so the motorway queues may be formed and propagated to the upstream of ramp 2, which explains why the motorway queues were formed between ramp 1 and ramp2.

4.3.3.2 Simulation results from two phases of search

Four tests had been conducted with different constrains of ramp queues (Table 4.5). The density profiles and the speed profiles were both plotted in 3D graphs to reveal traffic conditions on motorway and ramp queues were plotted in 2D graphs. Simulation results from the first test were shown in Fig. 4.12-4.13, and the simulation results from the remaining tests were given in Appendix A.



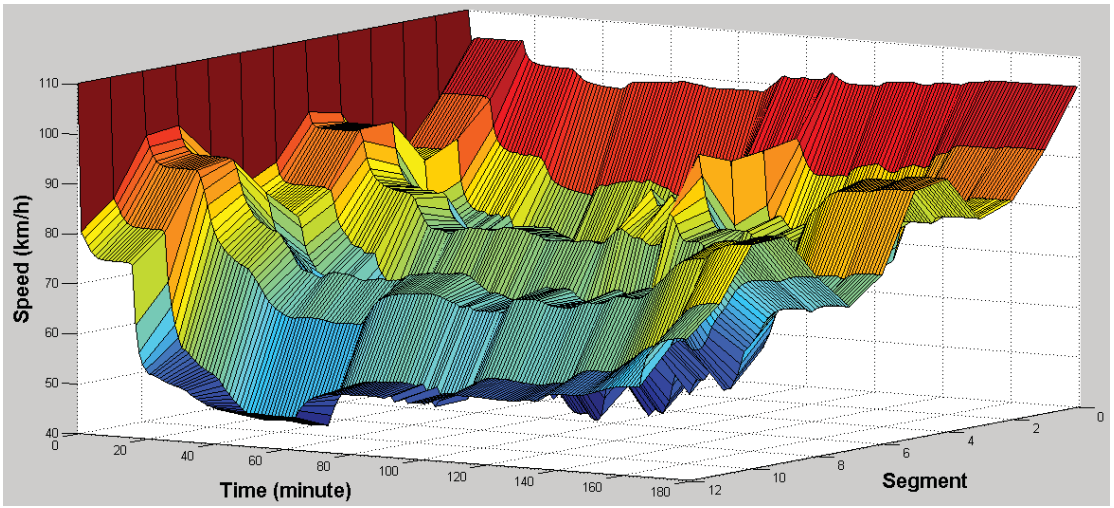
a) the density profile (front view 1)



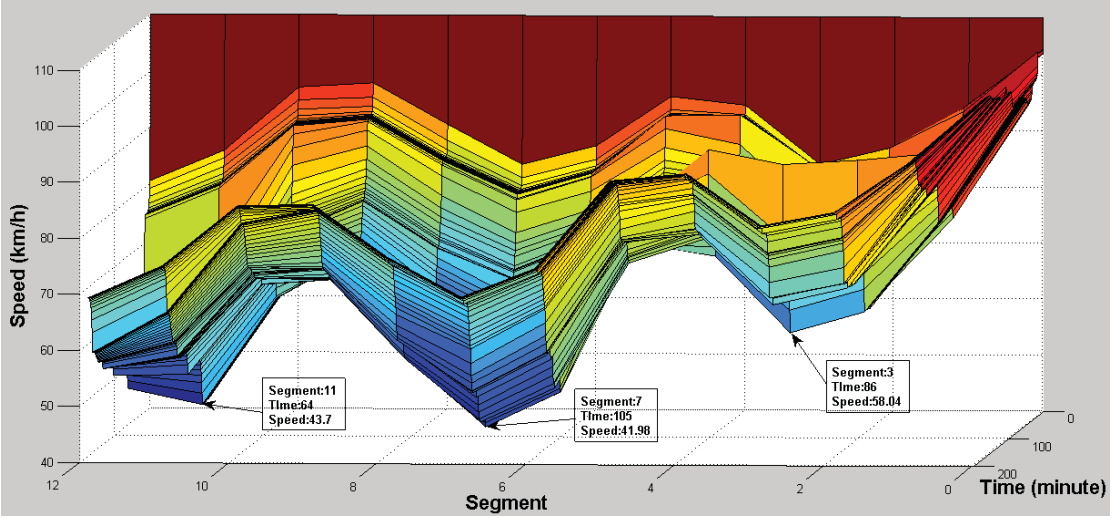
b) the density profile (front view 2)

Figure 4.18 The density profile under the control of DP (two phases of search)

The density profile in Fig. 4.18 ranges from 10veh/km/lane to about 50veh/km/lane. The bottleneck locations can be identified at the downstream of ramp 1 (segment 3), the downstream of ramp 2 (segment 7) and the downstream of ramp 3 (segment 12). The density profile between the bottleneck locations is below the critical density (31.4veh/km/lane), so no motorway queue was propagated between on-ramps. Comparing to the density profile under the control of the first phase of search (Fig. 4.12), the simulation results in Fig. 4.18 seems less fluctuant along the bottleneck locations. Three places with higher densities can be identified along bottleneck locations, where congestions may happen locally due to disturbances of ramp inflows



a) the speed profile (front view)



b) the speed profile (side view)

Figure 4.19 The density profile under the control of DP (two phases of search)

The further details can be revealed by the speed profile. In Fig. 4.19, the speed profile between on-ramps remains above 55km/h (Fig. 4.13a), so no motorway queue was formed between on-ramps during the simulation. Two local congestions can be identified at ramp 2 (segment 7) and at the downstream segment of ramp 3 (segment 11) (Fig. 4.13b), where speeds dropped to 41.98km/h and 43.70km/h at 7.45am (time: 105) and at 7:04am (time: 64), respectively. Comparing to the speed profile under the control of the first phase of search, traffic conditions in Fig. 4.19 had very little improvement.

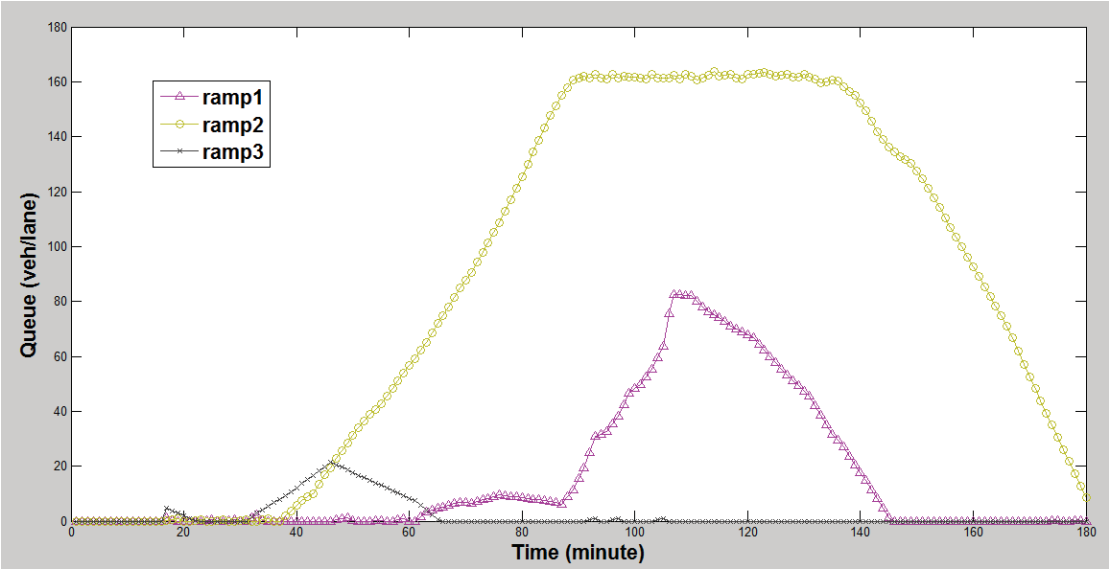


Figure 4.20 On-ramp queues under the control of DP- two phases of search (Test 1)

Fig. 4.20 shows that all ramp queues are constrained by the preset values in Table 4.4.

Comparing to Fig. 4.14, the queue lengths in Fig. 4.20 tended to stabilize at the preset values with less fluctuation due to finer control resolution from ramp metering (37 released metering rates for each ramp meter).

The simulation results from Test 2 and Test 4 also show similar 3D profiles and no motorway queue can be found from Test 2 and Test 3 (Appendix A). Comparing to the simulation results from the first phase of search, it was hardly to find any further improvement on traffic conditions from two phases of search. Ramp queues were also constrained within the range of the preset values in Test 2 and Test 3, but failed to restrict the queue lengths in Test 4 when constrains were overstrict.

4.4 TTS Comparisons

The performances of the proposed algorithm were evaluated in terms of the percentage improvements of TTS comparing to no-ramp-control case. ALINEA/Q was also compared to show the advantages of coordinated ramp metering over local responsive ramp metering.

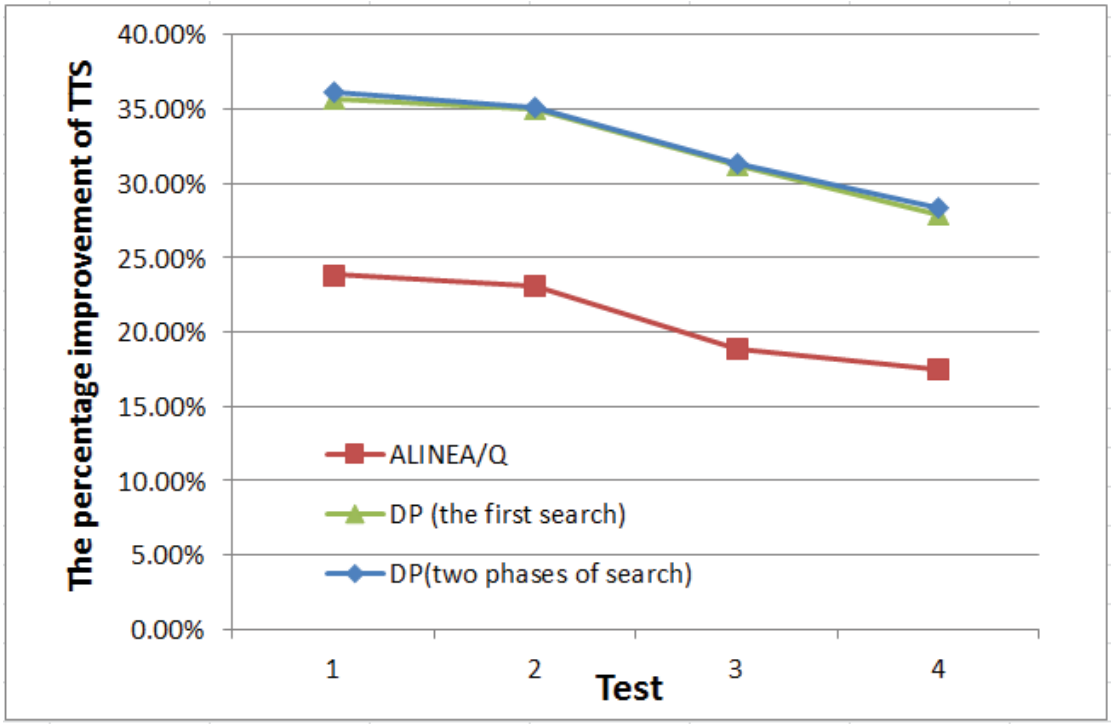


Figure 4.21 The percentage improvements of TTS in the macroscopic simulation

In Fig. 4.21, the proposed algorithm outperformed ALINEA/Q in every test, and about

10% further improvement on TTS values can be observed. It is also interesting to find out that the two phases of search DP have no obvious advantage over the first search of DP, and their performances on TTS values had very little difference, although the two phases of search DP refined the near-optimal trajectory from the first search and was able to generate 37 released metering rates for each ramp meter. This finding further prove previous work of Kotsialos et al. (2006), which concluded that the effect of the discretization can be negligible when the numbers of released metering rates are no less than ten for a local ramp meter.

Table 4.5 TTS values from the macroscopic simulation

Methods	TTS (veh*h)			
	Test 1	Test 2	Test 3	Test 4
No-ramp-control	2261.71	2261.71	2261.71	2261.71
ALINEA/Q	1722.66	1739.14	1834.74	1866.86
	(23.83%)	(23.11%)	(18.88%)	(17.46%)
DP (the first search)	1455.17	1471.07	1556.19	1630.67
	(35.66%)	(34.96%)	(31.19%)	(27.90%)
DP (two phases of search)	1444.36	1468.79	1553.38	1621.10
	(36.14%)	(35.06%)	(31.32%)	(28.32%)

4.5 Summary

In this chapter, the proposed algorithms were implemented in a deterministic environment to demonstrate their full potential, where the prediction model and the simulation model were the same and no prediction errors were considered. A simulation scenario constructed based on Auckland North Motorway was used to evaluate the performances of the proposed algorithm. Simulation results indicated that DP ramp metering was able to improve traffic condition under high traffic demands and manage ramp queue lengths under large ramp queue constrains but failed to prevent motorway queues and constrain the queue lengths when queue constrains were overstrict. Unlike the local responsive ramp metering strategy, coordinated ramp

signals were capable of isolating the local congestions from different bottleneck locations and preventing the propagation of motorway queues between on-ramps in most tests, while ALINEA/Q failed to stop the propagation of motorway queues in every test though ramp queues were constrained.

Simulation result also revealed that the second phase of search of DP made no further improvements on TTS values than the first search. The near-optimal trajectory with only 9 released metering rates was adequate to manage ramp queues and prevent the motorway congestions. This feature proved that the optimal trajectory converges very fast under the simplified DP decision networks and made it possible as an on-line control approach.

However, the macroscopic simulation in this section only indicated the performances of DP ramp metering in an ideal deterministic environment, so it is essential to evaluate the algorithm in a stochastic environment to understand the feasibility of its implementation in the real world.

CHAPTER 5 A CASE STUDY IN A MICROSCOPIC TRAFFIC SIMULATOR

This chapter presents a case study for the implementation of DP ramp metering in a microscopic traffic simulator, Aimsun (TSS, 2008). The study location in Chapter 4 was reconstructed in Aimsun as a microscopic simulation scenario, where the proposed algorithm was implemented to find the optimal trajectory along the prediction horizon. The macroscopic model employed by DP ramp metering was calibrated with the simulation results in an Aimsun scenario where the mainline traffic was under uncongested conditions. The proposed algorithm based on the calibrated model had been implemented in the Aimsun scenario to evaluate its performance under different traffic demands. The performances of DP ramp metering were also compared to ALINEA and no-ramp-control case in terms of TTS values calculated by Aimsun.

5.1 Aimsun 6 Simulation Environment

Aimsun6, as a microscopic simulator, was employed as the simulation environment to evaluate the proposed algorithm in this chapter. The microscopic simulator simulates network traffic in accordance with the collective behaviour of all vehicle-driver units within the range of network geometries. Each vehicle is individually modelled to interact with its surrounding traffic, based on the driver-behaviour models, such as lane changing models, gap-acceptance models and overtaking models. States of vehicles, states of detectors and states of traffic signals in the simulator are continuously updated over a preset time that can be split into simulation steps. Traffic devices present in a real motorway network can also be modeled in the microscopic simulator, including traffic lights, traffic detectors, Variable Message Signs, ramp meters, etc. The required inputs to Aimsun6 include the geometric information for road networks, traffic demands at origins as well as turning proportions at destinations.

Common outputs of Aimsun6 are macroscopic measures such as speed, density, flow and occupancy based on the preset statistic intervals. Due to detailed and realistic traffic features they produced, Aimsun6 was employed to test innovative algorithms for traffic management in many research areas (Barceló & Casas, 2005; Fang & Elefteriadou, 2006, 2008; Yu, Xu, Alam, Potgieter, & Fang, 2012a, 2012b).

5.2 An overview of the Simulation Study

5.2.1 Framework of the Simulation Study

An Aimsun API module had been provided to interface the user-built control algorithm with the simulator. The proposed ramp metering algorithm was programmed in Microsoft Visual C++ and interfaced to the Aimsun simulator by means of the API module. The framework of simulation study is shown as the following graph.

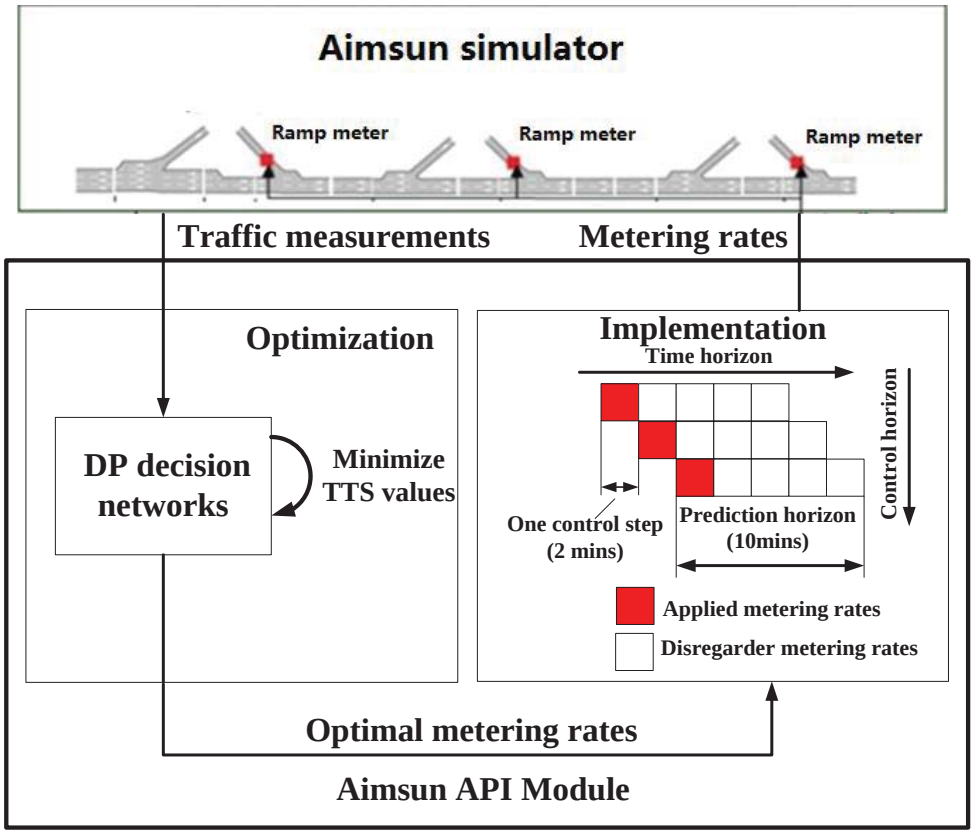


Figure 5.1 The framework of the implementation in Aimsun6

The traffic measurements in the Aimsun scenario were collected at every control step

of 2mins, based on which a macroscopic model (Section 3.1) was used to predict traffic states over a future time horizon of 10mins and evaluate the performance of the control strategy by TTS values. The DP method was applied to minimize the cost and achieve the optimized ramp metering rates along the prediction time horizon. Only the first control step of optimized ramp metering rates was actually applied to the motorway network in Aimsun. The rest ramp metering rates were disregarded. Afterwards, the prediction horizon shifted one control step forward, and the whole optimization process started over again. This optimization procedure was also known as receding horizon control or model predictive control

5.2.2 Procedures of the Implementation of the Proposed Algorithm

The Aimsun API module was provided to collect the required data (e.g. flow, occupancy, etc.) from the detectors in the simulation scenario. Based on the collected information, the proposed algorithms (or external applications) are able to send the corresponding actuations (change the traffic signal state) to emulate the operations of traffic devices (ramp metering signs).

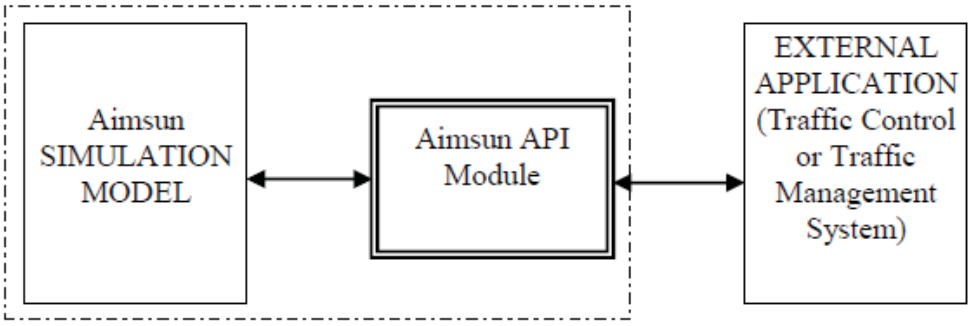


Figure 5.2 The conceptual structure of Aimsun API application

This communication process (Fig.5.2) is guaranteed by eight functions defined in Aimsun API module: AAPILoad, AAPIInit, AAPIManage, AAPIPostManage, AAPIFinish, AAPIUnLoad, AAPIEnterVehicle and AAPIExitVehicle (TSS, 2010).

- AAPILoad (): It is called when the module is loaded by Aimsun.
- AAPIInit (): It is called when Aimsun starts the simulation.

- AAPIManage (): This is called in every simulation step at the beginning of the cycle, and can be used to request detector measures, vehicle information and interact with junctions, metering and VMS in order to implement the control and management policy.
- AAPIPostManage (): This is called in every simulation step at the end of the cycle, and can be used to request detector measures, vehicle information and interact with junctions, metering and VMS in order to implement the control and management policy.
- AAPIFinish (): It is called when Aimsun finish the simulation.
- AAPIUnLoad (): It is called when the module is unloaded by Aimsun.

Since Aimsun6 is based on stochastic models, five replications were run for one simulation experiment in this Chapter, and simulation results from all replications were averaged as outputs of the experiment. The simulation step of the microscopic simulator was set to 0.5s. Every time at the end of one simulation step, the API module interacted with the simulator by means of AAPIPostManage () function, where, if one detection interval[3] (2mins) was completed, then the proposed ramp metering algorithm utilized the current traffic measurements to compute the optimal metering rates for next control step (2mins). The API module was coded in C++ and integrated to the simulator by Dynamic Link Library generated in Microsoft Visual Studio 2005 (Appendix C).

Fig. 5.3 shows the scheme of how the proposed algorithms interacted with Aimsun by means of the Aimsun API module.

[3] In Aimsun, detection intervals are time intervals, based on which statistical outputs of detectors can be calculated. Control intervals (or control steps) in this chapter were given to the same values (2mins) as detection intervals.

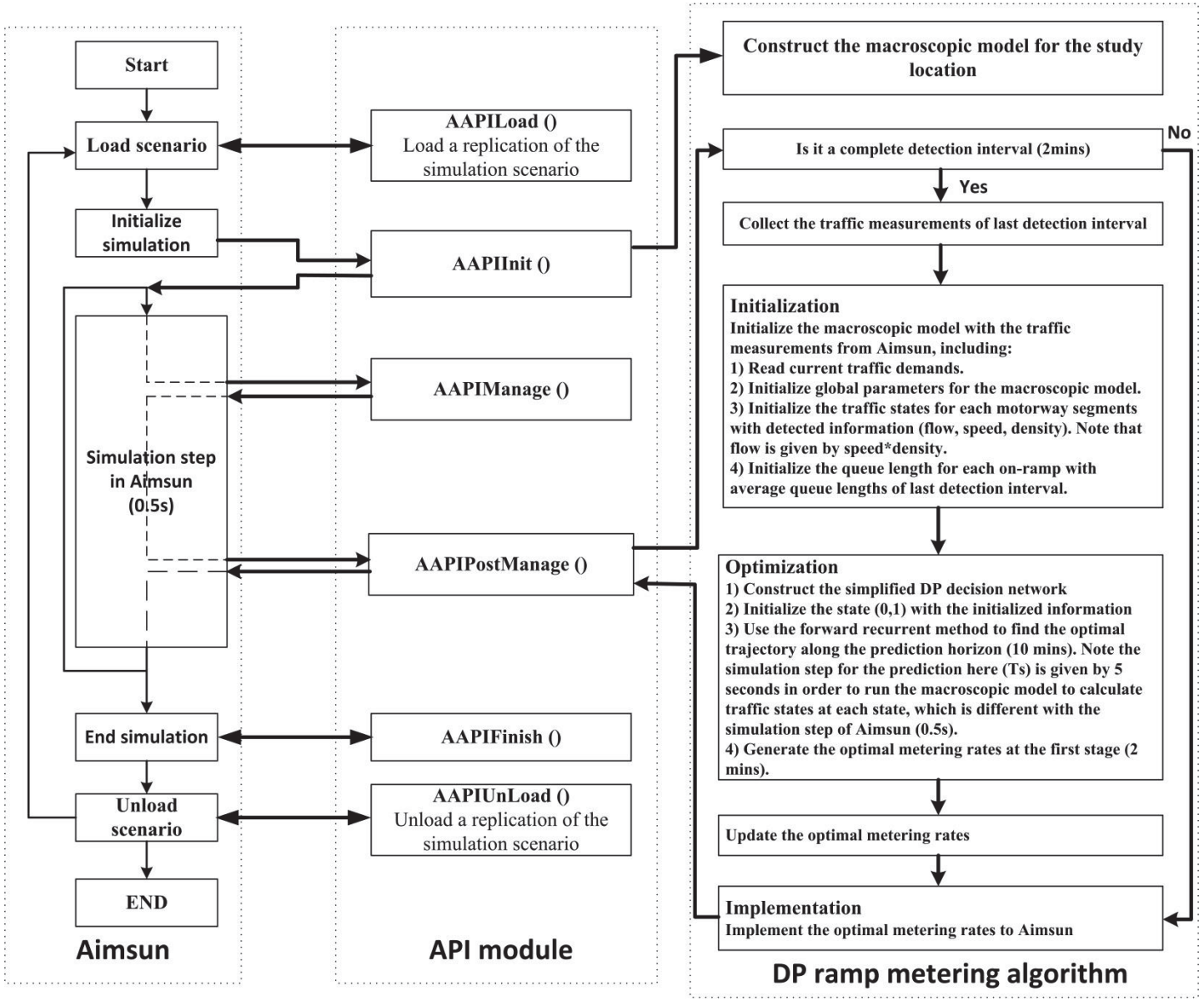


Figure 5.3 The scheme of the implementation of the proposed Algorithm

5.3 Study Area in Aimsun6

The simulation scenario was constructed based on the same study location in Chapter 4, which is a 6.7km motorway stretch from the upstream southbound off-ramp of Grenville interchange to the downstream southbound on-ramp of Tristram interchange in the Auckland Northern Motorway (Fig. 5.4). The study area includes three interchange, Greville interchange (1), Constellation interchange (2) and Trisram interchange (3). There are totally three two-lane on-ramps and three two-lane off-ramps in the motorway stretch.

5.3.1 Road Information and Vehicle Information

Basic road information in the simulation scenario is given by Table 5.1.

Table 5.1 Basic road information

Section Type	Speed Limit (km/hour)	Lane Width (meters)	Capacity (veh/hour/lane)
Freeway	100	3	2100
On ramp	90	3	900

No bus was included in the simulations because bus lanes are independent from on-ramps in study area. No HOVs (high occupancy vehicles) are allocated in the simulations since HOVs have uncontrolled priority lanes (e.g. T2 lane for two or more people per car) in study area. 'Car' is the only vehicle type employed in the simulations. Parameters of the vehicle type were defined by the default values of the simulator (Table 5.2).

Table 5.2 Vehicle information

Parameters	Mean	Min	Max	Deviation
Length	4meters	3.4meters	4.6meters	0.5meters
Width	2meters	2meters	2meters	0
Max desired speed	110km/h	80km/h	150km/h	10km/h
Max acceleration	3m/s^2	2.6 m/s^2	3.4 m/s^2	0.2 m/s^2

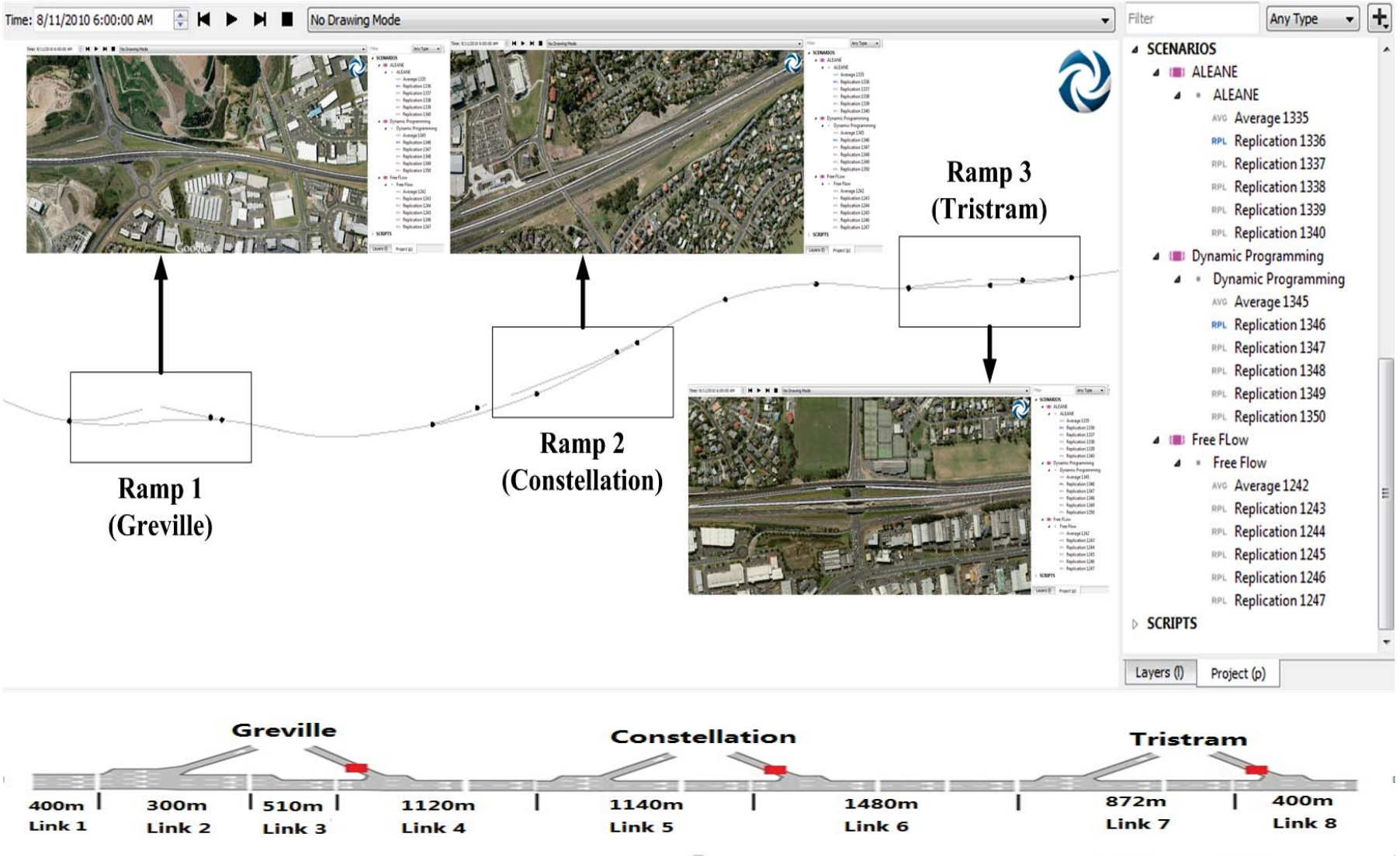


Figure 5.4 The study location in Aimsun6

5.3.2 The Distribution of Detectors

Traffic measurements, including speed (km/hour) and count (veh/lane), were collected at every detection interval (2mins) from the detectors installed in the Aimsun scenario. The detectors along the mainline motorway were distributed, according to the segment lengths in Table 4.1, to divide the motorway into 13 road segments as the following the graph.

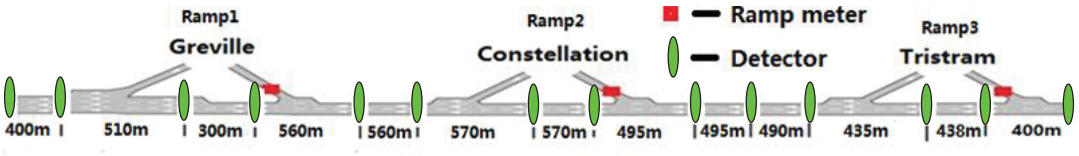


Figure 5.5 The distribution of mainline detectors

Each on-ramp was allocated with two detectors installed in front of the ramp meter and at the end of the on-ramp respectively. The front one to vehicles accessing the motorway is named as the check-in detector and the later one to vehicle approaching on-ramps is named as the queue detector.

Fig. 5.6~Fig. 5.8 shows the distribution of detectors along on-ramps.

Figure 5.6 The distribution of detectors at ramp1

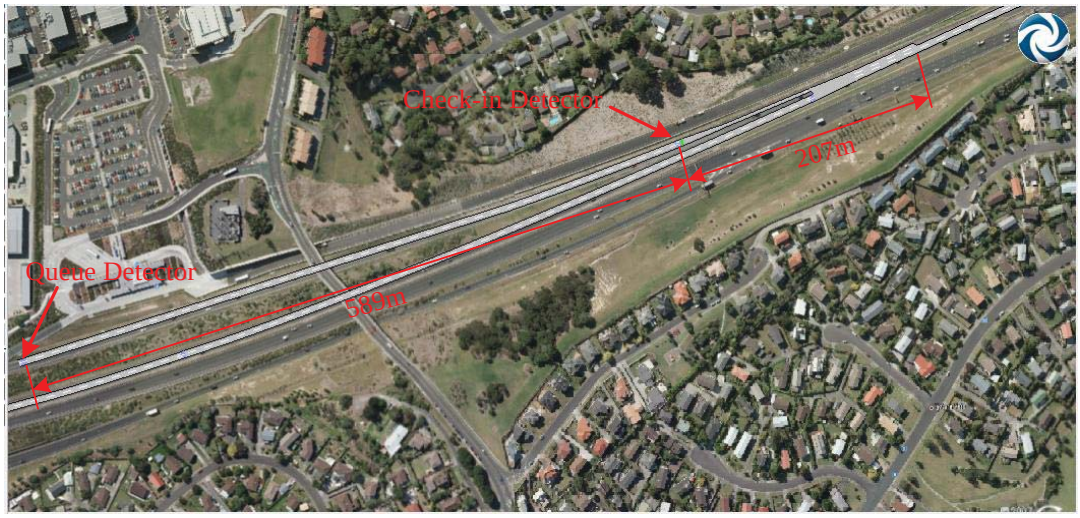


Figure 5.7 The distribution of detectors at ramp2

Figure 5.8 The distribution of detectors at ramp3

Two macroscopic measurements, speed (km/h) and density (veh/km/lane), were calculated by the detected measurements at each detection interval for each segment and delivered to the API module as the initial traffic conditions of the DP decision network. The speed is the average value of speeds from two adjacent detectors. The density is calculated by Eq. (5.1).

$$\rho(k) = \frac{c_{in}(k) - c_{out}(k)}{L_{m,i}} \tag{5.1}$$

Here, $\rho(k)$ is the density of i-th segment of link m at k-th detection interval. $c_{in}(k)$ is the traffic count of the upstream detector of the segment i at k-th detection interval, and $c_{out}(k)$ is the traffic count of the downstream detector of the segment i. $L_{m,i}$ is the length of the segment i.

5.4 Model Calibration under Uncongested Traffic Conditions

In order to estimate the traffic states of the Aimsun scenario, the study location was also constructed by the macroscopic model (referring to Section 4.2) in the Aimsun API module. The motorway network was divided into 8 links subdivided into 13 segments as shown in Fig. 5.4, and the geometric details of the network links of the study location are given in Table 5.3.

In this section, network parameters of the macroscopic model have to be identified by the simulation results from Aimsun to ensure the model reflects traffic dynamics from the Aimsun scenario. The calibrated model is able to demonstrate the traffic behavior of the study location in Aimsun when mainline traffic was under uncongested conditions during the simulation[4].

Table 5.3 The geometric details of the network links in the macroscopic model

Name	Lane number	Capacity (Vehicles/lane)	Length (km)	Segment number
Link1	*3*	*2100*	*0.4*	*1*
Link2	*3*	*2100*	*0.3*	*1*
Link3	*2*	*2100*	*0.5*	*1*
Link4	*2*	*2100*	*1.1*	*2*
Link5	*2*	*2100*	*1.1*	*2*
Link6	*2*	*2100*	*1.4*	*3*
Link7	*2*	*2100*	*0.8*	*2*
Link8	*2*	*2100*	*0.4*	*1*

5.4.1 Parameter Identification

The parameters needed to be identified are given in Table 5.4. The first four parameters (ρ_{jam}, $v_{free,m}$, $\rho_{critical}$ and a_m) in the Table are global parameters that

[4] It is difficult to calibrate one set of parameters for a macroscopic model to meet both free-flow and congested flow conditions, so the parameter identification in this chapter is only for uncongested flow conditions since the mainline traffic should remain uncongested as long as ramp metering does not fail.

were identified at the downstream segments of on-ramps (bottleneck locations) by the speed-density relationship. δ_m was identified for each on-ramp by the comparisons of the ramp queue lengths. The last three parameters, t_m, c_m and k_m, were identified for each link by the mainline traffic conditions, and the parameter identification procedure followed the previous work (Messme & Papageorgiou, 1981; J.-S. Yang, 2002).

Table 5.4 Network parameters to be identified

Symbol		Unit
$\boldsymbol{\rho_{jam}}$	*Jam density*	*veh/km/lane*
$\boldsymbol{a_m}$	*fitting parameter for Eq. (3.5)*	
$\boldsymbol{v_{free,m}}$	*free speed*	*veh/h*
$\boldsymbol{\rho_{critical}}$	*critical density*	*veh/km/lane*
$\boldsymbol{\delta_m}$	*merging parameter*	
$\boldsymbol{t_m}$	*tuning parameter for relaxation term*	*hour*
$\boldsymbol{c_m}$	*tuning parameter for anticipation term*	*km^2/h*
$\boldsymbol{k_m}$	*tuning parameter for anticipation term*	*veh/h*

5.4.1.1 Identification of ρ_{jam}, $v_{free,m}$, $\rho_{critical}$ and a_m

The global parameters, $v_{free,m}$, $\rho_{critical}$ and a_m, were employed by Eq. (3.5) to calculate the equilibrium speeds of each link, at which the relaxation term of Eq. (3.3) enables the perturbed speeds restored. In order to identify those parameters, the speed-density curve of Eq. (3.5) was adjusted to fit to the collected data from the bottleneck locations in the Aimsun scenario by trial and error.

A traffic simulation was run in Aimsun under the traffic demands of Table 5.5 to collect the speed-density data at bottleneck locations (downstream segments of on-ramps) in the simulator. The turning proportions at off-ramps (the percentages of exiting traffic from three off-ramps) were set to 10%, 15% and 15%.

Table 5.5 The traffic demands for parameter identification (ρ_{jam}, $v_{free,m}$, $\rho_{critical}$, a_m)

Time	Origin	Greville (Ramp 1)	Constellation (Ramp2)	Trisram (Ramp 3)
6:00am	*2000*	*400*	*400*	*400*
6:15am	*2200*	*800*	*800*	*800*
6:30am	*2900*	*1000*	*1000*	*1000*
6:45am	*3300*	*1200*	*1200*	*1200*
7:00am	*3500*	*1600*	*1600*	*1600*
7:15am	*6000*	*1800*	*1800*	*1800*
7:30am	*6000*	*1800*	*1800*	*1800*
7:45am	*3500*	*1600*	*1600*	*1600*
8:00am	*3300*	*1200*	*1200*	*1200*
8:15am	*2900*	*1000*	*1000*	*1000*
8:30am	*2200*	*800*	*800*	*800*
8:45am	*2000*	*400*	*400*	*400*

The identified speed-density curve was given by Eq. (5.2).

$$V[\rho_{m,i}(l)] = v_{free,m} exp\left(-\frac{1}{a_m}\left(\frac{\rho_{m,i}(l)}{\rho_{crit,m}}\right)^{a_m}\right)$$

$$= 110 exp\left(-\frac{1}{1.7}\left(\frac{\rho_{m,i}(l)}{33}\right)^{1.7}\right) \quad (5.2)$$

Here, $v_{free,m}$= 110km/h, $\rho_{critical}$=33veh/km/l and a_m=1.7.

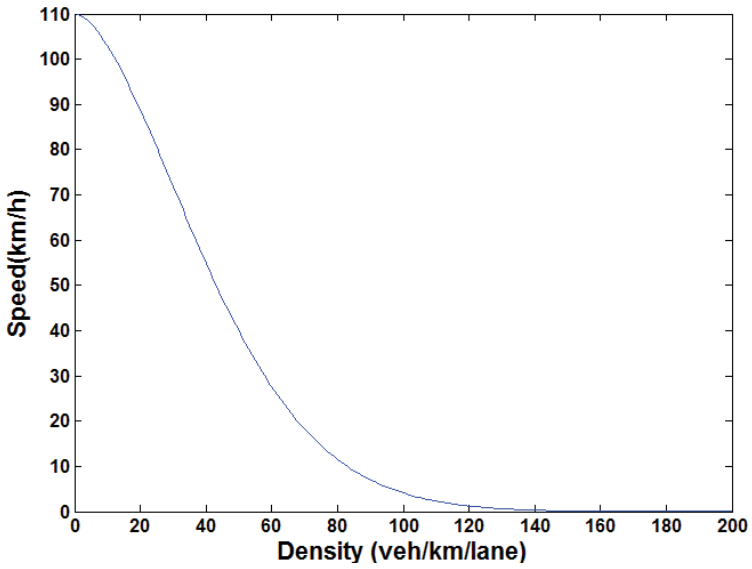


Figure 5.9 The identified speed-density relationship

The collected data from Aimsun can be seen in Fig. 5.10~Fig. 5.12.

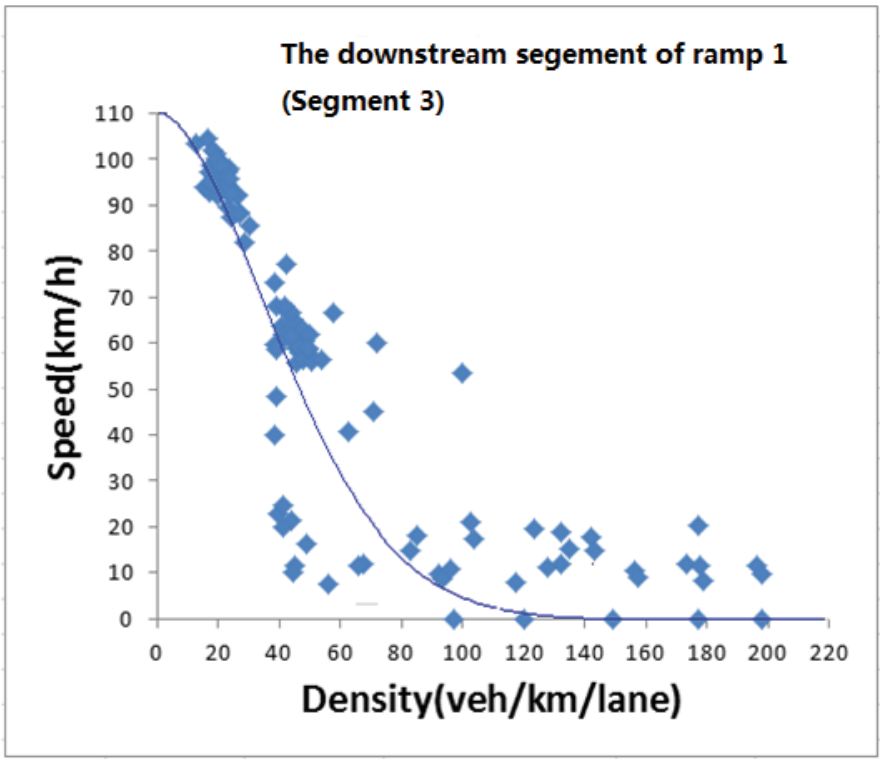


Figure 5.10 The speed-density relationship at segment 3

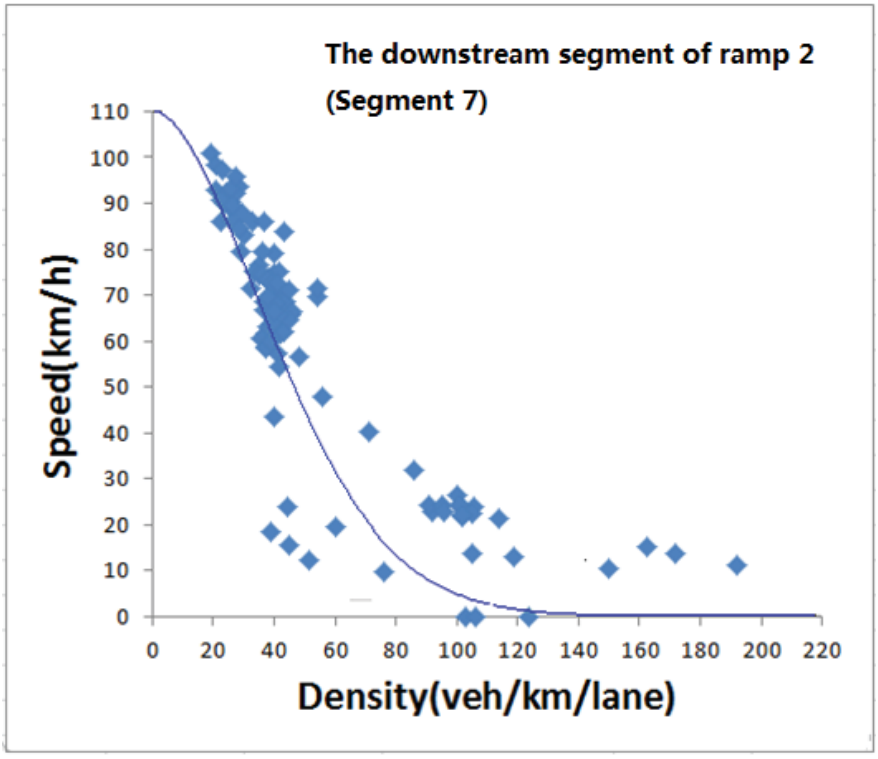


Figure 5.11 The speed-density relationship at segment 7

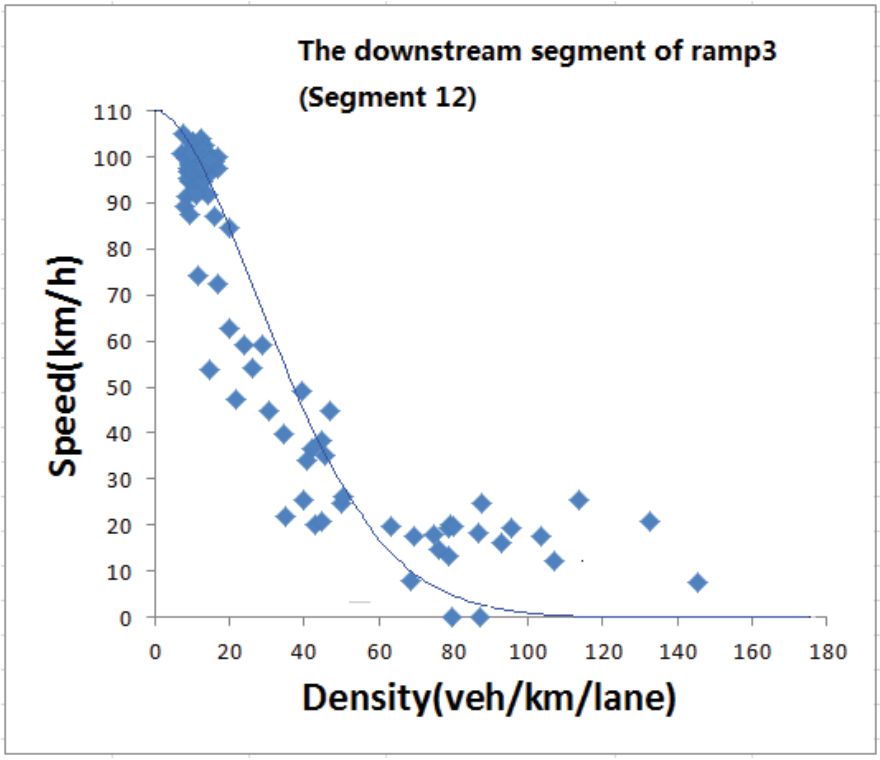


Figure 5.12 The speed-density relationship at segment 12

As can be seen in Fig. 5.10~Fig. 5.12, the identified speed-density curve has the best fit to collected density data ranging from 10veh/km/lane to 60veh/km/lane, but it fails to fit congested flow conditions where speed-density data are unclustered. In addition, ρ_{jam} can be set to 180veh/km/lane according to Fig. 5.9.[5]

5.4.1.2 Identification of t_m, c_m and k_m

t_m, c_m, and k_m were employed in Eq. (3.3) to tune the dynamic speed-density relationship to show road characteristics of each link, so they were identified by mainline traffic conditions for each link individually. Since every segment in the link share the same road characteristics, t_m, c_m and k_m should remain the same in every segment of the same link.

Eight experiments had been conducted to identify the parameters of eight links individually. Each experiment normally collected traffic data from the motorway section consisting of three roadway segments. The collected data from both ends of

[5] In Aimsun, the default value of jam density is 200veh/km/lane, but traffic speeds may not reach zero even at the jam density. The suggested value (180veh/km/lane) here can be any value above 140veh/km/lane (Fig. 5.9), since it is only involved in the calculation of queue lengths and influences of this value are minor in the simulations.

the section were treated as system inputs, while the collected data from the middle segment are taken as system outputs.

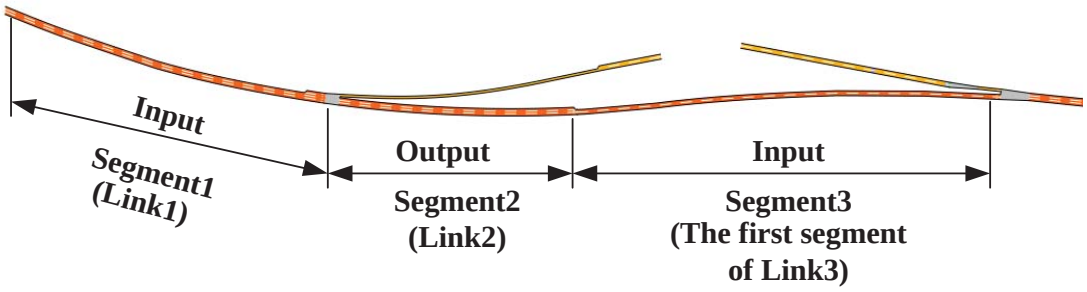


Figure 5.13 The motorway section to identify the parameters for Link2

Take Fig. 5.13 as an example. The graph shows the motorway section to identify the parameters of Link 2, where speeds and densities at segment 1 and segment 3 were recorded at every detection interval as inputs while speeds and densities at segment 2 were recorded as outputs. With the inputs from segment 1 and segment 3, the speeds and densities of segment 2 can be estimated by Eq. (3.2) and Eq. (3.3) at every detection interval. Then the parameter identification problem for segment 2 can be formulated as the following least squares error problem:

$$Min\ I(\theta) = Min \sum_{k=1}^{K} [(estimated\ data\ at\ segment2) - (measured\ data\ at\ segment2)]^2 \tag{5.3}$$

Here, θ is the parameter set, [t_m, c_m, k_m]; k is the index number of the detection interval; K is the simulation run length. Considering that traffic densities, which are essential measurements to calculate TTS values (Eq. (3.13)) and queue lengths (Eq. (3.09) and Eq. (3.10)), are relatively more important than traffic speeds. The Eq. (5.3) was rewritten and specified as Eq. (5.4), which was employed as the objective function for all experiments.

$$Min\ I(\theta) = Min \sum_{k=1}^{K} [(calculated\ densities\ at\ segment\ i) - (measured\ densities\ at\ segment\ i)]^2 \quad (5.4)$$

Fig. 5.13 only indicates the layout of the motorway section to identify the parameters for a link consisting of one segment. For links consisting of three segments, the data from the middle segment were collected as outputs while the data from both ends of the link were recorded as inputs. For links consisting of two segments, the data from the first segment were collected as outputs, while the data from the second segment and from the last segment of last link were recorded as inputs. For both ends of the motorway stretch (Link 1 or Link 8), only one downstream or upstream segment was used to collect inputs, and upstream or downstream densities and speeds were presumed to remain the same as outputs.

Table 5.6 The traffic demands for parameter identification (δ_m , t_m, c_m and k_m)

Time	Origin	Greville Ramp(1)	Constellation Ramp(2)	Trisram Ramp(3)
6:00am	*2800*	*1000,*	*1000,*	*1000*
6:15am	*3100*	*1300,*	*1200,*	*1200*
6:30am	*3500*	*1400,*	*1200,*	*1300*
6:45am	*3600*	*1600,*	*1300,*	*1400*
7:00am	*3500*	*1600,*	*1300,*	*1400*
7:15am	*3500*	*1700,*	*1300,*	*1200*
7:30am	*3400*	*1800,*	*1400,*	*1100*
7:45am	*3100*	*1800,*	*1500,*	*1000*
8:00am	*2800*	*1700,*	*1300,*	*900*
8:15am	*2600*	*1500,*	*1300,*	*800*
8:30am	*2500*	*1400,*	*1200,*	*800*
8:45am	*2500*	*1300,*	*1100,*	*700*

A traffic simulation was run in Aimsun under the traffic demands of Table 5.6, which

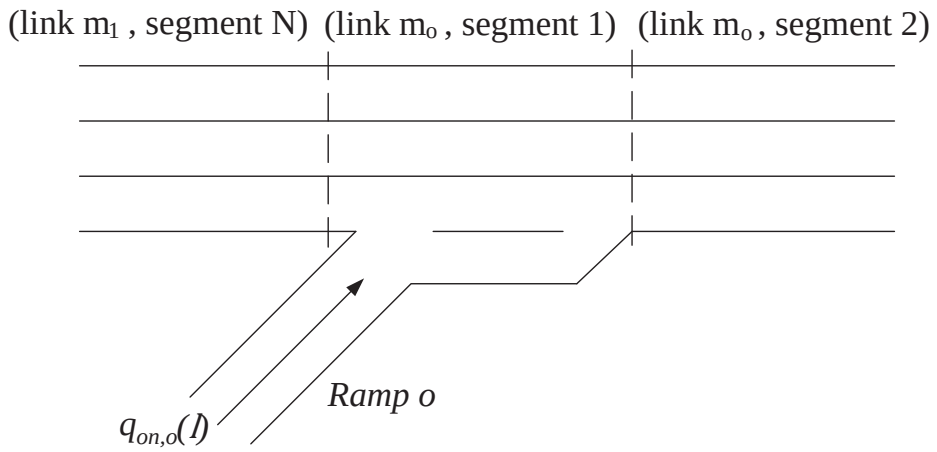
were adapted from the field data provided by NZTA (New Zealand Traffic Agency) collected by SCATS (Sydney Coordinated Adaptive Traffic System). The turning proportions at off-ramps (the percentages of exiting traffic from three off-ramps) were set to 10%, 15% and 15%. The detection interval was set to five minutes, so the statistical data were collected and recorded by the installed detectors from all roadway segments of the motorway stretch every five minutes[6]. Eq. (5.4) was solved by the GA function of the optimization toolbox of MATLAB. The identified parameters are given in Table 5.7.

Table 5.7 Identified parameters from mainline traffic

	Link1	**Link2**	**Link3**	**Link4**	**Link5**	**Link6**	**Link7**	**Link8**
t_m	0.0055	0.0058	0.0051	0.0049	0.0056	0.0042	0.0051	0.0053
c_m	34.75	14.51	2.12	100.51	15.13	150.29	20.31	120.97
k_m	12.31	12.16	13.51	13.10	12.71	12.45	13.21	13.13

5.4.1.3 Identification of δ_m and w_o

δ_m is the tuning parameter of the merging term, Eq. (3.7). w_o is a new weighting parameter inserted to Eq. (3.10) to ensure that ramp queues propagated as possible as the simulation did. The formation of ramp queues in Aimsun was estimated by Eq. (5.5) and (5.6).



[6] Aimsun is a microscopic stochastic simulator, so simulation results may oscillate within a small range if the detection interval is too short.

Figure 5.14 The layout of an on-ramp in the macroscopic model

$$w_o(l+1) = w_0(l) + T_s(D_o(l) - q_{on,o}(l)) \tag{5.5}$$

$$q_{on,o}(l) = Min[D_o(l) + \frac{w_o(l)}{T_s}, Q_{cap,o} Min\left(1, \frac{\rho_{jam} - \rho_{m_{o,1}}(l) - w_o}{\rho_{jam} - \rho_{queue}}\right) \tag{5.6}$$

Here, Eq. (5.5) remains the same as Eq. (3.9) in Section 3.1.1. Eq. (5.6) is a modified version of Eq. (3.10). In Eq. (5.6), $\rho_{queue}$is an important density value that caused the formations of ramp queues in the macroscopic model. Ramp queues start forming when the downstream density ($\rho_{m_{o,1}}(l)$) exceeds this value. ρ_{queue} was set to 23veh/km/lane, based on the observation of simulation results from Aimsun, which was much lower than the critical density (ρ_{crit}= 33veh/km/lane) used in Eq. (3.10). w_o is a weighting parameter that adjusts the charge and discharge rates of the ramp queue for the o-th on-ramp. w_o is given by two different values for the different situations,

$$w_o = \begin{cases} w_{o1}, when\ \rho_{m_{o,1}}(l) > \rho_{crit} \\ w_{o2}, when\ \rho_{m_{o,1}}(l) \leq \rho_{crit} \end{cases} \tag{5.7}$$

By using traffic demands of Table 5.6, a simulation was run in Aimsun and in calibrated macroscopic model respectively. Turning proportions at off-ramps were set to 10%, 15% and 15%. δ_m and $w_o$can be identified by adjusting and comparing queue lengths between Aimsun and the macroscopic traffic model (Table 5.8).

Table 5.8 Identified parameters from the formation of ramp queues

	w_{o1}	w_{o2}	$\boldsymbol{\delta_m}$
Greville Ramp(1)	*19*	*17*	*1.5*
Constellation Ramp(2)	*52*	*50*	*1.8*
Trisram Ramp(3)	*55*	*50*	*2.5*

The changes of the ramp queues in different traffic models were compared as shown in Fig. 5.15~ Fig. 5.17. The downstream density of each on-ramp was also compared, since how ramp queues in the macroscopic model are accumulated is based on the changes of downstream density. Ramp queue lengths in Aimsun were statistical results collected every five minutes, and simulation results from the calibrated model were

recorded every minute. In addition, the lengths of all on-ramps in Section 5.4 had been extended to 600m in Aimsun, since original lengths of on-ramps in Fig 5.6~5.8 are not long enough to demonstrate the full queue lengths of simulations. Also note that the recorded queue lengths are the sum of two queue lengths from both lanes, since each on-ramp in the simulations includes two lanes.

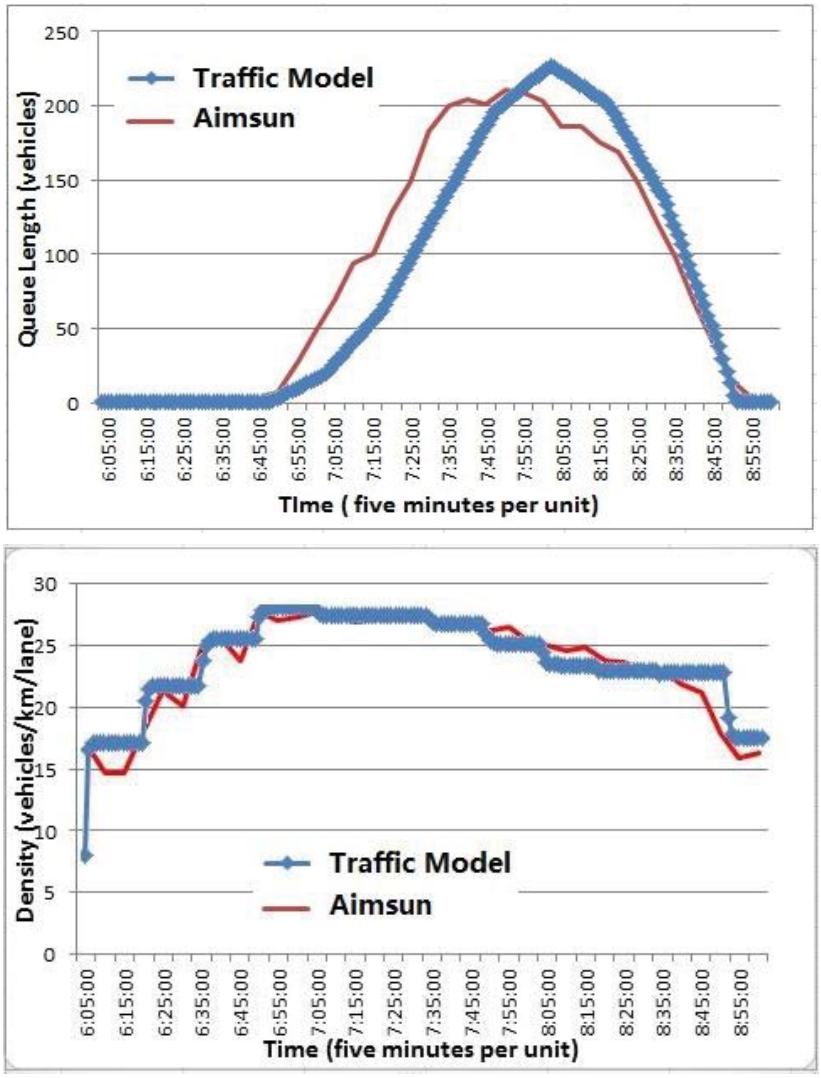


Figure 5.15 Comparison of simulation results on ramp 1 (Greville Road southbound on-ramp)

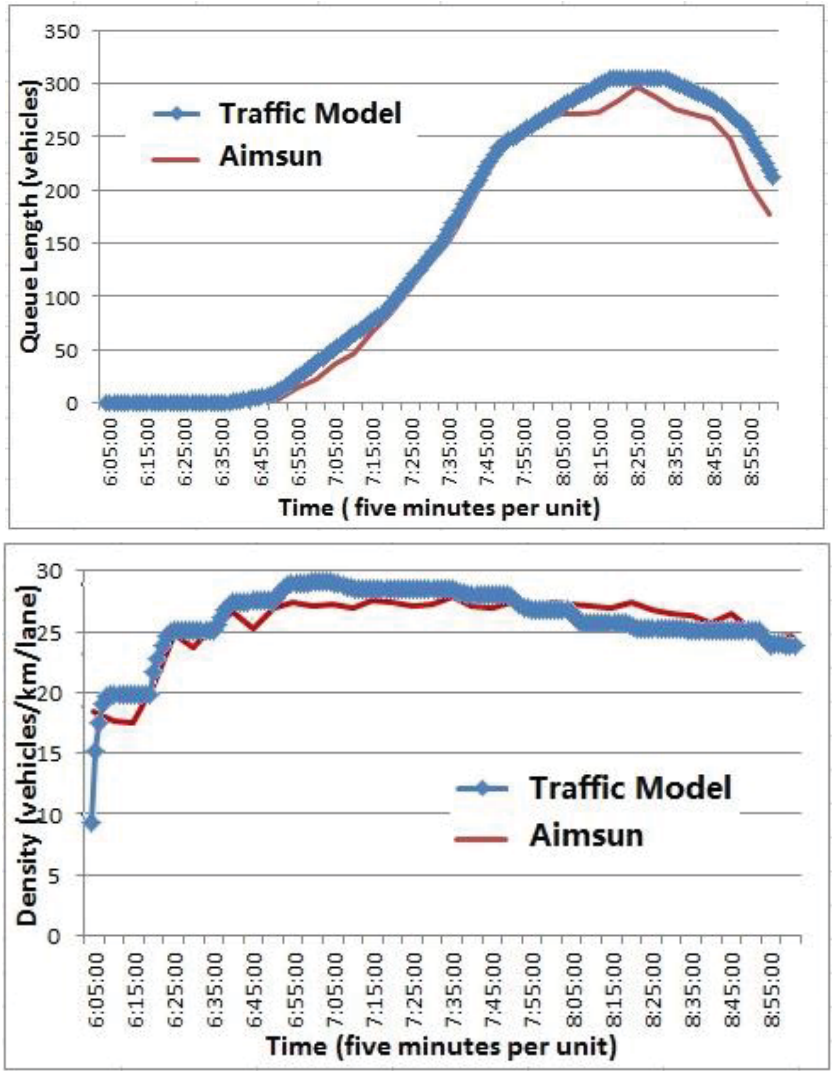
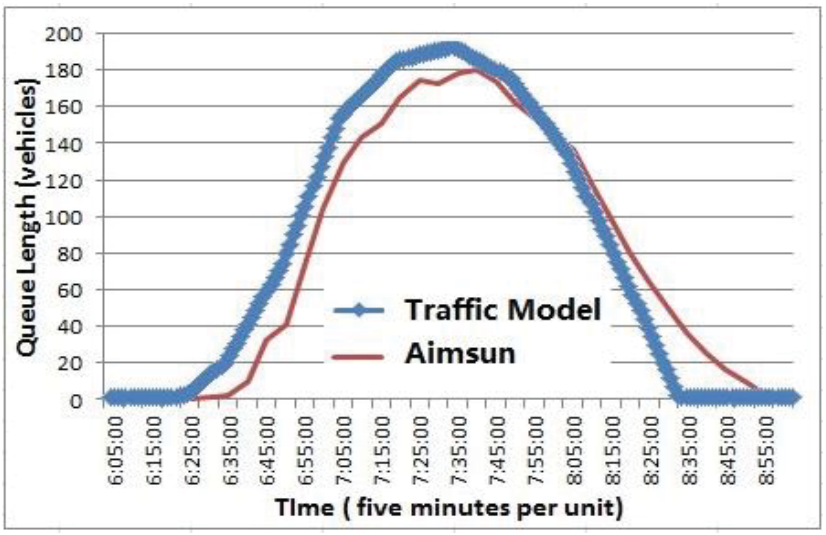


Figure 5.16 Comparison of simulation results on ramp 2 (Constellation Drive southbound on-ramp)



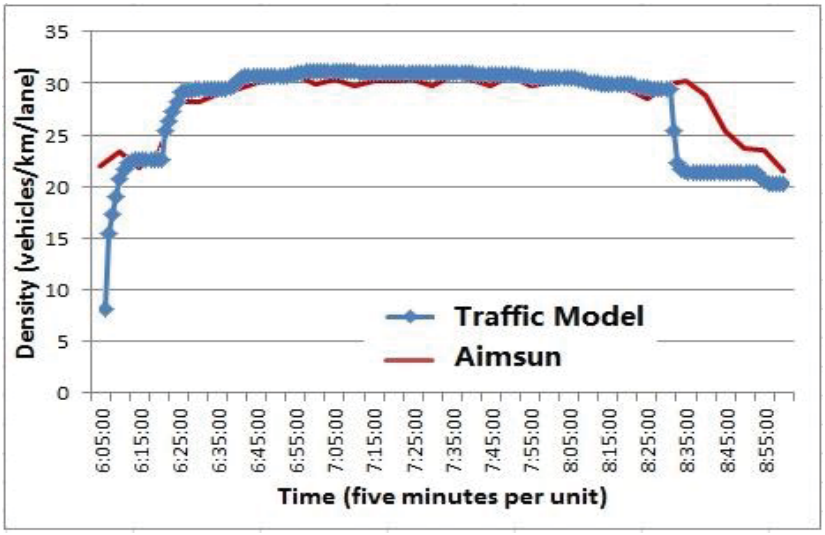


Figure 5.17 Comparison of simulation results on ramp 3 (Tristram Avenue southbound on-ramp)

A can be seen from Fig. 5.15 to Fig. 5.17, the calibrated traffic model generates the curves of downstream density similar as those in Aimsun. Ramp queues in both simulations started increasing when the downstream density exceeded about 23veh/km/lane and decreasing when ramp demands decreased.

5.4.2 The Identified Parameters

The parameter identification procedure in 5.4.1 can be depicted by the following flow chart (Fig. 5.18).

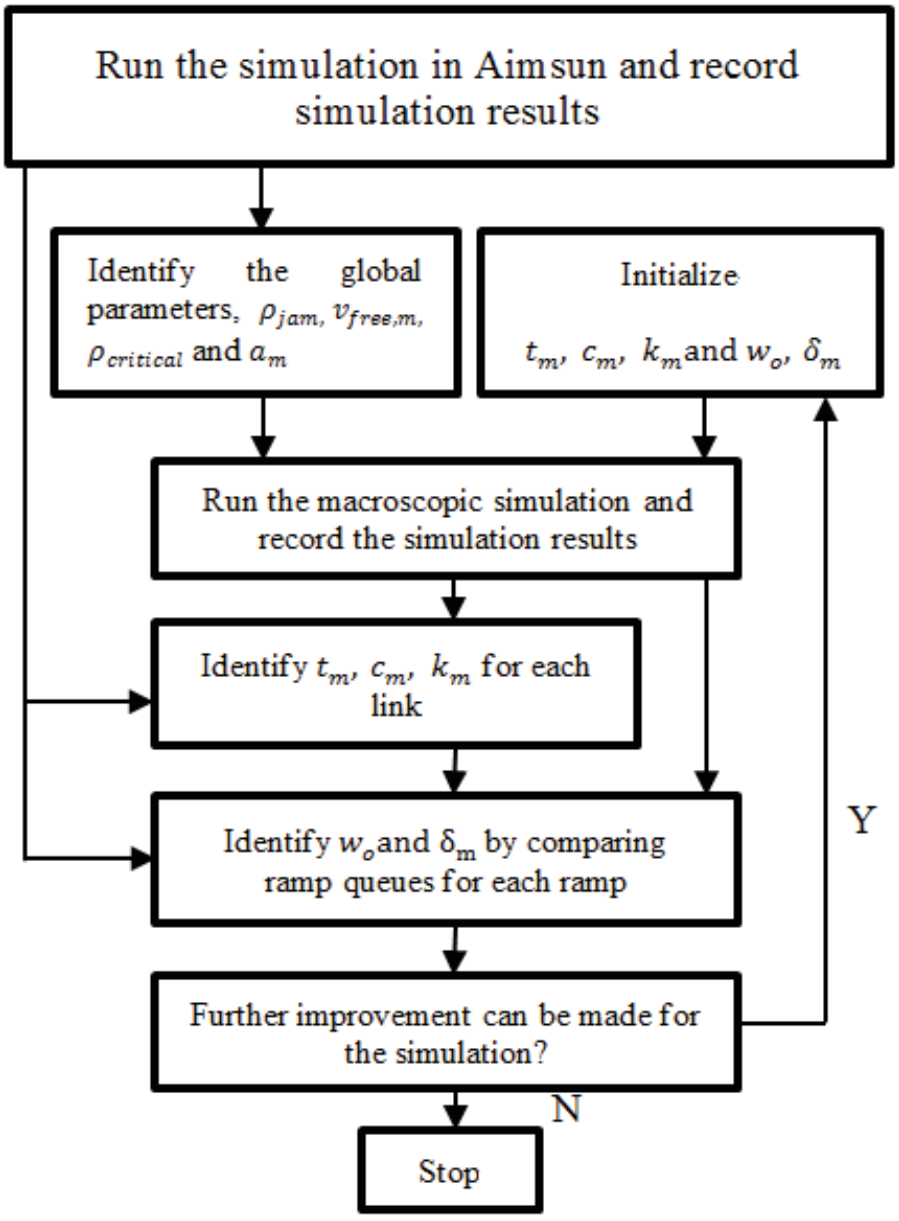


Figure 5.18 The parameter identification procedure

The identified parameters are summarized in Table 5.9.

Table 5.9 Identified parameters

	Link1	Link2	Link3	Link4	Link5	Link6	Link7	Link8
t_m	*0.0055*	*0.0058*	*0.0051*	*0.0049*	*0.0056*	*0.0042*	*0.0051*	*0.0053*
c_m	*34.75*	*14.51*	*2.12*	*100.51*	*15.13*	*150.29*	*20.31*	*120.97*
k_m	*12.31*	*12.16*	*13.51*	*13.10*	*12.71*	*12.45*	*13.21*	*13.13*

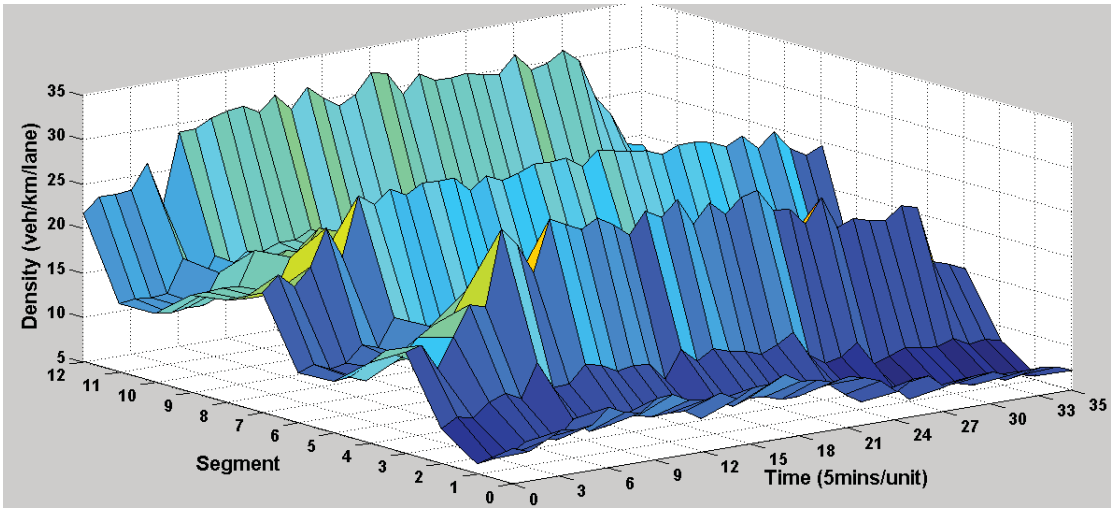
	Greville Ramp(1)	Constellation Ramp(2)	Tristram Ramp(3)
w_{o1}	*19*	*52*	*55*
w_{o2}	*17*	*50*	*50*
δ_m	*1.5*	*1.8*	*2.5*

	UNIT	Empirical Value
ρ_{jam}	***Veh/km/lane***	*180*
a_m		*1.7*
$v_{free,m}$	***Veh/hour***	*110*
$\rho_{critical}$	***Veh/km/lane***	*33*
ρ_{queue}	***Veh/km/lane***	*23*

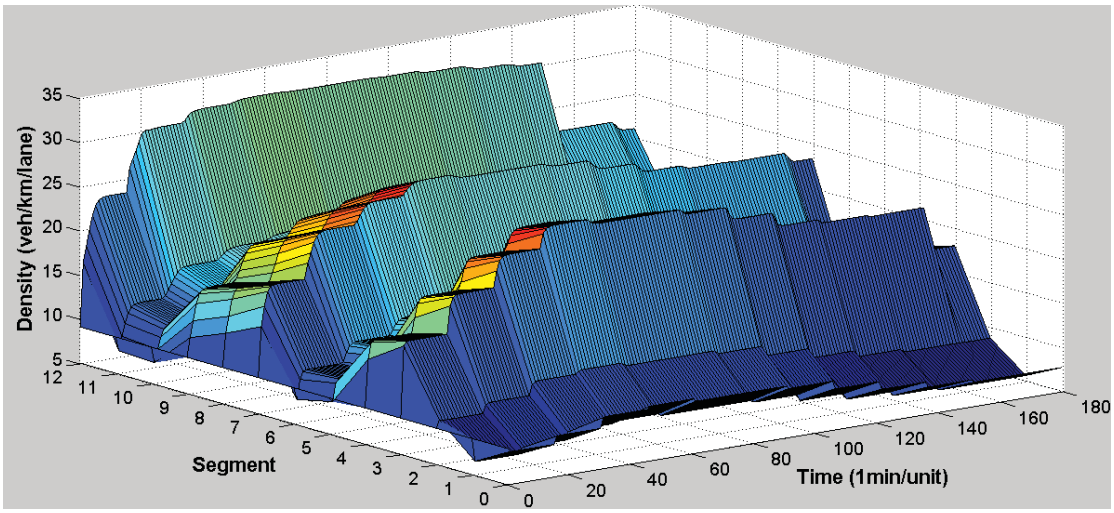
5.4.3 Model Validation

By using traffic demands in Table 5.6 and identified parameters in Table 5.9, the calibrated macroscopic model was simulated to compare with Aimsun. The duration of simulation was three hours from 6:00am to 9:00am. The turning proportions at off-ramps were set to 10%, 15% and 15%. Simulation results in Aimsun were collected every five minutes, and simulation results from the calibrated model were recorded every minute.

The density profile from Aimsun is shown in Fig. 5.19a and the density profile from the traffic model is shown in Fig. 5.19b. Comparison of the speed profiles is given in Fig. 5.20. To ensure that the identified traffic model can effectively estimate the traffic conditions when mainline traffic is under uncongested conditions, more simulations were conducted under different traffic demands (Table 5.10).

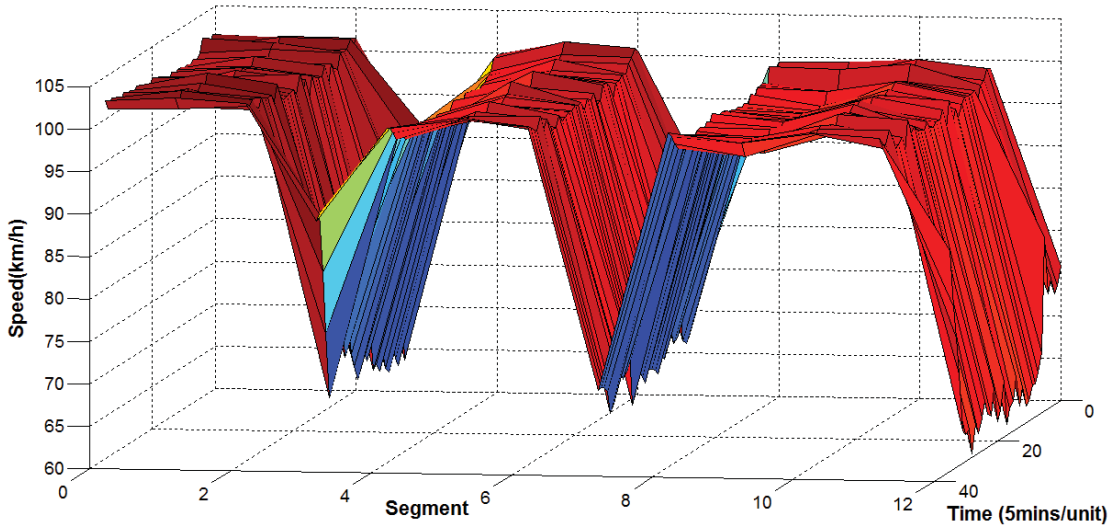


a) The density profile in Aimsun

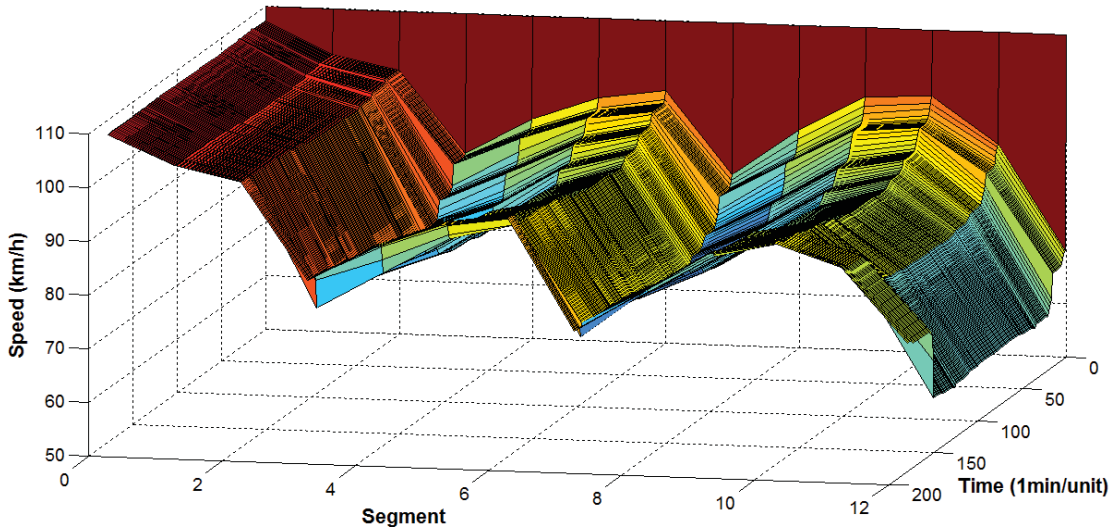


b) The density profile in the macroscopic model

Figure 5.19 Comparison of the density profiles along the motorway stretch



a) The speed profile in Aimsun



b) The speed profile in the macroscopic model

Figure 5.20 Comparison of the speed profiles along the motorway stretch

Comparison of the density profiles in Fig. 5.19 indicates that the calibrated model is able to estimate traffic conditions along the motorway stretch, where three bottleneck locations can be found at segment 3, segment 7 and segment 12 in both graphs. Both density profiles are below the critical density (33veh/km/lane), so no congestion occurred during the simulations. The same bottleneck locations can be identified from the speed profiles in Fig. 5.19, which also indicates that speeds between bottleneck locations recovered more slowly in the calibrated model than speeds in Aimsun. That is because Eq. (5.4) did not aim at speeds but at densities due to the fact that traffic speeds, which are not directly involved to calculate TTS values (Eq. (3.13)) and queue lengths (Eq. (3.09) and Eq. (3.10)), are relatively less important than traffic densities.

By increasing traffic demands at the origin between 7am to 7:30am (Table 5.10), the other three experiments had been conducted. The comparison of the corresponding changes of density profiles can be seen in Fig. 5.21~ Fig. 5.23.

Table 5.10 Traffic demands at origin

	7:00am	**7:15am**	**7:30am**
Experiment 1	*4500*	*4500,*	*4000,*
Experiment 2	*5500*	*5500,*	*5000,*
Experiment 3	*6500*	*6500,*	*6000,*

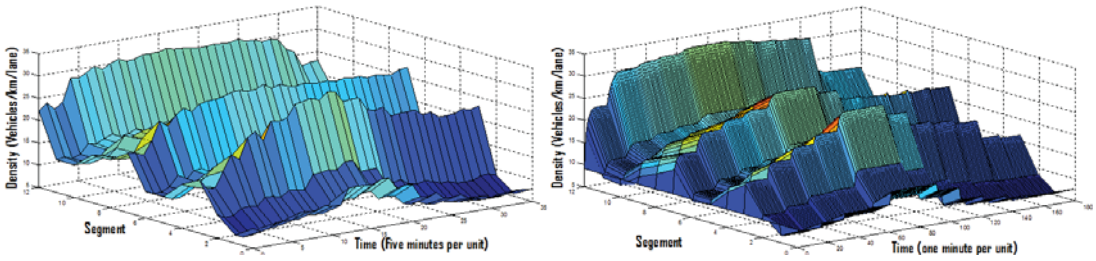


Figure 5.21 Comparison of the density profiles (Experiment1)

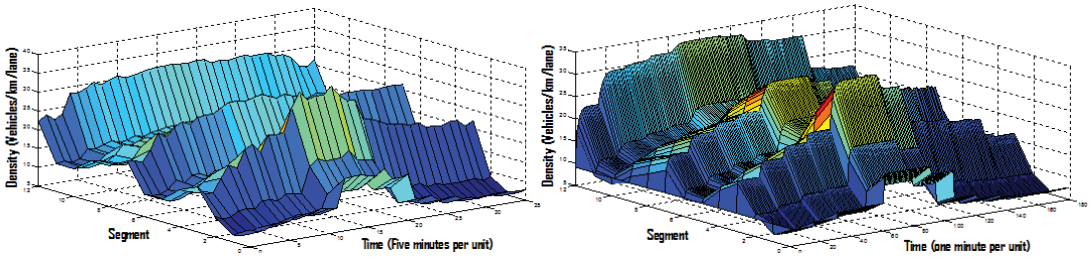


Figure 5.22 Comparison of the density profiles (Experiment2)

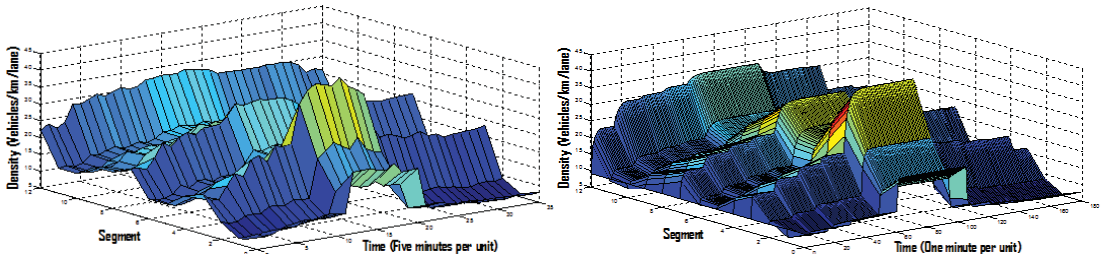


Figure 5.23 Comparison of the density profiles (Experiment3)

The experiments show that local congestions occurred in experiment 2 and experiment 3 when traffic demands exceeded 5500veh/h. The comparison of density profiles under different traffic demands indicates that the calibrated traffic model was able to estimate local congestions occurred at bottleneck locations in Aimsun and generate density profile similar as those in Aimsun when the traffic densities on the mainline is under 45veh/hour/lane.

Changes of queue lengths of each on-ramp in experiment 3 are given by Fig. 5.24~Fig. 5.26. Left graphs show simulation results from Aimsun.

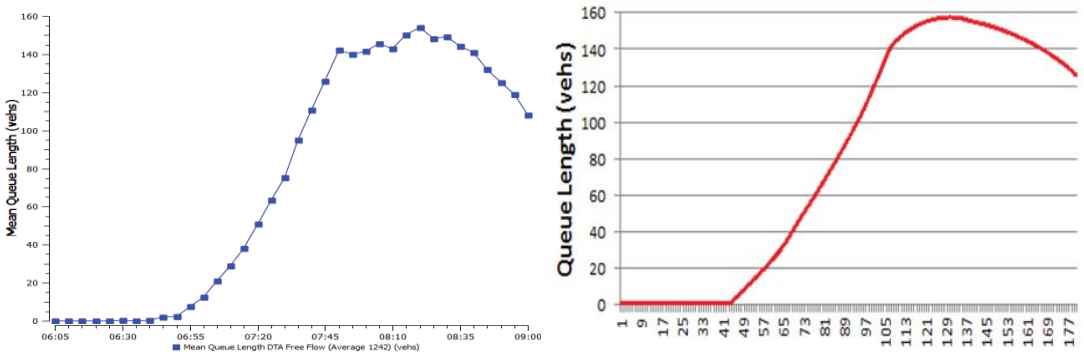


Figure 5.24 Comparison of queue lengths on ramp 1 (Experiment1)

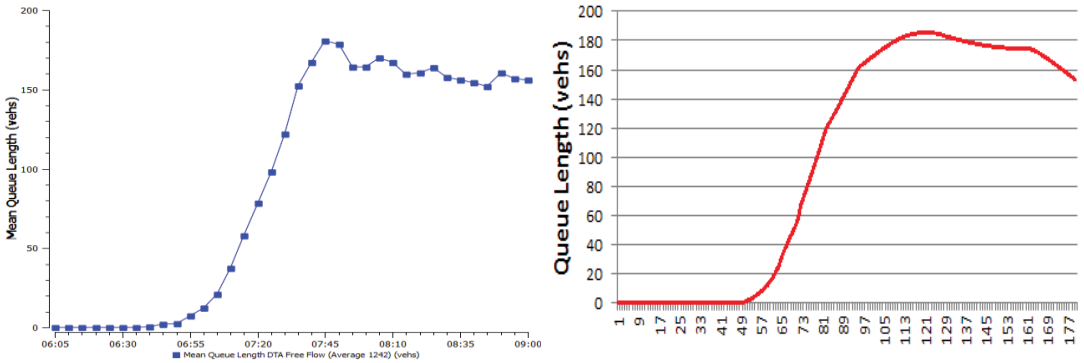


Figure 5.25 Comparison of queue lengths on ramp 2 (Experiment2)

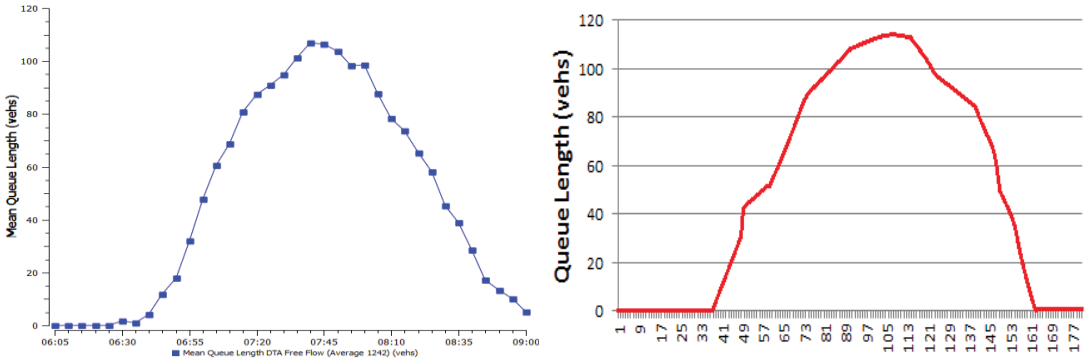


Figure 5.26 Comparison of queue lengths on ramp 3 (Experiment3)

5.5 Simulations under High Traffic Demands and Analysis of Simulation Results

In this section, the proposed DP ramp metering algorithms were implemented in the Aimsun scenario under high traffic demands (Table 5.11) as shown in Fig. 5.3. The traffic demands for each on-ramp were adapted from the traffic counts provided by NZTA, but traffic demands at origin were enlarged[7], since when traffic demands were

[7] The reason why traffic demands provided by NZTA are not enough to cause congestions is because congestions in the study location at peak hours are not caused by the upstream demands but by the disturbances from downstream bottleneck locations (e.g. The Auckland Harbor Bridge).

lower than 5500veh/h, there was no congestion occurred in the study location and ramp metering would be unnecessary. Also, the simulator presumes that the downstream of the simulation scenario is under free-flow conditions.

The simulation results were present by speed and density profiles, and also compared to those under no-ramp-control case and those under the control of ALINEA. The performances of ramp metering algorithms were measured by the percentage improvement of TTS values calculated by Aimsun, comparing to no-ramp-control case.

TTS values in Aimsun were calculated by Eq. (5.8) (TSS, 2008).

$$TotalTravelTime_{sys} = \sum_{i=1}^{N_{Sys}} TEX_i - TEN_i \quad (5.8)$$

Where, $\mathrm{TotalTravelTime_{sys}}$ is the total travel time experienced by all the vehicles during the simulation. TEN_i is the entrance time of the i-th vehicle in the simulation scenario. TEX_i is the exit time of the i-th vehicle from the simulation scenario. N_{Sys} is the total number of vehicles demanding to cross the network during the simulation.

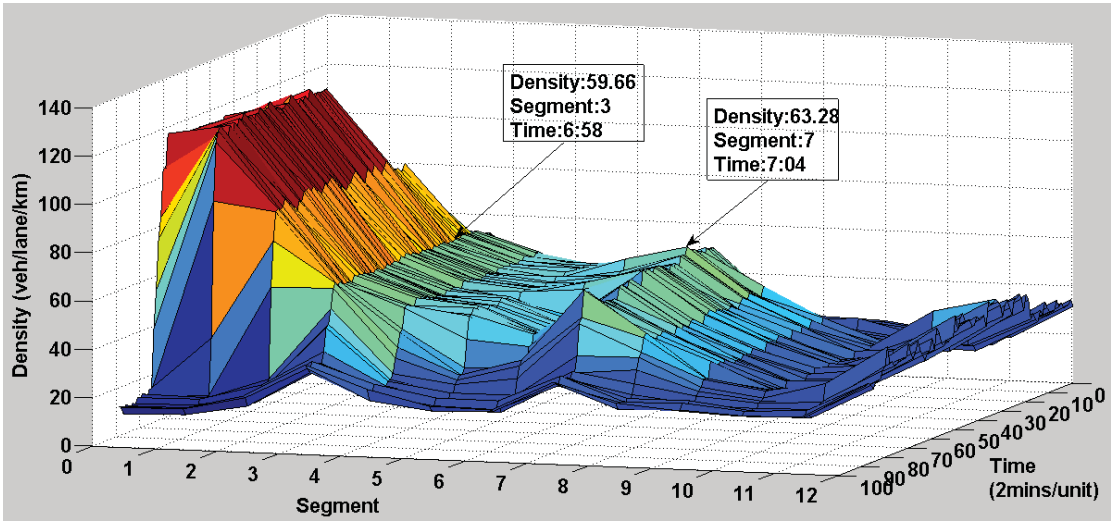
Table 5.11 High traffic demands

Time	Origin	Greville Ramp(1)	Constellation Ramp(2)	Trisram Ramp(3)
6:00am	*4000*	*1000*	*1000*	*1000*
6:15am	*5000*	*1300*	*1200*	*1200*
6:30am	*6000*	*1400*	*1300*	*1300*
6:45am	*6500*	*1600*	*1300*	*1400*
7:00am	*6500*	*1600*	*1400*	*1400*
7:15am	*6000*	*1700*	*1500*	*1300*
7:30am	*5500*	*1700*	*1500*	*1200*
7:45am	*5500*	*1600*	*1400*	*1200*
8:00am	*5000*	*1500*	*1300*	*1100*
8:15am	*4500*	*1400*	*1300*	*1100*
8:30am	*4000*	*1400*	*1200*	*1000*
8:45am	*3500*	*1300*	*1100*	*1000*

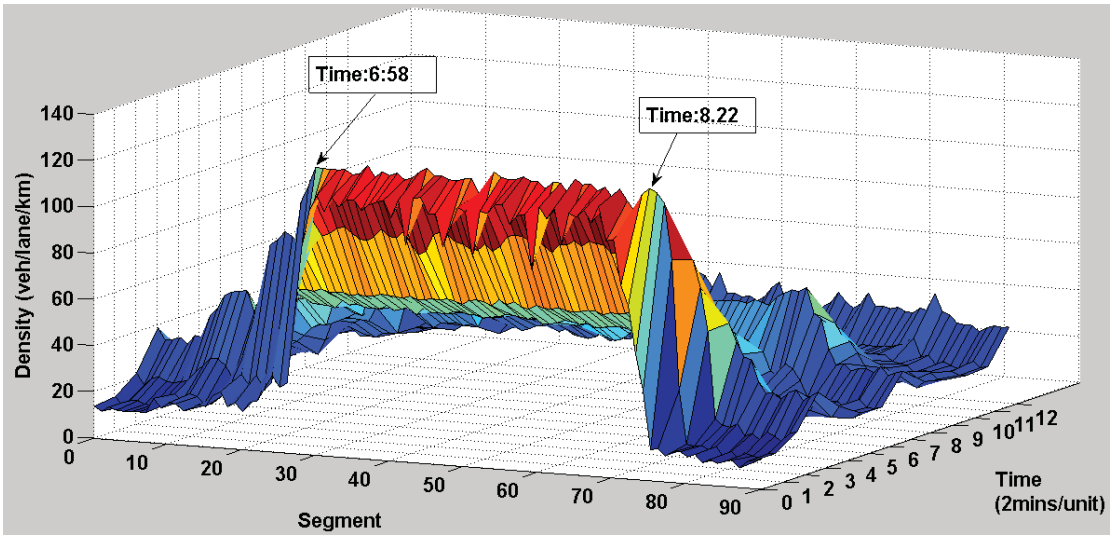
Turning proportions at off-ramps (the percentages of exiting traffic from three off-ramps) were set to 10%, 15% and 15%. The detection interval in Aimsun was set to 2mins, so simulation results were recorded every 2 minutes. No queue constrains were applied to DP ramp metering, since ramp lengths in the Aimsun scenario are too short and may cause overstrict queue restrictions.

5.5.1 Simulation Results of No-ramp-control Case

The density profile and the speed profile were both plotted in 3D graphs (Fig. 5.27~ ~5.26).

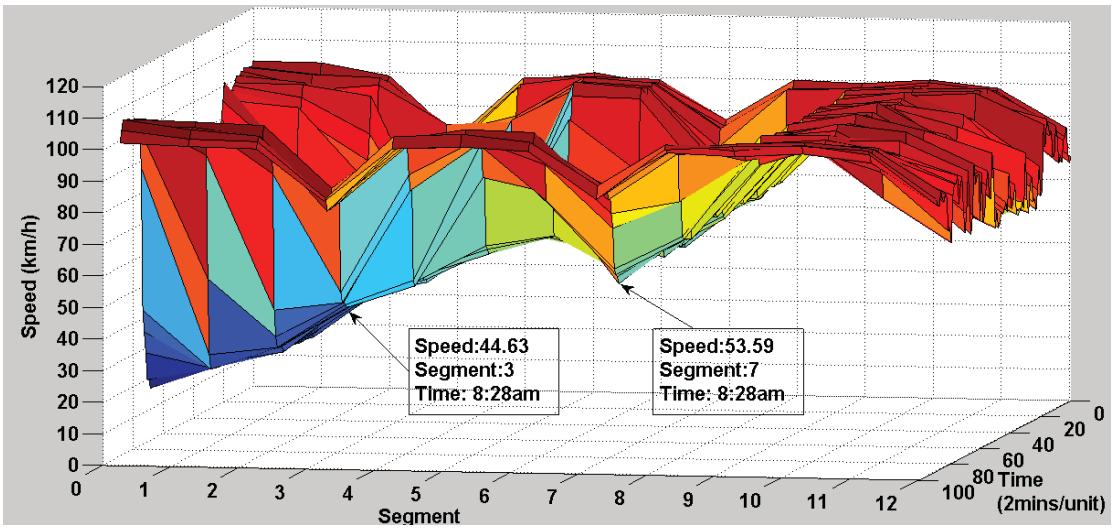


a) the density profile (side view)

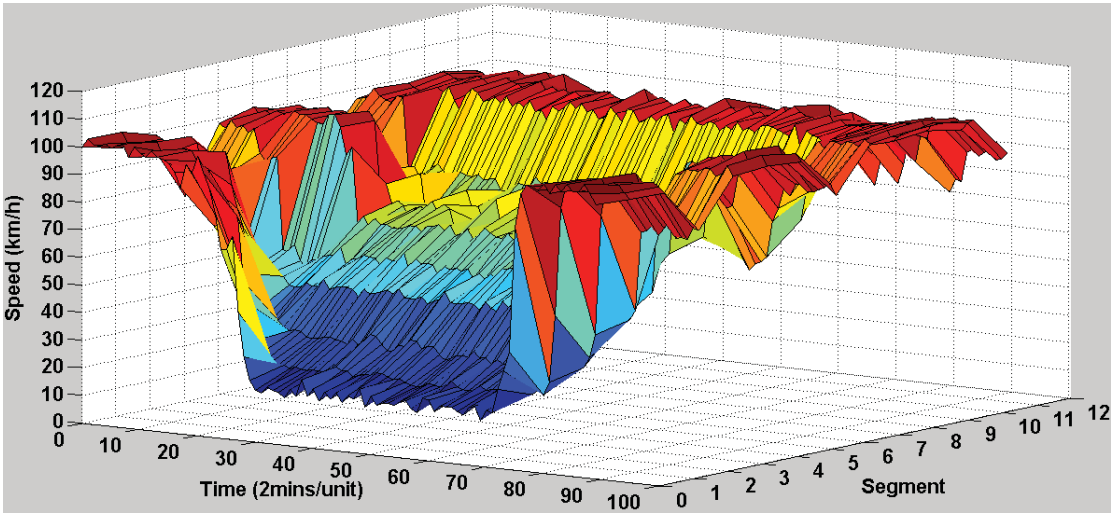


b) the density profile (front view)

Figure 5.27 The density profile under no-ramp-control case (Aimsun)



a) the speed profile (side view)



b) the speed profile (front view)

Figure 5.28 The speed profile under no-ramp-control case

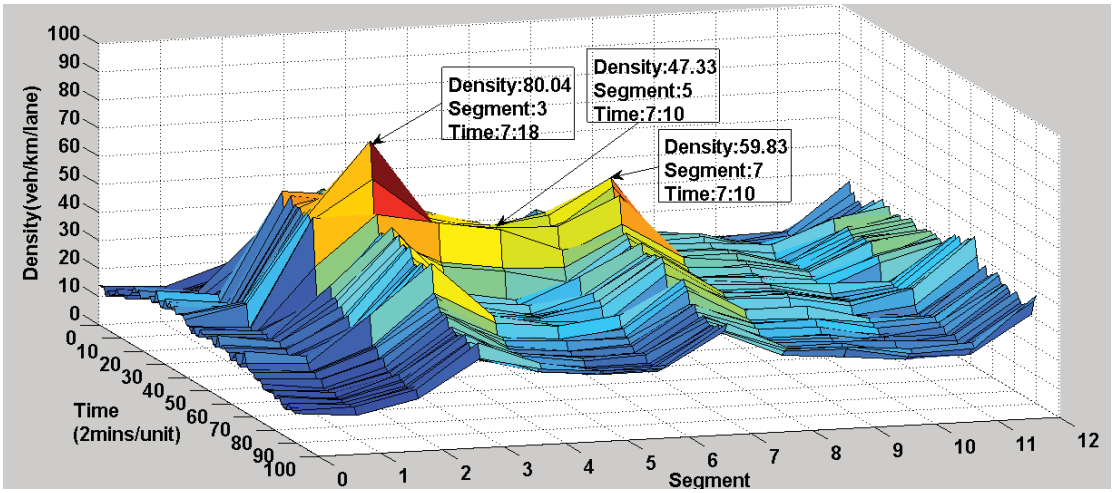
As can be seen in Fig. 5.27b, heavy traffic congestion occurred from about 6:58am to about 8:22am, where motorway queues emerged at the downstream of ramp 2 (segment 7) with densities of around 63km/veh/lane, being propagated to the origin (segment 0) with densities of about 110km/veh/lane (Fig. 5.27a), dissipating in front of ramp 3(segment 12).

During the congestion, traffic flow was oversaturated from the origin to the downstream of ramp 2 (segment 7), whose densities decreased from 110veh/km/lane to 60veh/km/lane (Fig. 5.27a) and whose speeds increased from 20km/h to 53km/h (Fig. 5.28a). Then, from the downstream of ramp 2 (segment 7) to segment 10, speeds

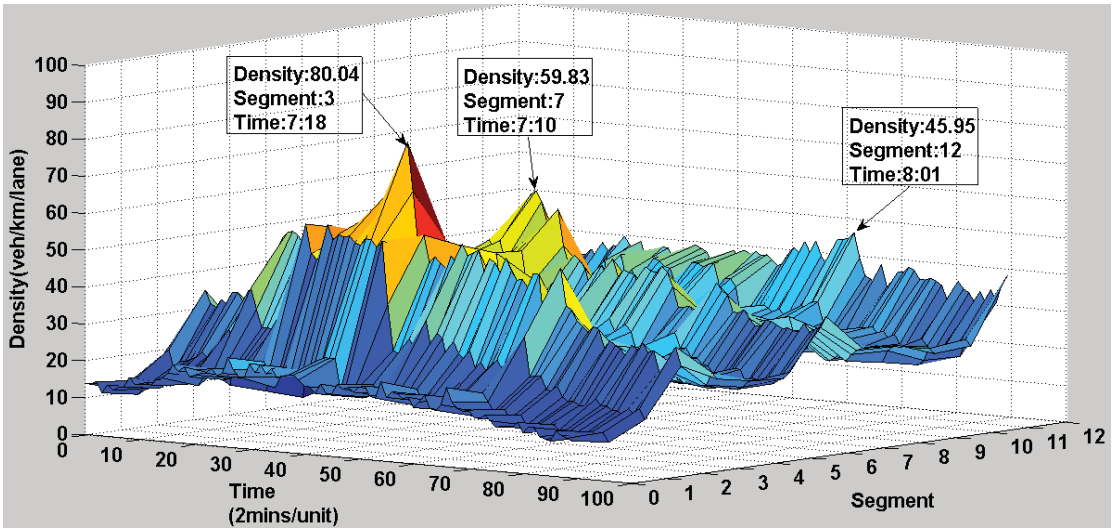
tended to restore within a range of 53km/h to 100km/h (Fig. 5.28a), where motorway queues were dissipating with densities decreasing from 53veh/km/lane to about 10veh/km/lane (Fig. 5.27a). Traffic flow remained undersaturated afterwards, even with the disturbances from ramp 3 where densities increased to about 30veh/km/lane and speed decreased to about 80km/h (Fig. 5.27a).

5.5.2 Simulation Results of ALINEA

The density profile and the speed profile were both plotted in 3D graphs (Fig. 5.29~5.30).

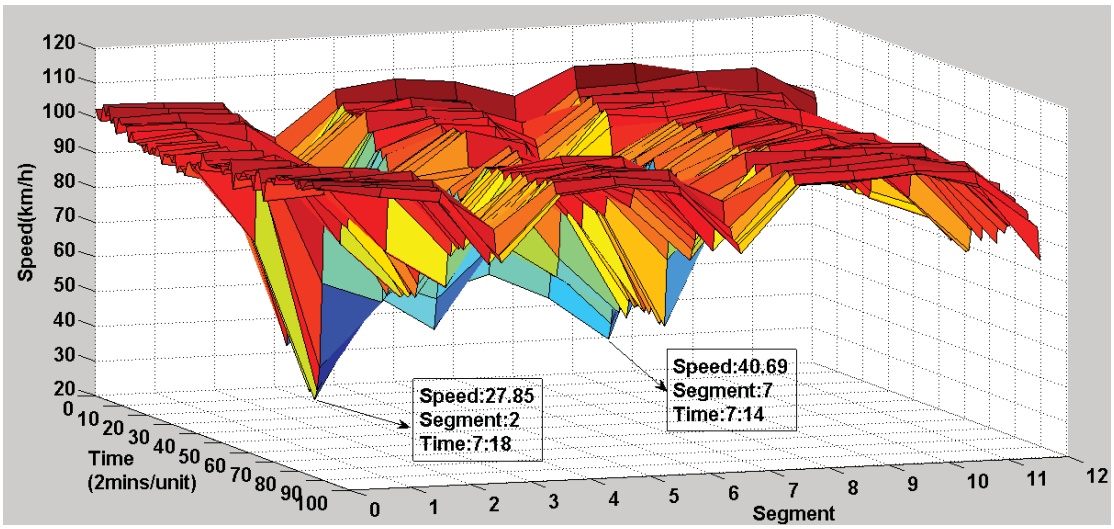


a) the density profile (side view)

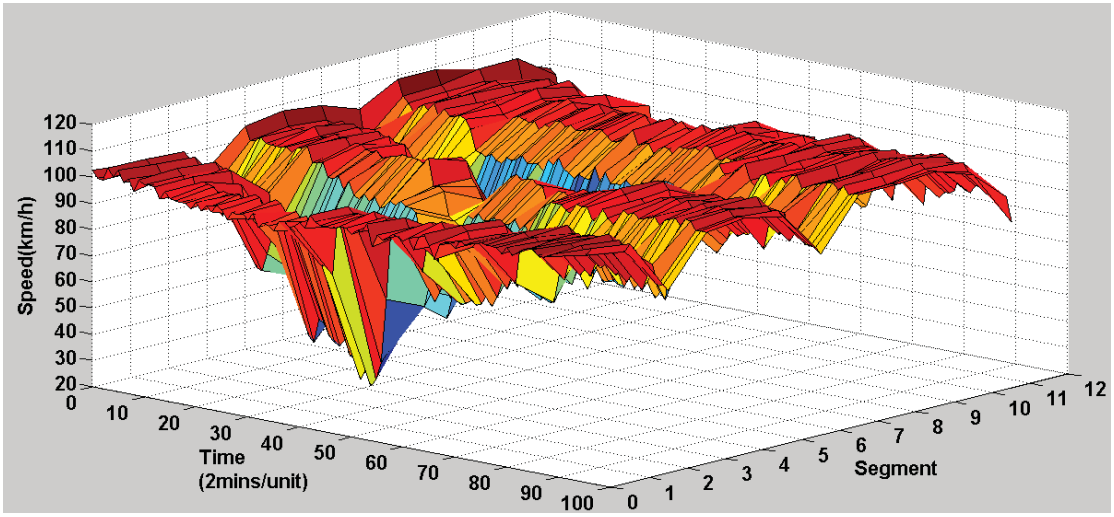


b) the density profile (front view)

Figure 5.29 The density profile under the control of ALINEA (Aimsun)



a) the speed profile (side view)



b) the speed profile (front view)

Figure 5.30 The speed profile under the control of ALINEA (Aimsun)

The bottleneck locations can be identified at the downstream of ramp 1 (segment 3), the downstream of ramp 2 (segment 7) and the downstream of ramp 3 (segment 12). The density profile in Fig. 5.29 ranges from 10veh/km/lane to 80veh/km/lane. The majority of the density profile along bottleneck locations is within a range between 10veh/km/lane and 50veh/km/lane (Fig. 5.29a).

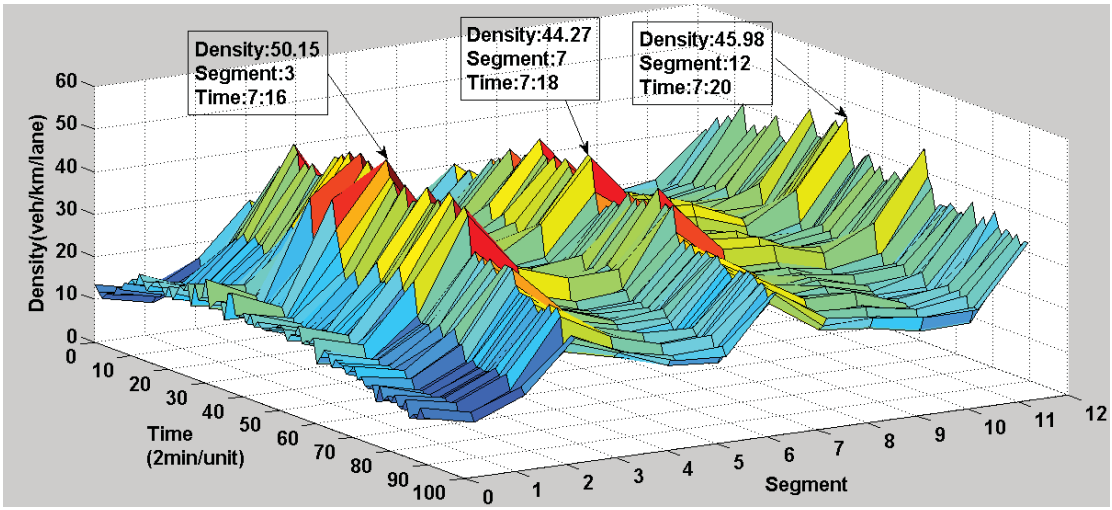
The motorway queues can be observed between the downstream of ramp 1 (segment 2) and the downstream of ramp 2 (segment 7) at about 7:15am (Fig. 5.29a). The motorway queues emerged from ramp 2 (segment 7) with a peak density of 59.83veh/km/lane, being propagated to ramp 1 (segment 3) and peaking at a density

of 80.04veh/km/lane, dissipating before reaching ramp 3 (segment 12). The reason of the formation of motorway queues could be explained by Table 5.11, where both traffic demands for ramp1 and ramp2 reached the highest values from 7:15am to 7:30am while uncoordinated ramp signals failed to restrict ramp inflows at ramp 2 and caused the propagation of the motorway queues.

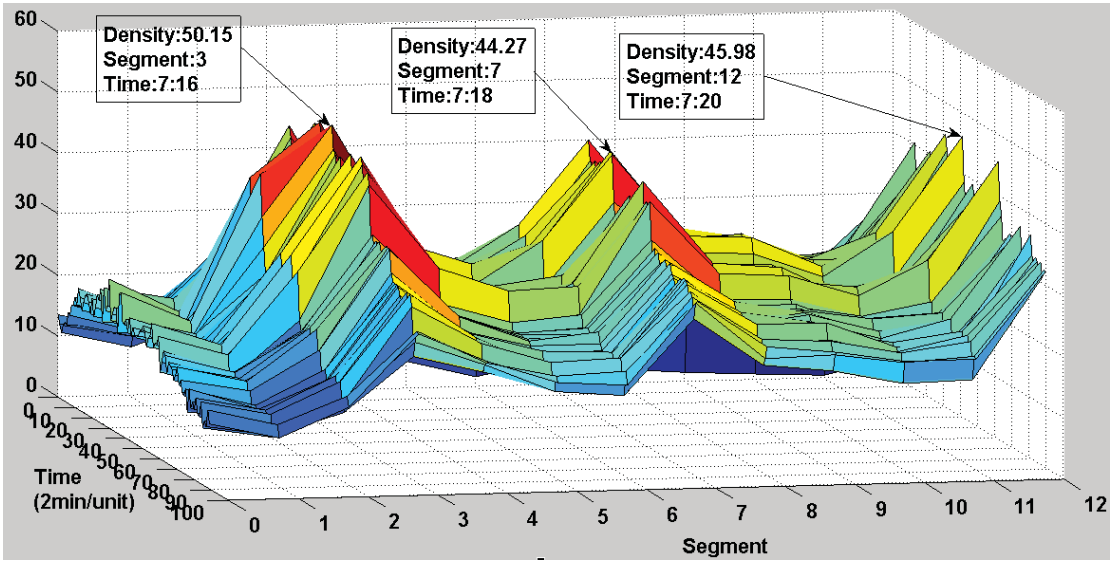
The speed profile in Fig. 5.30 reveals the further details of motorway queues. As shown in Fig. 5.30a, the speeds dropped to 27.85km/h at the upstream segment of ramp 1 (segment 2) and restored to about 50km/h before dropping to 40.69km/h (segment 7) due to the disturbances from ramp 2. Clearly, motorway queues were propagated between ramp 1 and ramp2. From segment 6 to segment 10, speeds restored to around 100km/h before reaching ramp 3 (segment 12) so no motorway queue was propagated between ramp 2 and ramp 3.

5.5.3 Simulation Results of DP based Ramp Metering

This subsection presents the simulation results of two experiments with regard to DP based ramp metering algorithms. The first experiment only implemented a simplified DP network (the first phase of search), in which each ramp meter only had 9 released metering rates. The second one applied two phases of search, in which each ramp meter had 37 released metering rates. The density profiles and the speed profiles of the first experiment were plotting in Fig. 5.31~5.32.



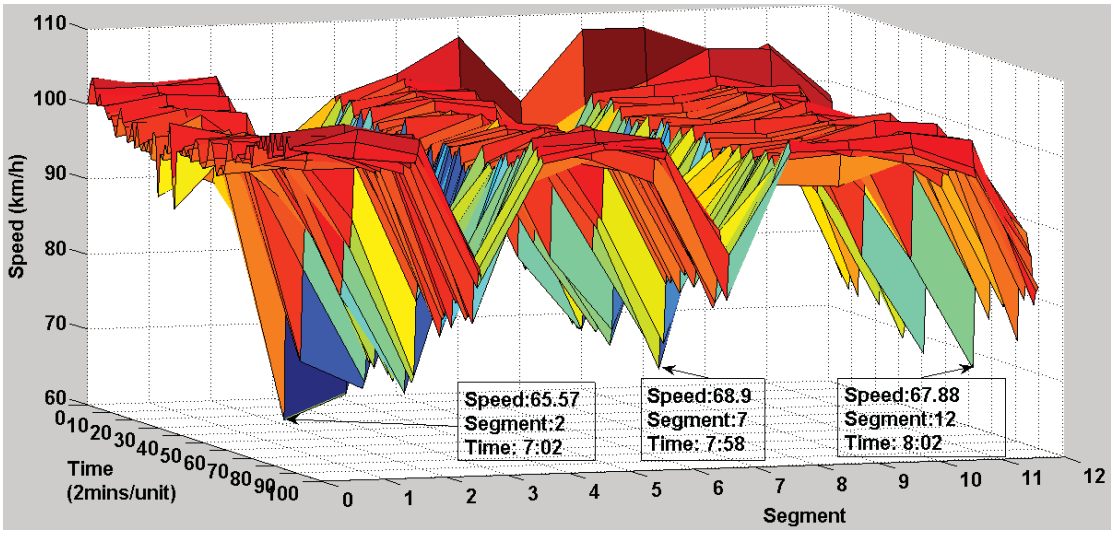
a) the density profile (front view)



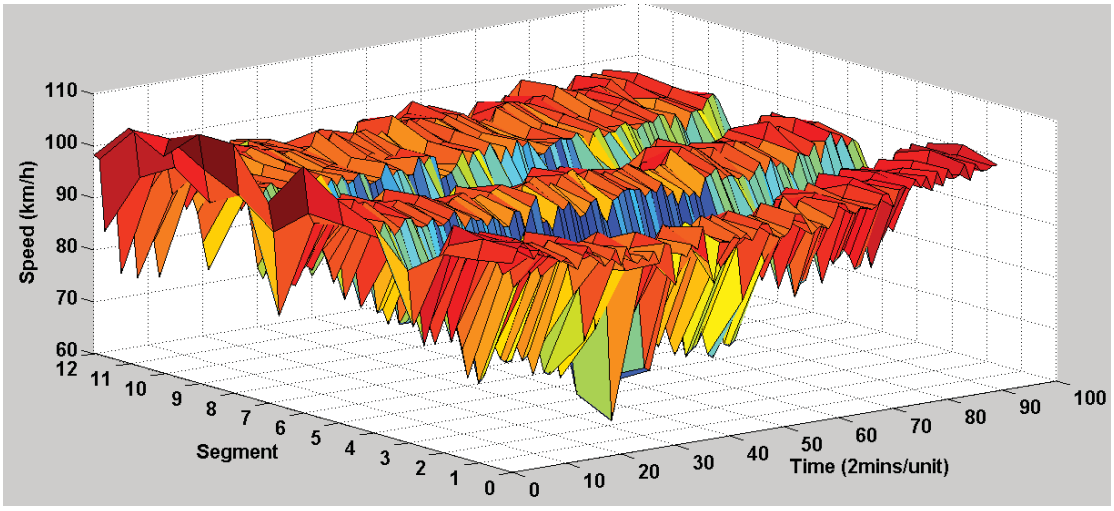
b) the density profile (side view)

Figure 5.31 The density profile under the control of DP (the first phase of search)

In Fig 5.31, the bottleneck locations can be identified at the downstream of ramp 1 (segment 3), the downstream of ramp 2 (segment 7) and the downstream of ramp 3 (segment 12). The density profile ranges from 10veh/km/lane to 50veh/km/lane. The profiles between the bottleneck locations were below the critical density (33veh/km/lane), so no motorway queue was propagated between on-ramps. Local congestions can be identified along the bottleneck locations where densities may be higher than the critical density due to the disturbances of ramp inflows (Fig. 5.31).



a) the speed profile (side view)



b) the speed profile (front view)

Figure 5.32 The speed profile under the control of DP (the first phase of search)

The further details can be observed by the speed profile in Fig 5.32. The speed profile between on-ramps stabilized around the free-flow speed (100km/h), so no motorway queue was propagated between on-ramps during the simulation and traffic flow between on-ramps remained under free-flow conditions.

Local congestions can be identified at the upstream segment of ramp 1 (segment 2), the downstream segment of ramp 2 (segment 7) and the downstream segment of ramp 3 (segment 12) (Fig. 5.32a), where lowest speeds can be found at 65.57km/h, 68.9km/h and 67.88am, respectively. Since speeds along the bottleneck locations tended to restored to the free-flow speed from about 60km/h, so traffic flow along the bottleneck locations can be identified as queue discharge flow (referring to section 1.1).

The density profiles and the speed profiles of the second experiment (two phases search) were plotting in Fig. 5.33~5.34.

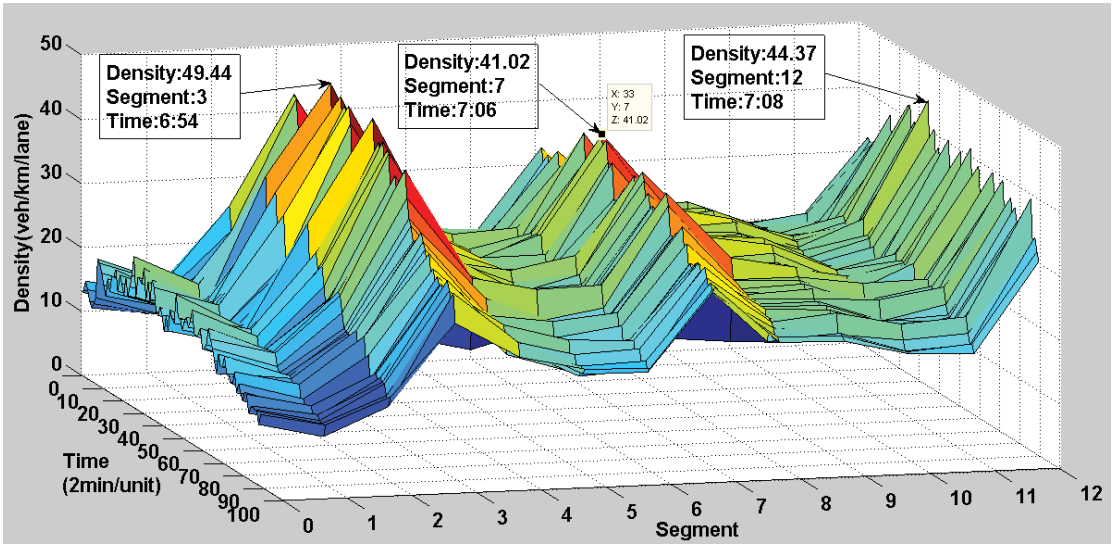


Figure 5.33 The density profile under the control of DP (two phases of search)

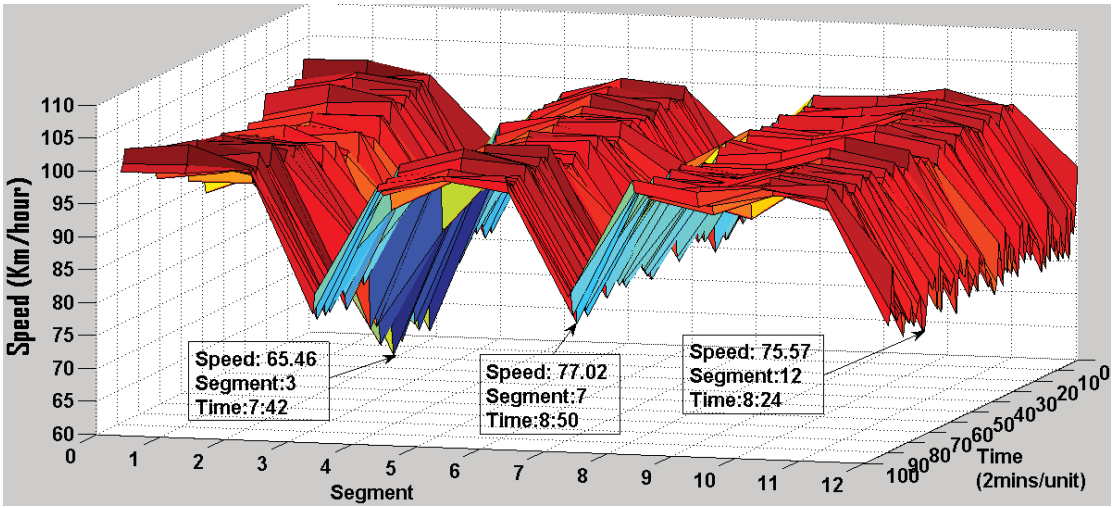


Figure 5.34 The speed profile under the control of DP (two phases of search)

The density profile in Fig. 5.33 ranges from 10veh/km/lane to about 50veh/km/lane. The bottleneck locations can be identified at the downstream of ramp 1 (segment 3), the downstream of ramp 2 (segment 7) and the downstream of ramp 3 (segment 12). The densities between the bottleneck locations were below the critical density (33veh/km/lane), so no motorway queue was propagated between on-ramps. Comparing to the density profile under the control of the first phase of search (Fig. 5.31), the simulation results in Fig. 5.33 seems less fluctuant along the bottleneck locations due to the finer control resolution. Density comparisons of bottleneck

locations between two experiments are shown in Fig. 5.35~Fig. 5.37.

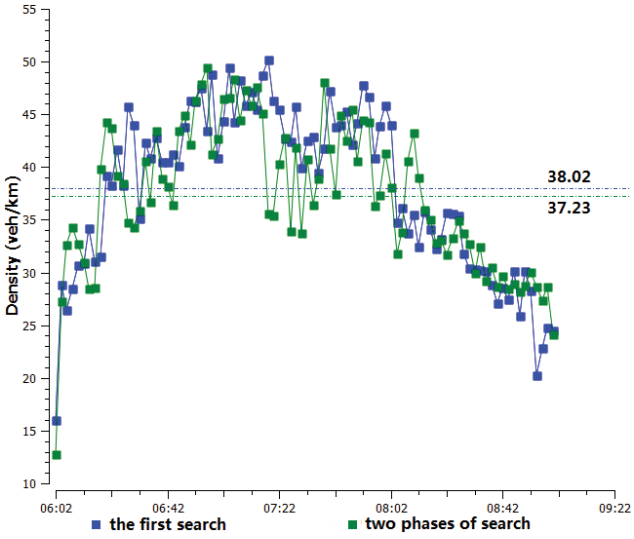


Figure 5.35 The density comparison of segment 3

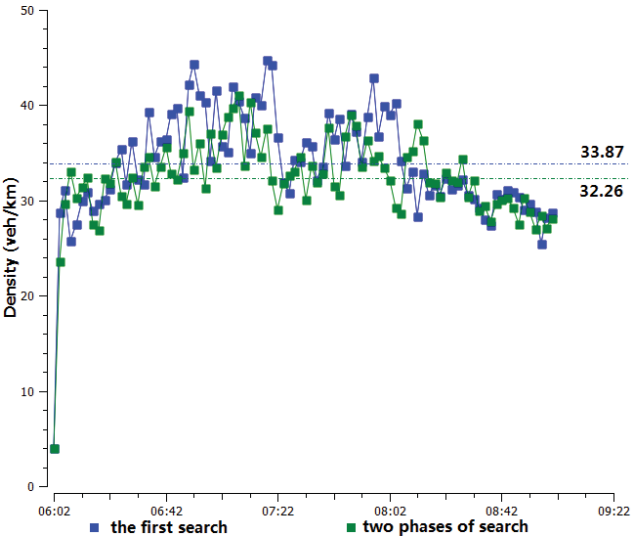


Figure 5.36 The density comparison of segment 7

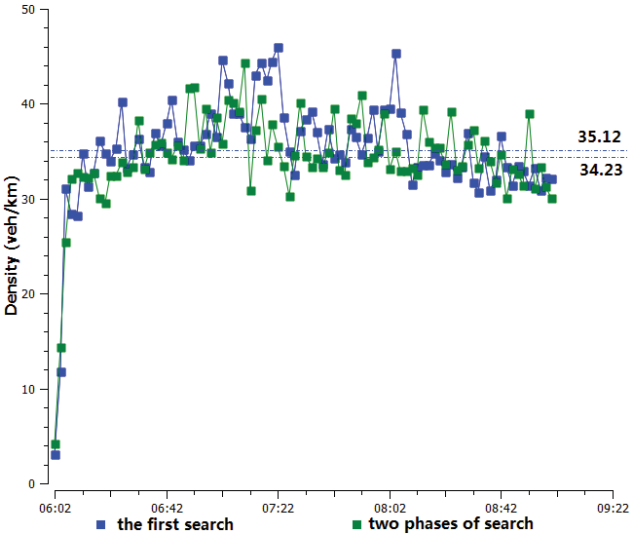


Figure 5.37 The density comparison of segment 12

The density comparisons show that the average densities of bottleneck locations under the control of two phases of search are lower than those under the first search, and that the most significant improvement can be found at segment 7 (ramp 2), where the average density under the control of two phases of search are below the critical density.

The further details can be revealed by the speed profile. In Fig. 5.34, the speed profile between on-ramps remains around the free-flow speed (Fig. 5.34), so no motorway queue was propagated between on-ramps during the simulation. One congestions can be identified at ramp 1 (segment 3), where speeds dropped to 65.46km/h at 7.42am. The speed profiles along ramp 2 (segment 7) and ramp3 (segment 12) remains above 70km/h, so traffic flow at ramp 2 and ramp 7 were undersaturated during the simulation.

5.6 Simulation Results under Medium Traffic Demands and Discussion

The proposed algorithms were also implemented under medium traffic demands.

Table 5.12 Medium traffic demands

Time	Origin	Greville Ramp(1)	Constellation Ramp(2)	Trisram Ramp(3)
6:00am	*3500*	*1000*	*1000*	*1000*
6:15am	*4000*	*1300*	*1200*	*1200*
6:30am	*4500*	*1400*	*1300*	*1400*
6:45am	*5000*	*1600*	*1300*	*1700*
7:00am	*5500*	*1600*	*1400*	*1700*
7:15am	*5500*	*1700*	*1500*	*1600*
7:30am	*5000*	*1700*	*1500*	*1500*
7:45am	*5000*	*1600*	*1400*	*1500*
8:00am	*4500*	*1500*	*1300*	*1100*
8:15am	*4000*	*1400*	*1300*	*1100*
8:30am	*3500*	*1400*	*1200*	*1000*
8:45am	*3000*	*1300*	*1100*	*1000*

Traffic demands in Table 5.12 were acquired by modifying Table 5.11 to study the performance differences between the proposed algorithms and the local responsive ramp metering approach under locally congested traffic conditions. By reducing traffic demand at origin and increasing traffic demands for ramp3 in Table 5.11, two local congestions were expected to be found at ramp1 and ramp3 when having ramp signals off, and no motorway queue was propagated on between on-ramps.

The density profile under no-ramp-control case is shown in Fig. 5.38.

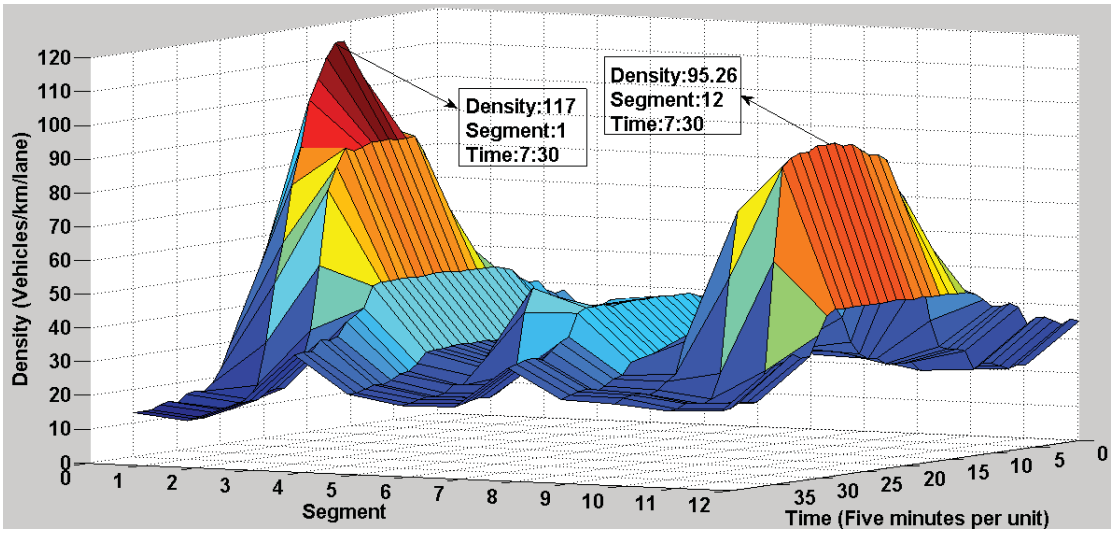


Figure 5.38 The density profile under medium traffic demands

In Fig. 5.38, two local congestions can be identified at the upstream of ramp1 and the downstream of ramp3, whose densities peaked at 117veh/km/lane and 95.26veh/km/lane, respectively. The density profile between on-ramps was below the critical density, so no motorway queue was propagated during the simulation.

The density profiles under the control of ALINEA and under the control of two phases of DP search are given in Fig. 5.39~Fig. 5.40. The simulation results show that both local and coordinated ramp control strategies were capable of eliminating the local congestion and that density profile along the bottleneck locations were a bit less fluctuant under the control of DP ramp metering than those under the control of ALINEA, but performance differences were very little.

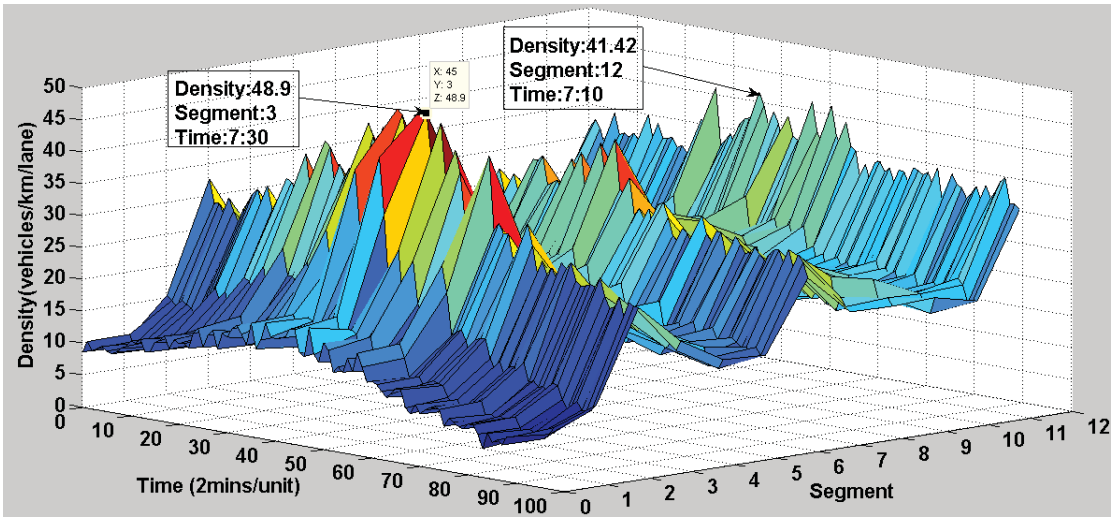


Figure 5.39 The density profile under the control of ALINEA (medium traffic demands)

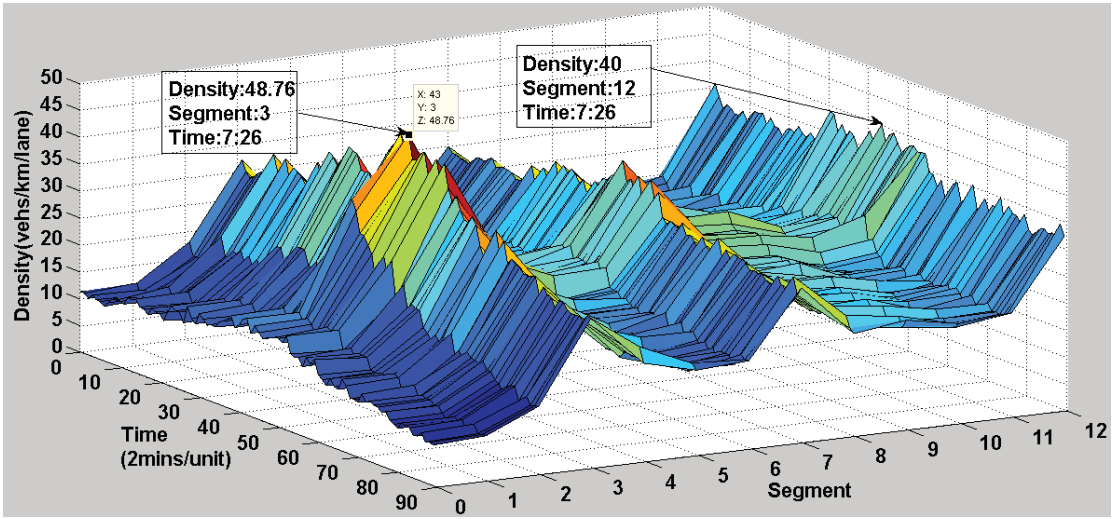


Figure 5.40 The density profile under the control of two phases of search (medium traffic demands)

5.7 TTS Comparisons

TTS values, as the most important system MOE were compared in Table 5.13~5.14, and ramp MOEs were also given to evaluate the performances of the proposed ramp metering algorithms at each on-ramp. Measures of effectiveness are defined as follows:

Ramp MOEs

1) Total Ramp Travel Time (hours): Total time experienced by all vehicle travelling on the ramp section per kilometre.

2) Average Ramp Delay (seconds per vehicle): Average delay time per vehicle while travelling on the ramp section.

3) Average Flow Rate (vehicles per hour): The number of vehicles travelling on the ramp section during the simulation (one hour).

System MOEs

4) Total Time Spent (hours): Total travel time accumulated by all vehicles travelling in the Aimsun traffic network.

MOEs under high traffic demands are given in Table 5.13.

Table 5.13 Comparisons of MOEs under high traffic demands

		MOEs	No CONTROL	ALINEA	DP1 (the first)	DP2 (two phases)	Percentage Reductions (%)		
							ALINEA	DP1	DP2
Ramp MOEs	1	Total Travel Time (hour)	*363.03*	*297.15*	*255.42*	*253.21*	*-18.15%*	*-29.64%*	*-30.25%*
		Average Ramp Delay (sec/veh)	*524.82*	*443.25*	*368.28*	*341.53*	*-15.54%*	*-29.83%*	*-34.92%*
		Average Flow Rate (veh/h)	*713*	*792*	*880.*	*890*	*11.08%*	*23.42%*	*24.82%*
	2	Total Travel Time (hour)	*571.32*	*529.61*	*506.32*	*498.12*	*-7.30%*	*-11.38%*	*-12.81%*
		Average Ramp Delay (sec/veh)	*987.22*	*794.82*	*763.86*	*735.56*	*-19.49%*	*-22.63%*	*-25.49%*
		Average Flow Rate (veh/h)	*648*	*692*	*722*	*770*	*6.79%*	*11.42%*	*18.83%*
	3	Total Travel Time (hour)	*173.21*	*176.67*	*177.08*	*174.31*	*2.00%*	*2.23%*	*0.64%*
		Average Ramp Delay (sec/veh)	*265.35*	*273.60*	*277.70*	*267.98*	*3.11%*	*4.65%*	*0.99%*
		Average Flow Rate (veh/h)	*892*	*882*	*866*	*904*	*-1.12%*	*-2.91%*	*1.35%*
System MOE		**Total Time Spent (hour)**	***2701.53***	***2541.31***	***2443.74***	***2358.21***	***-5.93%***	***-9.21%***	***-11.38%***

MOEs under medium traffic demands are given in Table 5.13.

Table 5.14 Comparisons of MOEs under medium traffic demands

		MOEs	NO CONTROL	ALINEA	DP1 (the first)	DP2 (two phases)	Percentage Reductions (%)		
							ALINEA	DP1	DP2
Ramp MOEs	1	Total Travel Time (hour)	*307.03*	*260.63*	*254.73*	*253.38*	*-15.11%*	*-17.03%*	*-17.47%*
		Average Ramp Delay (sec/veh)	*484.82*	*385.67*	*369.67*	*370.62*	*-20.45%*	*-23.75%*	*-23.56%*
		Average Flow Rate (veh/h)	*763*	*852*	*876*	*880*	*11.66%*	*14.81%*	*15.33%*
	2	Total Travel Time (hour)	*518.12*	*494.35*	*497.35*	*496.92*	*-4.59%*	*-4.01%*	*-4.09%*
		Average Ramp Delay (sec/veh)	*779.38*	*713.93*	*733.13*	*732.09*	*-8.40%*	*-5.93%*	*-6.07%*
		Average Flow Rate (veh/h)	*706*	*786*	*780*	*778*	*11.33%*	*10.48%*	*10.20%*
	3	Total Travel Time (hour)	*196.21*	*174.30*	*175.31*	*175.40*	*-11.17%*	*-10.65%*	*-10.61%*
		Average Ramp Delay (sec/veh)	*320.35*	*268.64*	*269.64*	*270.10*	*-16.14%*	*-15.83%*	*-15.69%*
		Average Flow Rate (veh/h)	*794*	*874*	*870*	*868*	*10.08%*	*9.57%*	*9.32%*
System MOE		**Total Travel Spent (hour)**	***2547.09***	***2347.52***	***2349.37***	***2346.18***	***-7.84%***	***-7.76%***	***-7.89%***

MOEs in Table 5.13~5.14 indicate that the proposed algorithm outperformed ALINEA under high traffic demands, under which coordinated ramp signals can prevent the motorway queues between on-ramps and isolated the motorway congestions, and had about 5% (183.1h) improvement on TTS values over ALINEA. However, under medium traffic demands where no motorway queue was propagated between on-ramps, DP ramp metering did not have any advantages over ALINEA.

Unlike the results from macroscopic simulation, two phases of search DP made obvious further improvement, about 3% (85.53h) over the first search of DP, on TTS values under high traffic demands, which is because, in the microscopic simulator, two phases of search DP with more released metering rates can more effectively control the densities at the bottleneck locations when traffic demands were very high and less fluctuant densities made the model prediction more accurate. Under medium traffic demands, their performances on TTS values had very little difference.

5.8 Summary

This chapter demonstrated the implementation of the proposed algorithms in a stochastic environment, where the study location in Chapter 4 had been reconstructed as a new simulation scenario. An API module was provided by the microscopic simulator to interface the ramp metering algorithms. The macroscopic model employed by DP ramp metering was coded in the API module and calibrated with simulation results of the simulation scenario under uncongested conditions to replicate the traffic behavior of the study location in Aimsun.

Two experiments were conducted to implement the DP based ramp metering algorithms under different traffic demands and compare their performances to ALINEA and no-ramp-control case in terms of TTS values. The simulation results from the first experiment indicated that DP based ramp metering outperformed the local responsive ramp metering when traffic demands were high enough to cause the motorway queues between on-ramps in no-ramp-control situation. TTS values had been further improved about 5% (183.1h) due to the reason that coordinated ramp signals was able to isolate the local congestion but local responsive ramp metering failed to do so. The second experiment was conducted to demonstrate the performances of the proposed algorithms under medium traffic demands that were not high enough to propagate the motorway queues but enough to cause the local congestions at bottleneck locations. DP based ramp metering had no advantages over the local responsive ramp metering in this situation. The simulation results also indicated that the second search of DP made obvious improvements on TTS values over the first search of DP under high traffic demands due to reason that the downstream densities of bottleneck locations can be more effectively controlled by more released metering rates.

CHAPTER 6 CONCLUSIONS AND RECOMMENDATIONS

6.1 A Summary of the Thesis Research

A critical and extensive literature review was conducted to study the development and implementation of macroscopic traffic models in the traffic signal control and optimization. From fundamental diagrams, to the first-order continuum models, to the second-order models, macroscopic traffic models were employed in a wide range of traffic engineering tasks, from local responsive ramp metering, to signal plans of interchanges, to networked ramp metering control. Model-based coordinated ramp metering strategies were developed based on discreted second-order models, which formulated the optimal ramp control problem as a nonlinear constrained minimization problem. The existing model-based ramp metering strategies tended to release continuous metering rates and their performances were evaluated in macroscopic traffic simulations by the employed second-order models themselves.

This research developed model-based coordinated ramp metering strategies by using DP algorithm. DP decision networks were proposed, where a traffic system can be modeled as a number of discrete traffic states and separated by time stages. The state transitions between two neighboring stages are determined by feasible actions (ramp metering rates). Optimal metering rates can be obtained by searching the optimal trajectory of actions (or decisions), along which TTS values returned from each stage are minimized. The search procedure, by following the concept of IDP (Iterative Dynamic Programming), was broke into two phases to reduce the computational load. The first search was conducted by a simplified DP decision network to find a near-optimal trajectory, where each ramp meter can release 9 discrete metering rates, and the second search was performed by a generic DP decision network to refine the trajectory found from the first search, where each ramp meter can release 37 discrete metering rates. The second search was optional.

The proposed algorithms were evaluated in macroscopic traffic simulations, where the prediction model and the simulation environment were the same and no prediction

errors were considered. Experiments were conducted under same traffic demands but different queue constrains. Conclusions and contributions based on simulation results from macroscopic simulations can be summarized as follows:

1. The DP ramp metering was able to improve traffic condition under high traffic demands and managed ramp queue lengths under large ramp queue constrains but failed to eliminate motorway queues so when queue constrains were overstrict.
2. The DP ramp metering was capable of isolating the local congestions from different bottleneck locations when traffic demands were very high and preventing the propagation of motorway queues between on-ramps, while the local responsive ramp metering failed to do so.
3. The proposed simplified DP decision network converged very fast to find the optimal trajectory due to the fact that the second search of DP made no further improvement on the first search of DP, and their performances on TTS values had very little difference. In other words, 9 released metering rates for each ramp meter seems adequate to manage ramp queues and prevent the motorway congestions. This feature significantly reduces computational load and makes DP ramp metering possible to be online control approaches

In order to understand the feasibility of DP ramp metering in the real world, the proposed algorithms were evaluated in a stochastic environment by a microscopic traffic simulator, Aimsun. Experiments were conducted under different traffic demands without queue constrains. Conclusions and contributions based on the implementation procedures and simulation results can be summarized as follows:

4. In order to acquire the initial traffic conditions, traffic measurements should be collected at every control step from detectors along the mainline motorway. Distributions of the detectors can be arranged according to how to divide the motorway stretch into road segments.
5. In order to replicate traffic dynamics in Aimsun, the prediction model of DP ramp metering have to be calibrated. The model can be calibrated by the simulation results that are acquired when the mainline traffic under uncongested conditions, since the mainline traffic conditions should remain uncongested as long as DP

ramp metering does not fail.

6. DP ramp metering outperformed the local responsive ramp metering when traffic demands were high enough to cause the motorway queues between on-ramps, but had no advantage over the local responsive ramp metering under medium traffic demands that caused no motorway queue between on-ramps but only local congestions at bottleneck locations.
7. Unlike the results from macroscopic simulation, the second search of DP made obvious further improvement (about 3% over the first search of DP) on TTS values under high traffic demands, which is because two phases of search DP with more released metering rates can more effectively control the densities at the bottleneck locations when traffic demands were very high and less fluctuant densities made the model prediction more accurate. Therefore, the second search is still necessary for the implementation in the real world.

6.2 Recommendations

Upon the completion of the study, the limitations of current research can be identified to suggest directions of future research.

1. All experiments conducted in this research assumed perfect knowledge of traffic demands, so a method for short-term forecasting of detector counts (Pohlmann & Friedrich, 2013) can be integrated to the experiments of DP ramp metering.
2. The major problem of DP is that it has difficulties to handle the high-dimensional problems, which is also known as “the curse of dimensionality”. For large scale motorway networks where a large number of on-ramps are included, the proposed DP decision networks are incapable to handle all on-ramps together due to high computational load not suitable for adaptive online control approaches. It is suggested that the entire motorway network is divided into a number of overlapped motorway stretches consisting of 3~N on-ramps. DP ramp metering is implemented for each motorway stretch individually. Ramp meters located at overlapped road segment may have two optimal metering rates at one control step,

and stricter one is selected. The value of N could be deduced from the longest motorway queues appearing on the motorway. An example is shown in Fig. 6.1.

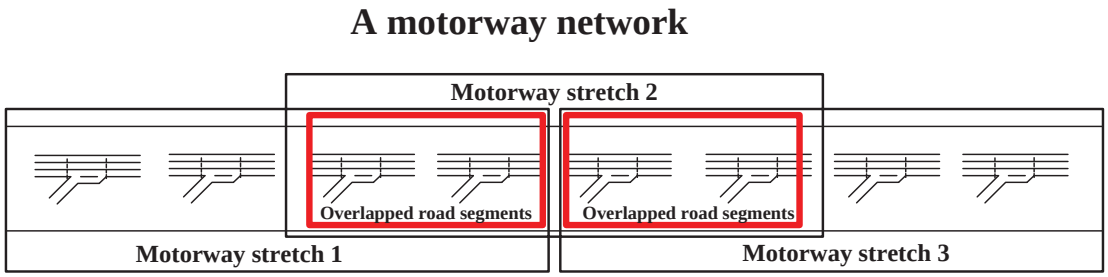


Figure 6.1 The presumed implementation structure for large scale motorway networks

In Fig. 6.1, N=4 and each motorway stretch consists of 4 on-ramps. Ramp meters on the overlapped road segments could have two optimal metering rates at one control step, and only stricter one is selected. By implementing DP ramp metering as showing in 6.1, the proposed algorithms could handle a motorway network where a large number of on-ramps are involved. Better implementation structures need to be identified by further study.

BIBLIOGRAPHY

Abdelfatah, A., Ramadan, A., & Darwish, G. (2012). Utilizing Vissim to Evaluate Ramp Metering Performance. *International Journal of Traffic and Transportation Engineering*, 1-6.

Ardekani, S., Ghandehari, M., & Nepal, S. (2011). Macroscopic speed-flow models for characterization of freeway and managed lanes. *Institutul Politehnic din Iasi. Buletinul. Sectia Constructii. Arhitectura, 57*(1).

Banks, J. H. (1989). Freeway speed-flow-concentration relationships: more evidence and interpretations (with discussion and closure). *Transportation Research Record*(1225).

Banks, J. H. (1990). Flow processes at a freeway bottleneck. *Transportation Research Record*(1287).

Banks, J. H. (1991). The two-capacity phenomenon: some theoretical issues. *Transportation Research Record*(1320).

Barceló, J., & Casas, J. (2005). Dynamic network simulation with AIMSUN *Simulation Approaches in Transportation Analysis* (pp. 57-98): Springer.

Baskar, L. D., Schutter, B. D., & Hellendoorn, H. (2012). Traffic Management for Automated Highway Systems Using Model-Based Predictive Control. *IEEE Transactions on Intelligent Transportation Systems*

838-847.

Bellemans, T. (2003). Traffic Control on Motorways. Leuven, Belgium: Ph.D. dissertation, Katholieke Univ.

Bellman, R. E., & Dreyfus, S. E. (1966). *Applied dynamic programming* (Vol. 7962): Princeton University Press.

Bogenberger, K. (2000). Adaptive Fuzzy Systems for Traffic Responsive and Coordinated Ramp Metering. Ph.D. Thesis at the Fachgebiet.

Bogenberger, K., & Keller, H. (2001). *An Evolutionary Fuzzy System for Coordinated and Traffic Responsive Ramp Metering.* Paper presented at the Hawaii International Conference on Systems Sciences (34th : 2001 : Maui, Hawaii). Proceedings.

Bogenberger, K., & May, A. D. (1999). Advanced Coordinated Traffic Responsive Ramp Metering Strategies: University of California, Berkeley.

Bogenberger, K., Vukanovic, S., & Keller, H. (2001). *A Neuro-Fuzzy Algorithm for Coordinated Traffic Responsive Ramp Metering.* Paper presented at the IEEE 4th International Conference on Intelligent Transportation Systems, California.

Brackstone, M., & McDonald, M. (1999). Car-following: a historical review. *Transportation Research Part F: Traffic Psychology and Behaviour, 2*(4), 181-196.

Ceder, A., & May, A. (1976). Further evaluation of single-and two-regime traffic flow models. *Transportation Research Record*(567), 1-15.

Chandler, R. E., Herman, R., & Montroll, E. W. (1958). Traffic dynamics: studies in

car following. *Operations research, 6*(2), 165-184.

Chaudhary, N. A., Tian, Z., Messer, C. J., & Chu, C.-L. (2004). Ramp metering algorithms and approaches for Texas: Texas Transportation Institute, Texas A & M University System.

Chin, H. C., & May, A. D. (1991). Examination of the speed-flow relationship at the Caldecott Tunnel. *Transportation Research Record* (1320).

Chu, L. Y., Liu, H. X., & Recker, W. (2001). Paramics API development document for actuated signal, signal coordination and ramp control. *PATH Working Paper*, HCB-ITS-PWP, 37 pp.

Chu, L. Y., Liu, H. X., Recker, W., & Zhang, H. M. (2004). Performance Evaluation of Adaptive Ramp-MeteringAlgorithms Using Microscopic Traffic Simulation Model. *Journal of Transportation Engineering,American Society of Civil Engineering (ACSE)*, 330-338.

Daganzo, C. F. (1997). *Fundamentals of transportation and traffic operations*. Pergamon: Oxford.

Drake, J. S., Schofer, J. L., & May, A. D. (1967). *A Statistical Analysis of Speed Density Hypotheses.* Paper presented at the Proceedings of the Third International Symposium on the Theory of Traffic Flow, New York.

Drew, D. R. (1968). Traffic flow theory and control.

Easa, S. (1982). Selecting two-regime traffic-flow models. *Transportation Research Record*(869).

Easa, S., & May, A. (1980). Generalized Procedure for Estimating Single-and Two-Regime Traffic-Flow Models. *Transportation Research Record*(772).

Edie, L. C. (1961). Car-following and steady-state theory for noncongested traffic. *Operations research, 9*(1), 66-76.

Fang, F. C., & Elefteriadou, L. (2006). Development of an Optimization Methodology for Adaptive Traffic Signal Control at Diamond Interchanges. *Journal of Transportation Engineering, American Society of Civil Engineering (ACSE)*, 132(138),629-638.

Fang, F. C., & Elefteriadou, L. (2008). Capability-Enhanced Microscopic Simulation With Real-Time Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems*

625-632.

Fehon, K., & Klim, T. (2010). Modelling Active Traffic Management with Paramics. *IEEE Transactions on Intelligent Transportation Systems*

14-18.

Ferrari, P. (1989). The effect of driver behaviour on motorway reliability. *Transportation Research Part B: Methodological, 23*(2), 139-150.

Fritzsche, H. (1994). A MODEL FOR TRAFFIC SIMULATION. *Traffic Engineering and control, 35*(5).

Gartner, N., Messer, C. J., & Rathi, A. K. (2005). Revised monograph on traffic flow theory: a state-of-the-art report. *Special Report by the Transportation Research Board of the National Research Council*.

Gazis, D. C., Herman, R., & Rothery, R. W. (1961). Nonlinear follow-the-leader

models of traffic flow. *Operations research, 9*(4), 545-567.

Gerlough, D. L., & Huber, M. J. (1976). Traffic flow theory: a monograph. Transportation Research Board.

Geroliminis, N., Srivastava, A., & Michalopoulos, P. (2011). A Dynamic-Zone-Based Coordinated Ramp-Metering Algorithm With Queue Constraints for Minnesota's Freeways. *IEEE Transactions on Intelligent Transportation Systems*

1576-1586.

Ghods, A. H., Fu, L., & Rahimi-Kian, A. (2010). An Efficient Optimization Approach to Real-Time Coordinated and Integrated Freeway Traffic Control. *IEEE Transactions on Intelligent Transportation Systems*

873-884.

Ghods, A. H., Kian, A. R., & Tabibi, M. (2009). Adaptive freeway ramp metering and variable speed limit control: a genetic-fuzzy approach. *Intelligent Transportation Systems Magazine, IEEE*, 27-36.

Gipps, P. G. (1981). A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological, 15*(2), 105-111.

Greenberg, H. (1959). An analysis of traffic flow. *Operations research, 7*(1), 79-85.

Greenshields, B. D. (1935). A Study of Traffic Capacity. *HRB Proceedings,* 448-477.

Haberman, R. (1977). *Mathematical models*: SIAM.

Hadj-Salem, H., Blosseville, J., & Papageorgiou, M. (1990). *ALINEA: a local feedback control law for on-ramp metering; a real-life study.* Paper presented at the Road Traffic Control, 1990., Third International Conference on.

Hall, F. L., Allen, B. L., & Gunter, M. A. (1986). Empirical analysis of freeway flow-density relationships. *Transportation Research Part A: General, 20*(3), 197-210.

Hall, F. L., & Hall, L. M. (1990). Capacity and speed-flow analysis of the Queen Elizabeth Way in Ontario. *Transportation Research Record*(1287).

Hall, F. L., Hurdle, V., & Banks, J. H. (1992). Synthesis of recent work on the nature of speed-flow and flow-occupancy (or density) relationships on freeways. *Transportation Research Record*(1365).

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

Hillier, F. S. (2001). Lieberman. Introduction to operation research: McGraw-Hill New York.

Hou, Z., Xu, X., Yan, J., Xu, J.-X., & Xiong, G. (2011). A Complementary Modularized Ramp Metering Approach Based on Iterative Learning Control and ALINEA. *IEEE Transactions on Intelligent Transportation Systems*

1305-1318

Hou, Z., Yan, J., Xu, J.-X., & Li, Z. (2012). Modified Iterative-Learning-Control-Based Ramp Metering Strategies for Freeway Traffic Control With Iteration-Dependent Factors. *IEEE Transactions on Intelligent Transportation Systems*

606-618.

Kometani, E., & Sasaki, T. (1959). A safety index for traffic with linear spacing. *Operations research, 7*(6), 704-720.

Koshi, M., Iwasaki, M., & Ohkura, I. (1983). *Some findings and an overview on vehicular flow characteristics.* Paper presented at the Proceedings of the 8th International Symposium on Transportation and Traffic Flow Theory.

Kotsialos, A., & Papageorgiou, M. (2004). Nonlinear Optimal Control Applied to Coordinated Ramp Metering. *IEEE Trans.on Control Systems Technology*, 920-933.

Kotsialos, A., Papageorgiou, M., Diakaki, Y. P., & Middleham, F. (2002). Traffic Flow Modelling of Large-scale Motorway Networks Using the Macroscopic Modelling Tool METANET. *IEEE Transactions on Intelligent Transportation Systems*, 282-292.

Kotsialos, A., Papageorgiou, M., Hayden, J., Higginson, R., McCabe, K., & Rayman, N. (2006). Discrete release rate impact on ramp metering. *IEE Proc. Intell. Transp. Syst., 153*.

Kouvaritakis, B., Cannon, M., & Rossiter, J. A. (1999). Non-linear Model Based Predictive Control. *International Journal of Control*, 72(10):919-928.

Kuhne, R. D. (2008). *Foundations of Traffic Flow Theory I: Greenshields' Legacy– Highway Traffic.* Paper presented at the Symposium on the Fundamental Diagram: 75 Years (Greenshields 75 Symposium), Transportation Research Board.

LeVeque, R. (1992). *Numerical methods for conservation laws*: Lectures in Mathematics ETHZurich.

Lighthill, M. J., & Whitham, G. B. (1955). On kinematic waves. II. A theory of traffic flow on long crowded roads. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences, 229*(1178), 317-345.

Liu, G., Lyrintzis, A. S., & Michalopoulos, P. G. (1998). Improved high-order model for freeway traffic flow. *Transportation Research Record: Journal of the Transportation Research Board, 1644*(1), 37-46.

Luus, R. (2002). *Iterative Dynamic Programming*: Chapman and Hall, CRC.

May, A. D. (2000). *Traffic Flow Fundamentals*: Prentice Hall, Englewood Cliffs.

May, A. D., & Keller, H. E. (1967). Non-integer car-following models. *Highway Research Record*, 19-32.

May, A. D., & Keller, H. E. (1968). *Evaluation of single-and two-regime traffic flow models*. Paper presented at the Fourth International Symposium on the Theory of Traffic Flow Proceedings, Karlsruhe, West Germany.

Messme, A., & Papageorgiou, M. (1981). Parameter Identification for a Traffic Flow Model. *Automatica*, 837-843.

Messmer, A. (2000). METANET- A Simulation Program for Motorway Networks: Technical University of Crete, Dynamic System and Simulation Laboratory.

Messmer, A., & Papageorgiou, M. (1990). METANET: A Macroscopic Simulation Program for Motorway Networks. *Traffic Engineering and control*, 466-470.

Michaels, R. (1963). *Perceptual factors in car following.* Paper presented at the Proceedings of the 2nd International Symposium on the Theory of Road

Traffic Flow (London, England), OECD.
Michalopoulos, P. G., Beskos, D. E., & Lin, J.-K. (1984a). Analysis of interrupted traffic flow by finite difference methods. *Transportation Research Part B: Methodological, 18*(4), 409-421.
Michalopoulos, P. G., Beskos, D. E., & Lin, J.-K. (1984b). Multilane traffic flow dynamics: some macroscopic considerations. *Transportation Research Part B: Methodological, 18*(4), 377-395.
Michalopoulos, P. G., & Stephanopoulos, G. (1979). Modelling and analysis of traffic queue dynamics at signalized intersections. *Transportation Research Part A: General, 13*(5), 295-307.
Michalopoulos, P. G., & Stephanopoulos, G. (1981). An application of shock wave theory to traffic signal control. *Transportation Research Part B: Methodological, 15*(1), 35-51.
Moridpour, S., Sarvi, M., & Rose, G. (2010). Lane changing models: A critical review. *Transportation letters, 2*(3), 157.
Olstam, J. J., & Tapani, A. (2004). *Comparison of Car-following models*: Swedish National Road and Transport Research Institute.
Papageogiou, M., Hadj-Salem, H., & Blosseville, J. M. (1991). ALINEA:A Local Feedback Control Law for On-ramp Metering. *Tran. Research Record 1320*, 58-64.
Papageorgiou, M. (1983). *Applications of automatic control concepts to traffic flow modeling and control* (Vol. 50): Springer Verlag.
Papageorgiou, M. (1998). Some remarks on macroscopic traffic flow modelling. *Transportation Research Part A: Policy and Practice, 32*(5), 323-329.
Papageorgiou, M., Blosseville, J., & Hadj-Salem, H. (1989). Macroscopic Modelling of Traffic Flow on the Boulevard Peripherique in Paris. *Trans.Res.Board, Vol. 23B*, 29-47.
Papageorgiou, M., & Kotsialos, A. (2002). Freeway ramp metering: an overview. *Intelligent Transportation Systems, IEEE Transactions on, 3*(4), 271-281.
Papageorgiou, M., & Marinaki, M. (1995). A Feasible Direction Algorithm for the Numerical Solution of Optimal Control Problems: Dynamics Syst. Simulation Lab.,Tech. Univ. Crete, Chania, Greece.
Papageorgiou, M., & Papamichail, I. (2007). Handbook of ramp metering *Document for EUropean Ramp Metering Project (EURAMP)*.
Papamichail, I., Kotsialos, A., Margonis, I., & Papageorgiou, M. (2010). Coordinated ramp metering for freeway networks–A model-predictive hierarchical control approach. *Transportation Research Part C: Emerging Technologies, 18*(3), 311-331.
Papamichail, I., & Papageorgiou, M. (2008). Traffic-Responsive Linked Ramp-Metering Control. *IEEE Transactions on Intelligent Transportation Systems*
111-121.
PARAMICS. (2000). PARAMICS V3. 0 User Guide *Quadstone Ltd., UK*.
Payne, H. J. (1971). Models of freeway traffic and control. *Proc.Math. Models Public*

Syst., Simul. Council, 51-61.

Pipes, L. A. (1967). Car following models and the fundamental diagram of road traffic. *Transportation Research, 1*(1), 21-29.

Pohlmann, T., & Friedrich, B. (2013). A combined method to forecast and estimate traffic demand in urban networks. *Transportation Research Part C: Emerging Technologies, 31*, 131-144.

Quandt, R. E. (1958). The estimation of the parameters of a linear regression system obeying two separate regimes. *Journal of the american statistical association, 53*(284), 873-880.

Quandt, R. E. (1960). Tests of the hypothesis that a linear regression system obeys two separate regimes. *Journal of the american statistical association, 55*(290), 324-330.

Richards, P. I. (1956). Shock waves on the highway. *Operations research, 4*(1), 42-51.

Sun, X. (2005). Modeling, estimation, and control of freeway traffic. Ph. D. Thesis: University of California, Berkeley.

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

Taylor, C., Meldrum, D., & Jacobson, L. (1998). Fuzzy ramp metering: Design overview and simulation results. *Transportation Research Record: Journal of the Transportation Research Board, 1634*(1), 10-18.

Treiterer, J., & Myers, J. A. (1974). *The hysteresis phenomenon in traffic flow.* Paper presented at the Transportation and Traffic Theory, Proceedings.

TSS. (2008). Microsimulator and Mesosimulator in Aimsun 6 User's Manual. Barcelona, Spain.

TSS. (2010). Aimsun 6.1 Microsimulator API Manual. Barcelona, Spain.

Underwood, R. T. (1961). Speed, Volume, and Density Relationships: Quality and Theory of Traffic Flow. *Yale Bureau of Highway Traffic, New Haven, CT, USA*, 141-188.

VISSIM. (2000). VISSIM User's Manual, Version 3.5. *PTV planning transport verkehr AG*.

Whitham, G. (1974). *Linear and nonlinear waves*: Wiley (New York).

Wiedemann, R., & Reiter, U. (1992). Microscopic traffic simulation: the simulation system MISSION, background and actual state *Project ICARUS (V1052) Final Report. Brussels, CEC* (Vol. 2).

Yang, J.-S. (2002). *A Case Study of Highway Trafic Flow Model - Model Validation and Simulation.* Paper presented at the Proceedings of the 2002 IEEE International Conference on Control Applications, Glasgow, Scotland, U.K.

Yang, Q., & Koutsopoulos, H. N. (1996). A microscopic traffic simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies, 4*(3), 113-129.

Yu, X., Xu, W., Alam, F., Potgieter, J., & Fang, C. (2012a). *Genetic fuzzy logic*

approach to local ramp metering control using microscopic traffic simulation. Paper presented at the Mechatronics and Machine Vision in Practice (M2VIP), 2012 19th International Conference.

Yu, X., Xu, W., Alam, F., Potgieter, J., & Fang, C. (2012b). *Optimal coordination of networked motorway ramp meters.* Paper presented at the Mechatronics and Machine Vision in Practice (M2VIP), 2012 19th International Conference.

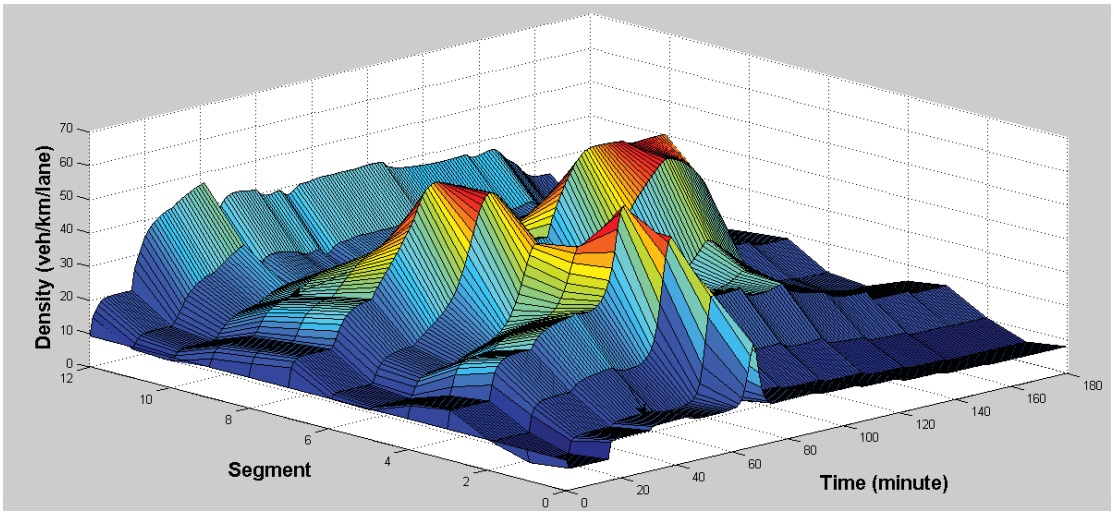
Zegeye, S. K., Schutter, B. D., Hellendoorn, J., Breunesse, E. A., & Hegyi, A. (2012). A Predictive Traffic Controller for Sustainable Mobility Using Parameterized Control Policies. *IEEE Transactions on Intelligent Transportation Systems* 1420-1429.

Zhang, H. M. (2002). A non-equilibrium traffic model devoid of gas-like behavior. *Transportation Research Part B: Methodological, 36*(3), 275-290.

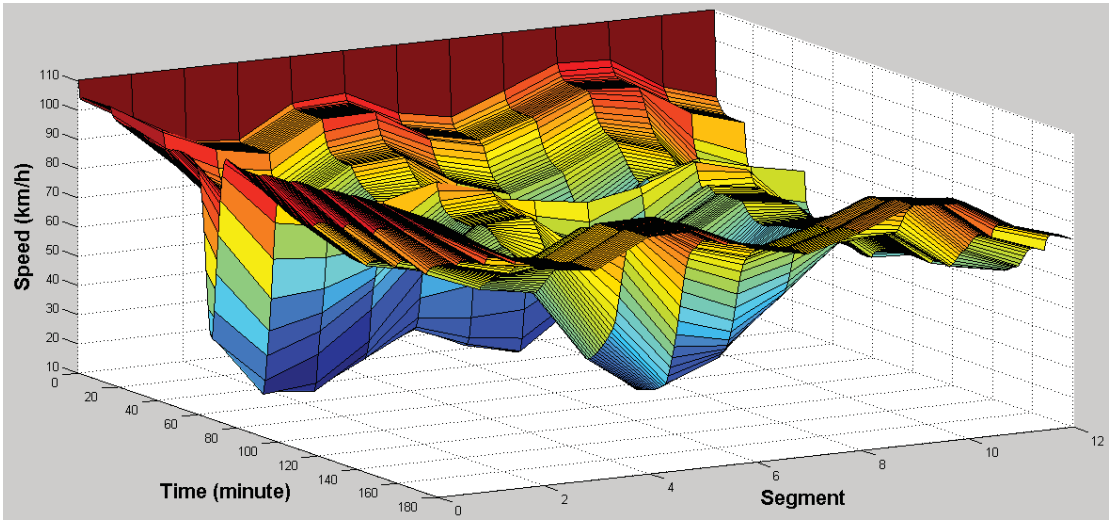
Zhao, D., Xu, J., & Yu, W. (2011). DHP Method for Ramp Metering of Freeway Traffic. *IEEE Transactions on Intelligent Transportation Systems* 990-999.

APPENDIX A: SIMULATION RESULTS FROM MACROSCOPIC TRAFFIC SIMULATION

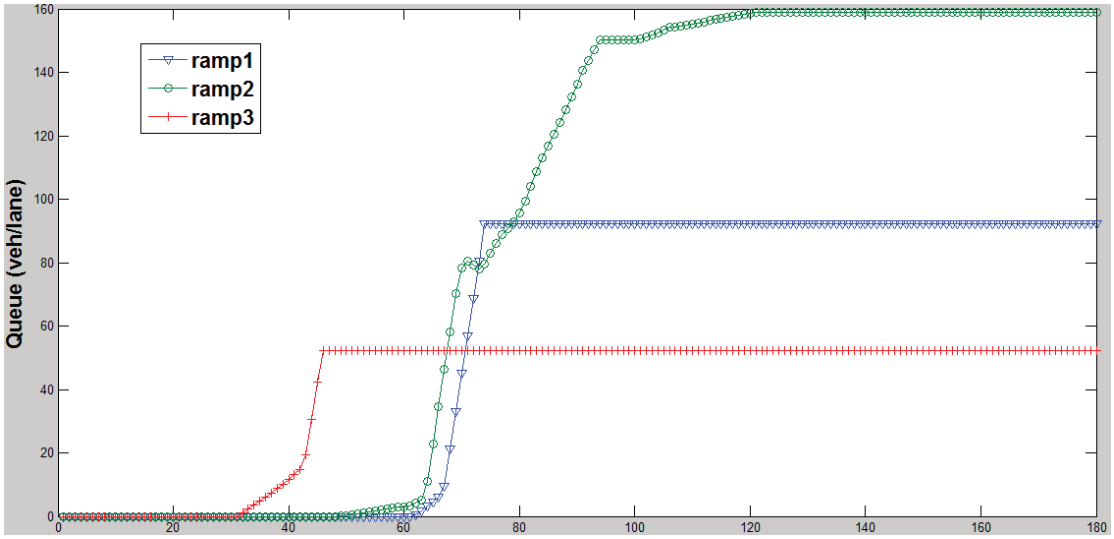
A.1. Simulation results of ALINEA/Q (Test2 ~ Test4)



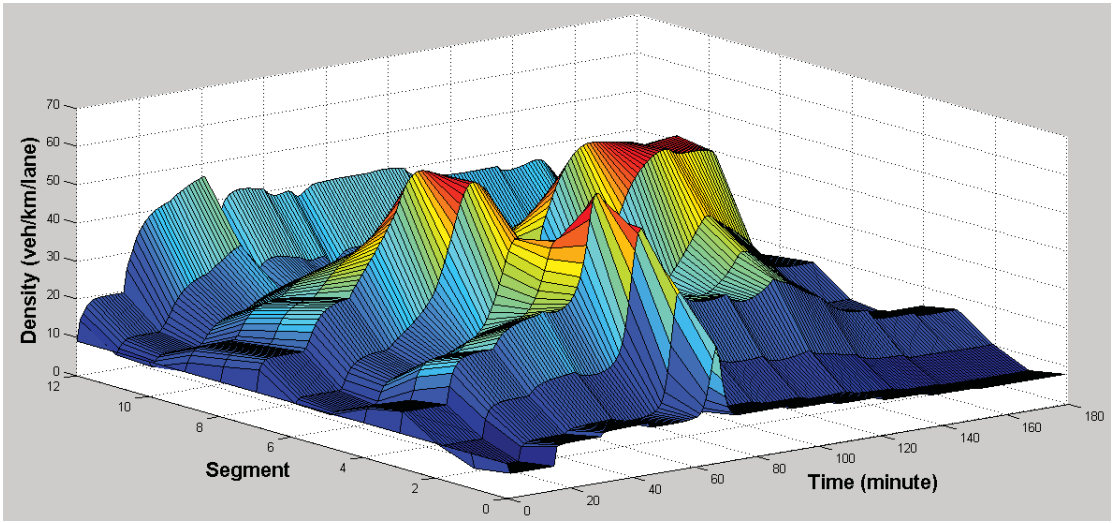
The density profile under the control of ALINEA/Q (Test 2)



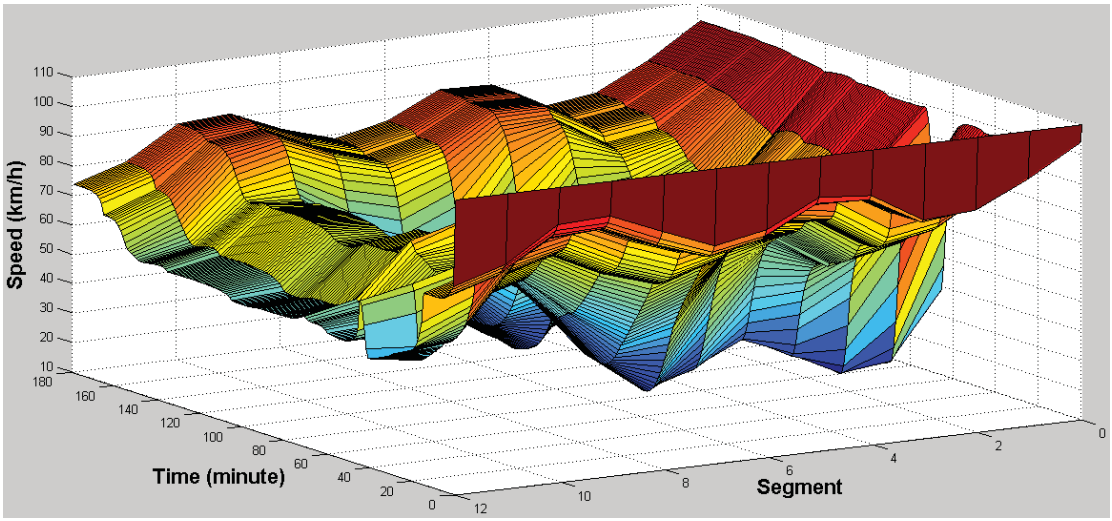
The speed profile under the control of ALINEA/Q (Test 2)



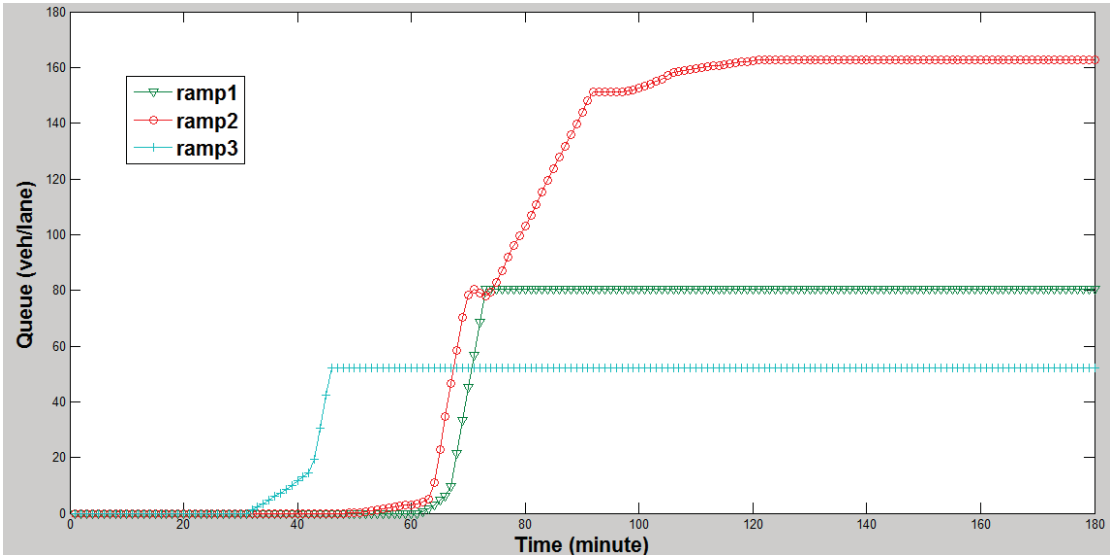
On-ramp queues under the control of ALINEA/Q (Test 2)



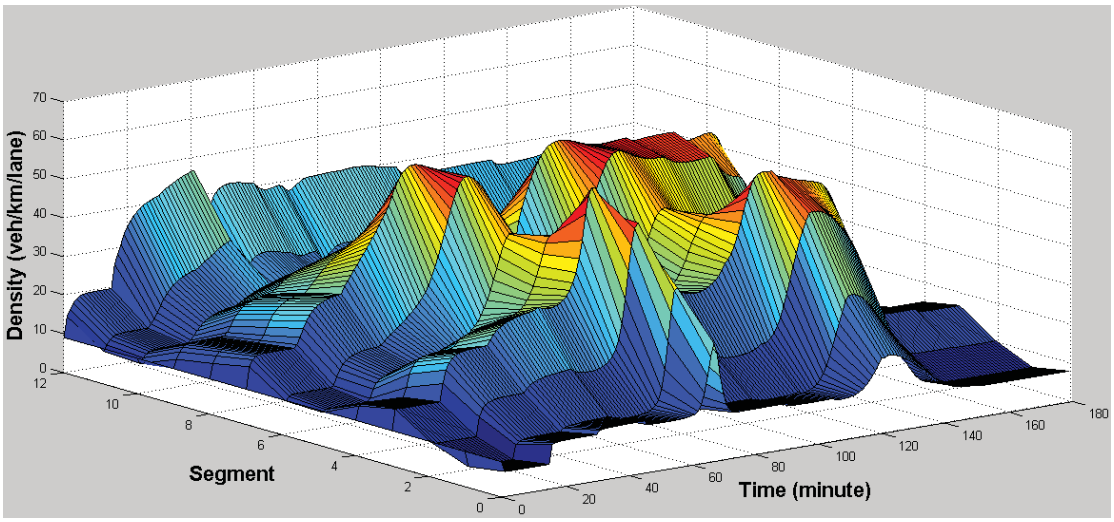
The density profile under the control of ALINEA/Q (Test 3)



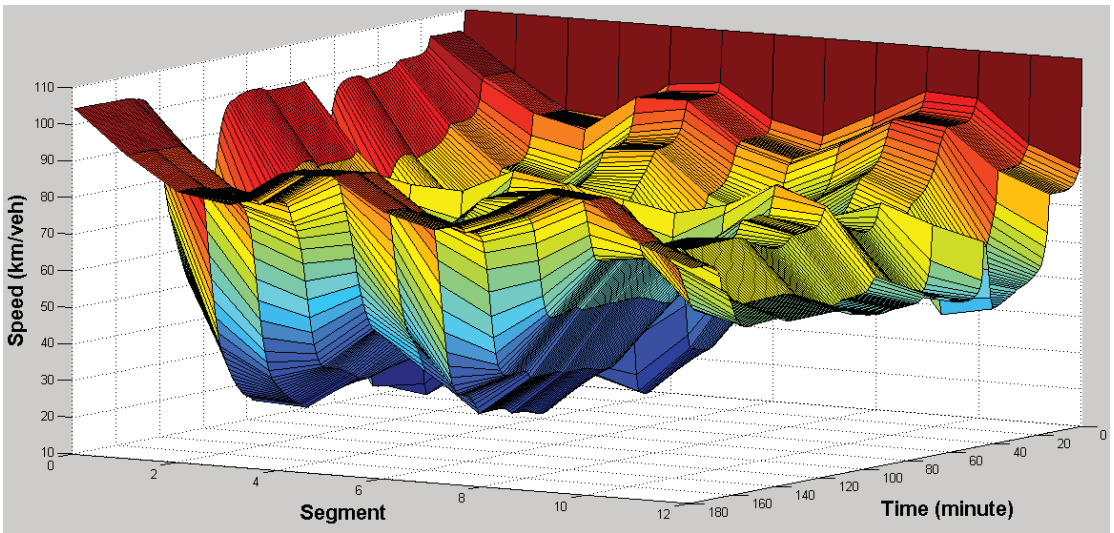
The speed profile under the control of ALINEA/Q (Test 3)



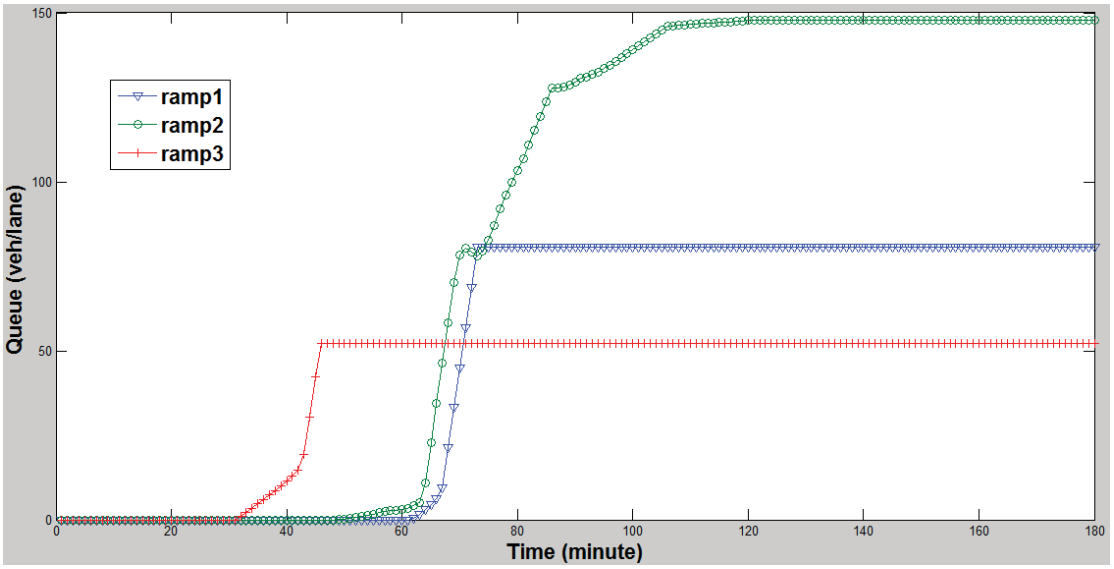
On-ramp queues under the control of ALINEA/Q (Test 3)



The density profile under the control of ALINEA/Q (Test 4)

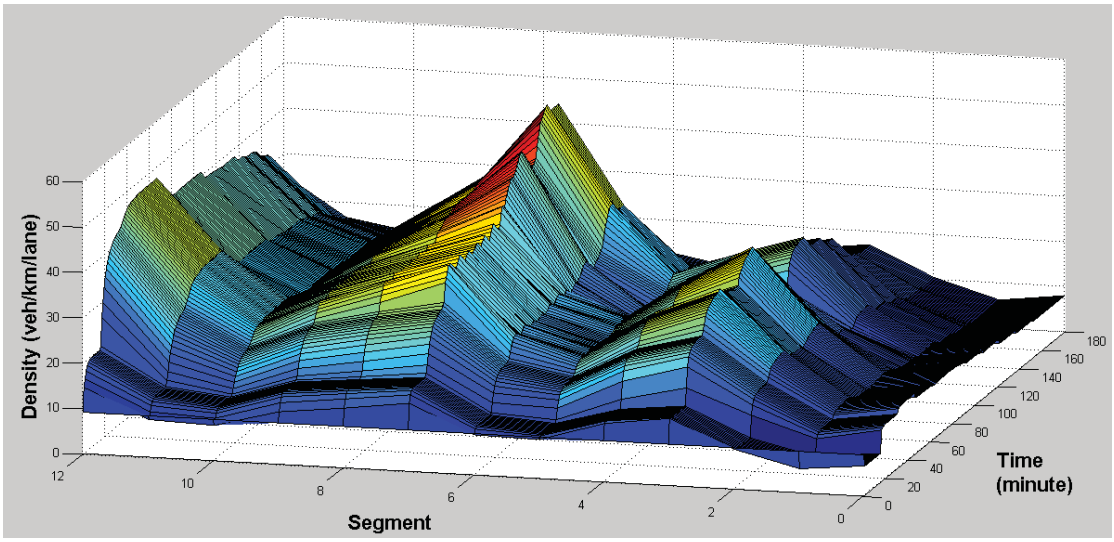


The speed profile under the control of ALINEA/Q (Test 4)

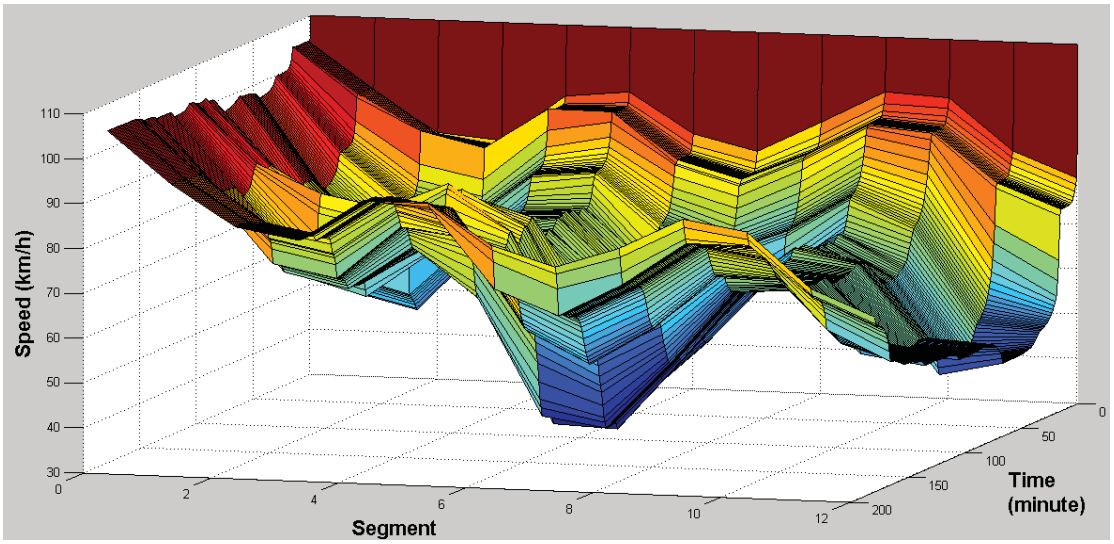


On-ramp queues under the control of ALINEA/Q (Test 4)

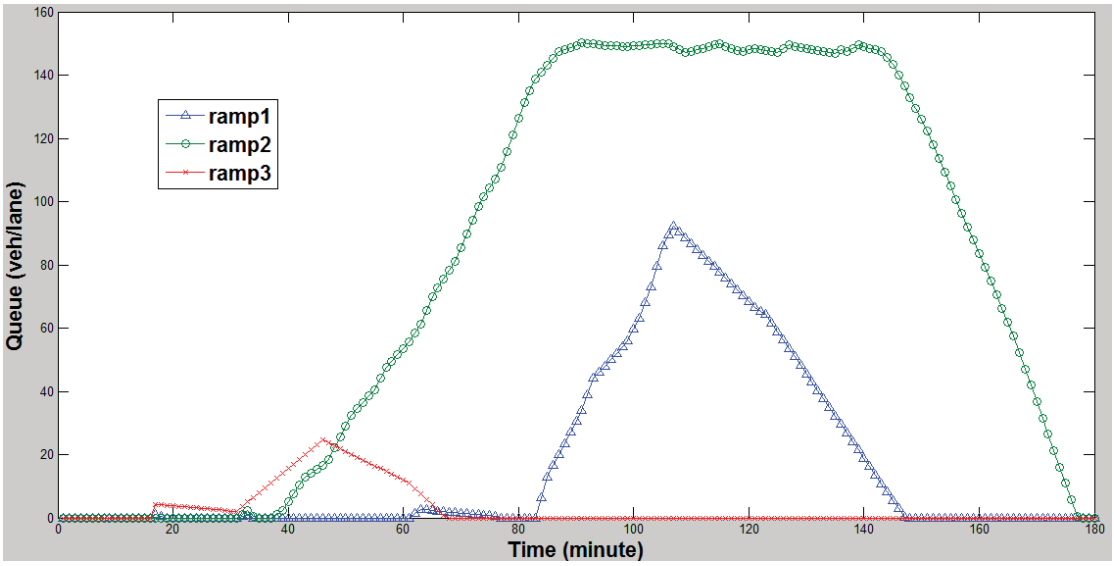
A.2. Simulation results of DP- the first phase of search (Test2 ~ Test3)



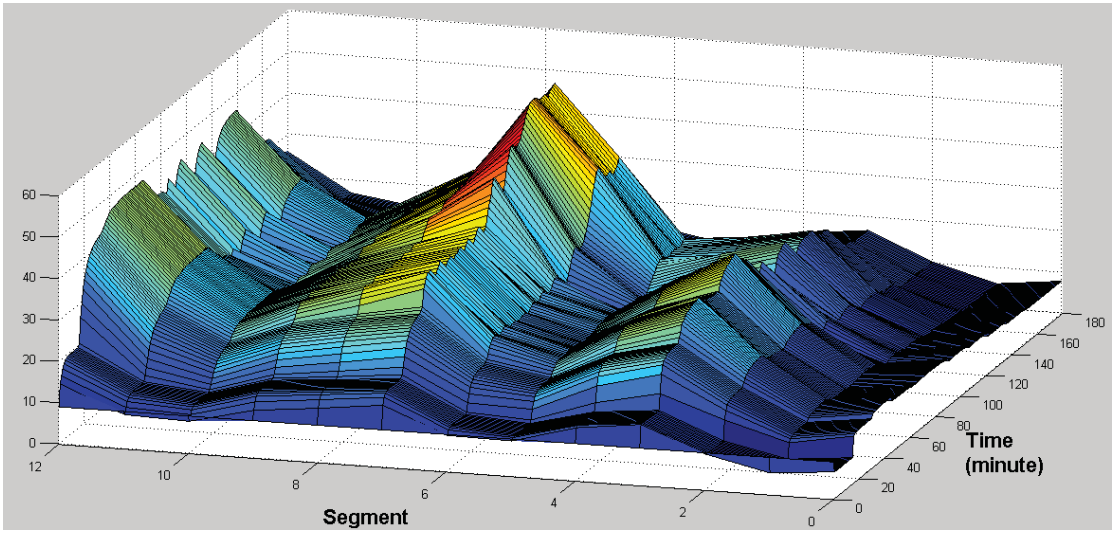
The density profile under the control of DP (the first phase of search) (Test 2)



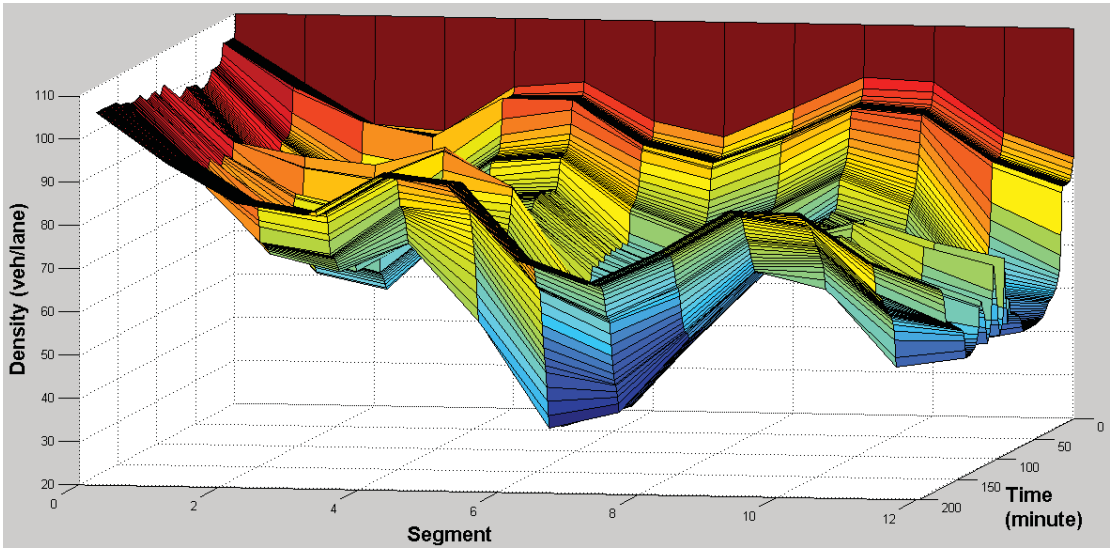
The speed profile under the control of DP (the first phase of search) (Test 2)



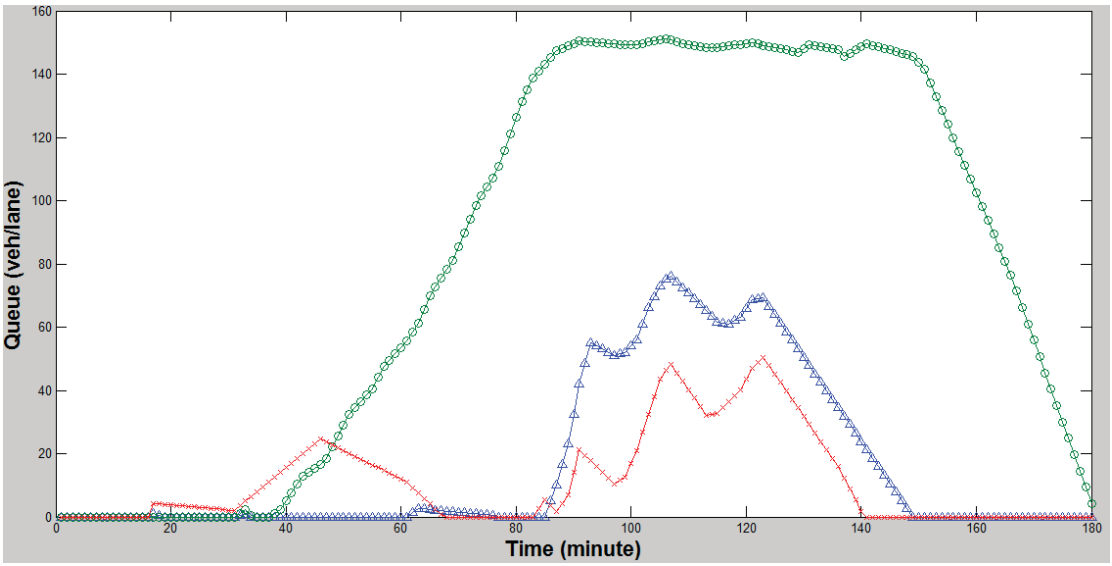
On-ramp queues under the control of DP - the first phase of search (Test 2)



The density profile under the control of DP (the first phase of search) (Test 3)

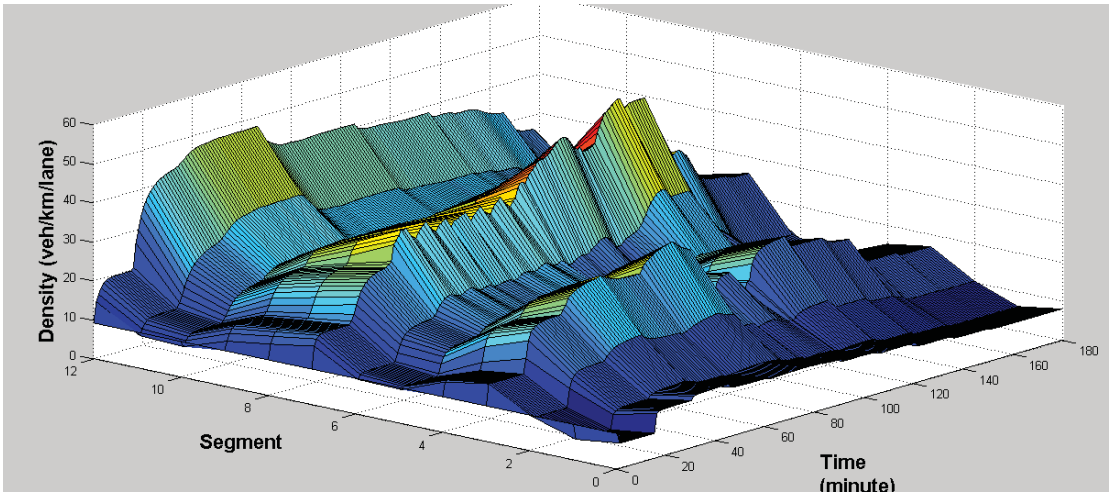


The speed profile under the control of DP (the first phase of search) (Test 3)

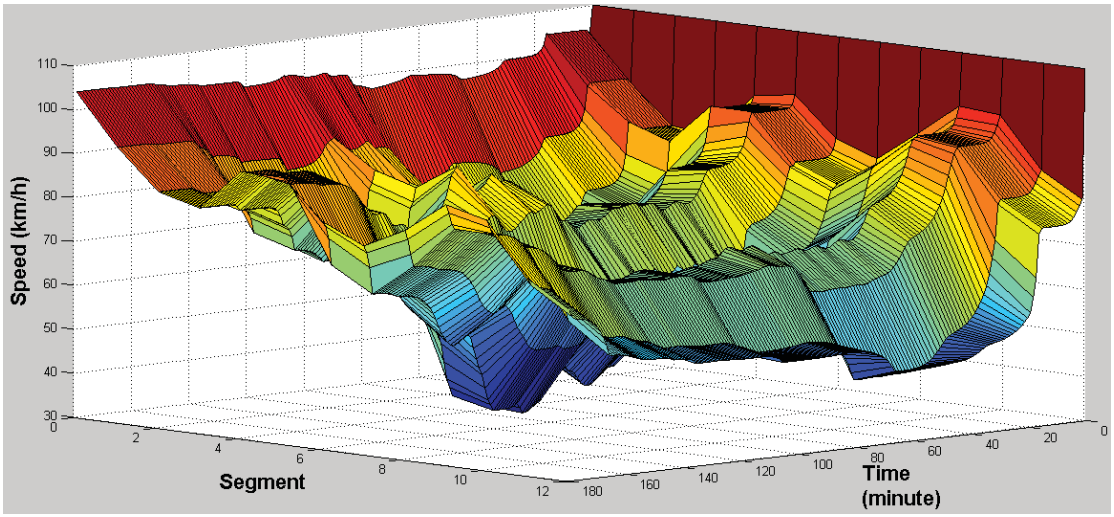


On-ramp queues under the control of DP - the first phase of search (Test 3)

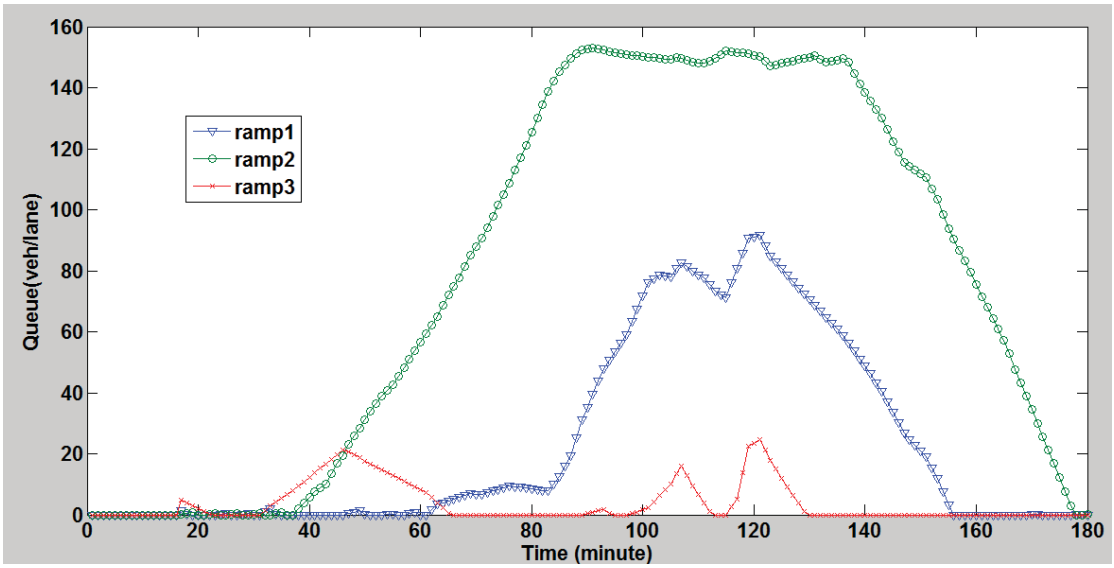
A.3. Simulation results of DP- the first phase of search (Test2 ~ Test4)



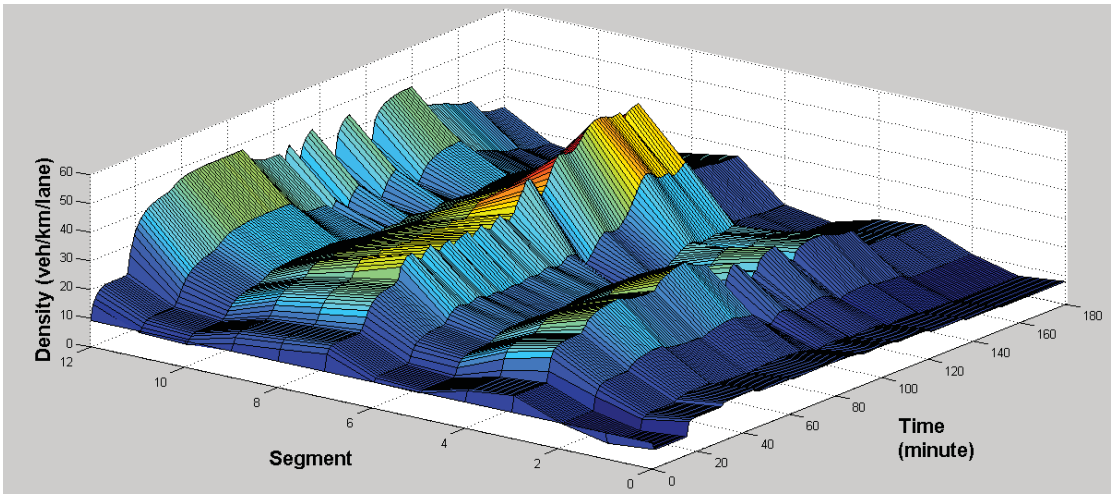
The density profile under the control of DP (two phases of search) (Test 2)



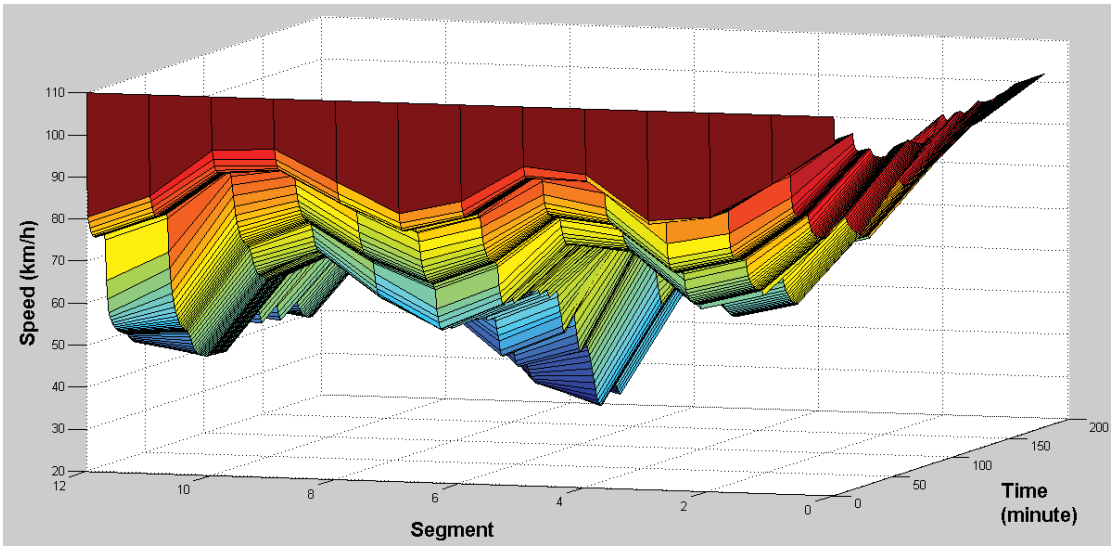
The speed profile under the control of DP (two phases of search) (Test 2)



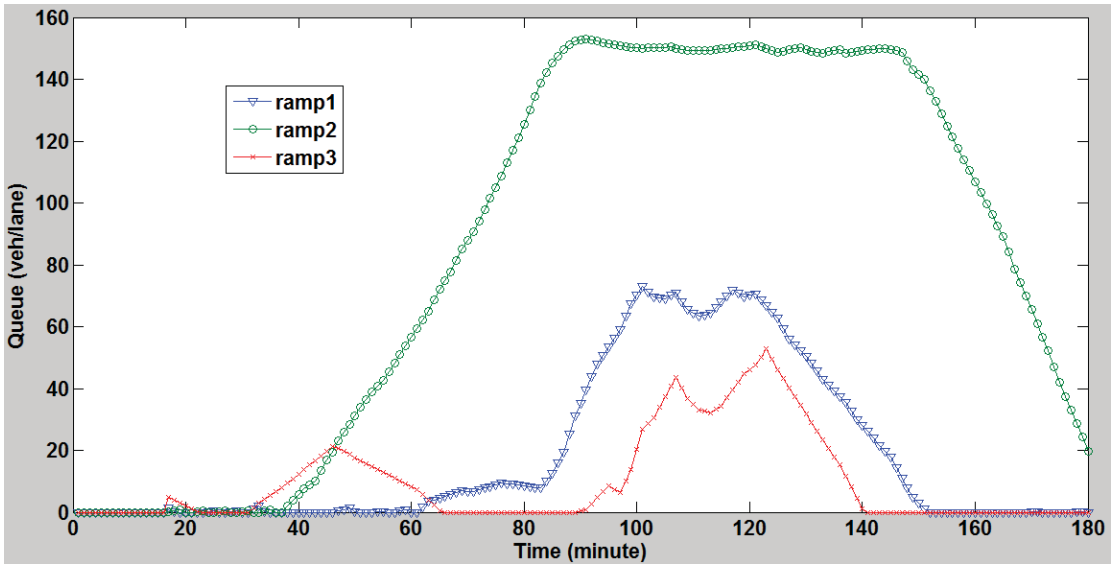
On-ramp queues under the control of DP - two phases of search (Test 2)



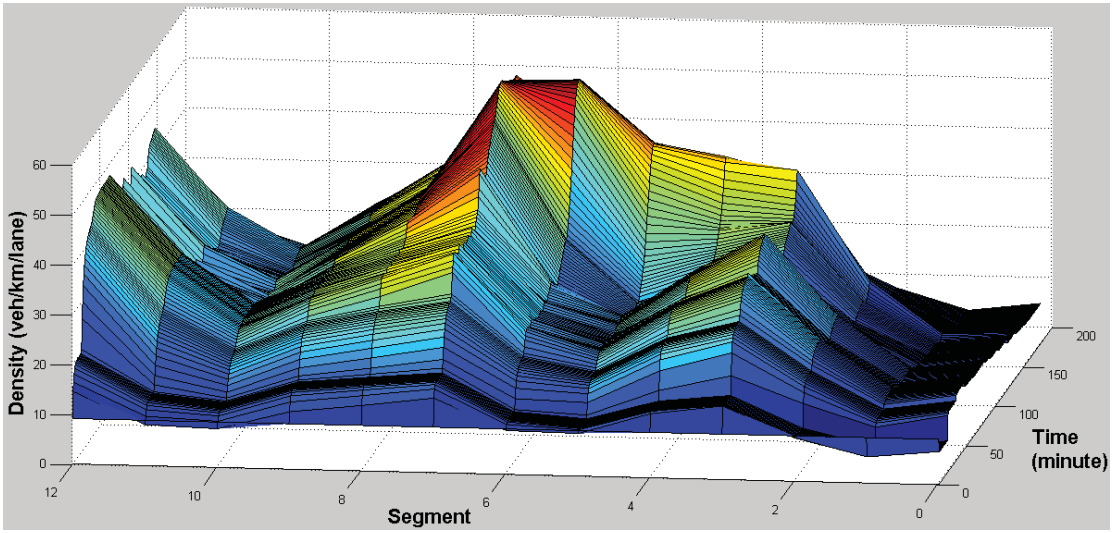
The density profile under the control of DP (two phases of search) (Test 3)



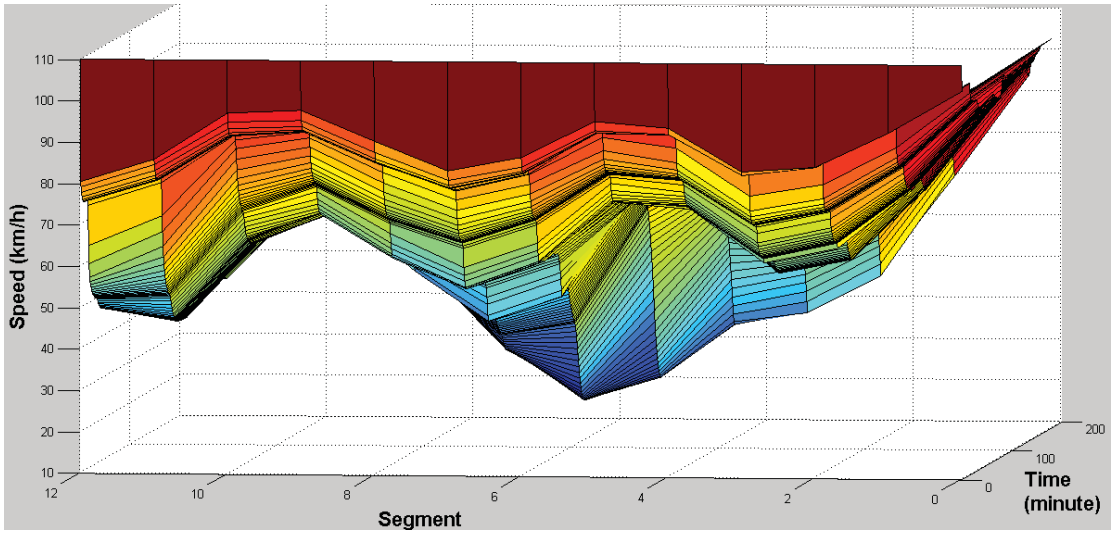
The speed profile under the control of DP (two phases of search) (Test 3)



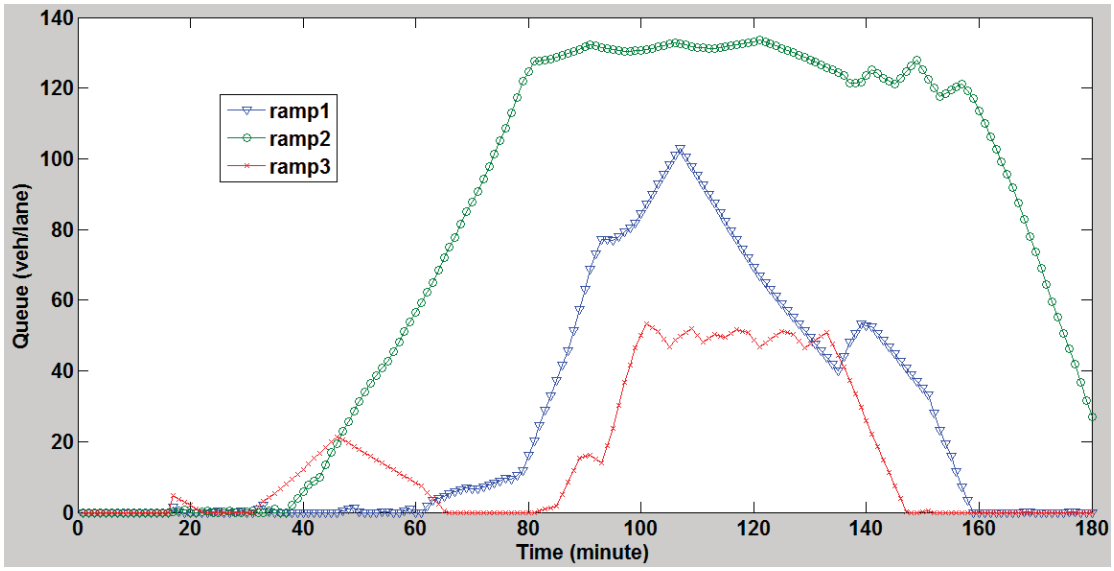
On-ramp queues under the control of DP - two phases of search (Test 3)



The density profile under the control of DP (two phases of search) (Test 4)



The speed profile under the control of DP (two phases of search) (Test 4)



On-ramp queues under the control of DP - two phases of search (Test 4)

APPENDIX B: SOURCE CODE FOR MACROSCOPIC TRAFFIC SIMULATION

```
// SOURCE CODES FOR CHAPTER 4;
// "METANET_DP.H" INCLUDES CLASSES DEFINED FOR DP AND THE EMPLOYED MACROSCOPIC MODEL (APPENDIX D)
// Yu Xue Feng - PHD CANDIDATE OF ENGINEERING IN MECHATRONICS
// SEAT - MASSEY UNIVERSITY - AUCKLAND - NEW ZEALAND
// 2010-2014 - All rights reserved
#include "stdafx.h"
#include "METANET_DP.h"
int _tmain(int argc, _TCHAR* argv[])
{
	//link_para [link_index][v_free, p_crit, romax, a_m, t_m, v_m, k_m, ramp_m, weaving_m]
	//seg_map
[seg_index][0_link_index,1_lane_number,2_chk_ramp_effect(0_0,1_on,2_off)),3_chk_weaving_effect(0_none,num_diff
erencs of lanes), 4_chk_d_or_o(0_none, 1_origin, 2_destination)]
	float link_map[8][2]={{1,0.4f},{1,0.3f},{1,0.5f},{2,0.55f},{2,0.55f},{3,0.5f},{2,0.45f},{1,0.4f}};
	int seg_map[13][5]={	{0,3,0,0,1},
				{1,3,2,0/*1*/,0},
				{2,2,0,0,0},
				{3,2,1,0,0},{3,2,0,0,0},
				{4,2,2,0,0},{4,2,0,0,0},
				{5,2,1,0,0},{5,2,0,0,0},{5,2,0,0,0},
				{6,2,2,0,0},{6,2,0,0,0},
				{7,2,1,0,2}};
	float link_para[8][10]=		{{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100},
					{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100},
					{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100},
					{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100},
					{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100},
					{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100},
					{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100},
					{100,31.4f,180,2.0f,0.0056f,35,13,0.8f,2,2100}};
	float seg_initial[13][3]={		{9.09f,110,2000},
					{9.09f,110,2000},
					{9.09f,110,2000},
					{9.09f,110,2000},
					{9.09f,110,2000},
					{9.09f,110,2000},
					{9.09f,110,2000},
```

```
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000}};

float seg_recovery[13][3]={ {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000},
                                        {9.09f,110,2000}};
int num_link=8;
int sim_step=12;
int num_on_ramp=3;
int num_off_ramp=3;
float origin_demand=2000;
float origin_queue=0;
int num_seg=0;
for(int i=0; i<num_link; i++)
num_seg=int(link_map[i][0])+num_seg;
//initialization
segment *seg = new segment[num_seg];
for(int i=0; i<num_seg; i++)
{
      seg[i].get_para(i,link_map,seg_map,link_para);
}
ramp *on= new ramp[num_on_ramp];
on[0].initial_ramp(2,900);
on[0].initial_ramp_demand(1800);
on[0].initial_ramp_metering_rate(1);
on[0].initial_ramp_queue(0);
on[0].initial_ramp_location(0,3);
on[1].initial_ramp(2,900);
on[1].initial_ramp_demand(1800);
on[1].initial_ramp_metering_rate(1);
```

```
on[1].initial_ramp_queue(0);
on[1].initial_ramp_location(1,7);
on[2].initial_ramp(2,900);
on[2].initial_ramp_demand(1800);
on[2].initial_ramp_metering_rate(1);
on[2].initial_ramp_queue(0);
on[2].initial_ramp_location(2,12);
off_ramp *off= new off_ramp[num_off_ramp];
off[0].initial_off_ramp_rate(0.3f);
off[0].initial_off_ramp_inflow(0);
off[0].initial_ramp_location(0,1);
off[1].initial_off_ramp_rate(0.3f);
off[1].initial_off_ramp_inflow(0);
off[1].initial_ramp_location(1,5);
off[2].initial_off_ramp_rate(0.3f);
off[2].initial_off_ramp_inflow(0);
off[2].initial_ramp_location(2,10);
int time_slice=0;
float origin_queue_recovery=origin_queue;
float ramp_queue_initial[3]={on[0].get_ramp_queue(),on[1].get_ramp_queue(),on[2].get_ramp_queue()};
float ramp_queue_recovery[3]={on[0].get_ramp_queue(),on[1].get_ramp_queue(),on[2].get_ramp_queue()};
float ramp_demand_matrix[12][3]={{1000,1100,1000},
                                 {1200,1200,1600},
                                 {1300,1400,1800},
                                 {1400,1500,1500},
                                 {1600,1600,1400},
                                 {1500,1700,1400},
                                 {1500,1600,1400},
                                 {1400,1500,1300},
                                 {1300,1400,1300},
                                 {1300,1200,1200},
                                 {1100,1100,1200},
                                 {1000,1000,1000}};
float origin_demand_matrix[12]={2000,3600,3900,3600,3500,3500,3300,3100,2800,2600,2500,2500};
float solution[5][3]={{1,1,1},
                      {1,1,1},
                      {1,1,1},
                      {1,1,1},
                      {1,1,1}};
float TTS=0;
TTS=0;
for(int k=0; k<180; k++)
{
      //Redefine traffic demand
```

```
if(k==time_slice)
{
	//cout<<"time_slice:"<<time_slice<<endl;
	for(int i=0; i<3; i++)
	{
		on[i].initial_ramp_demand(ramp_demand_matrix[time_slice/15][i]);
	}
	origin_demand=origin_demand_matrix[time_slice/15];//time_slice/15
	time_slice=time_slice+15;
}
for(int i=0;i<13;i++)
{
		sprintf_s(file_name, "G:\\Research\\Test\\seg %d.csv",i);
		ofstream fout(file_name, ios::out|ios::app);
		if(!fout)
			{
				cout<<"open failed!"<<endl;
				exit(1);
			}
		sprintf_s(astring," %f, %f,  %f",
		seg_initial[i][0],seg_initial[i][1],seg_initial[i][2]);//d,s,f
		fout << astring << endl;
		fout.close();
}
for(int i=0; i<13; i++)
{
	seg_recovery[i][0]=seg_initial[i][0];
	seg_recovery[i][1]=seg_initial[i][1];
	seg_recovery[i][2]=seg_initial[i][2];
}
for(int i=0; i<3; i++)
{
	ramp_queue_recovery[i]=ramp_queue_initial[i];
	on[i].initial_ramp_metering_rate(solution[0][i]);
	//on[i].initial_ramp_metering_rate(1);
	on[i].initial_ramp_queue(ramp_queue_initial[i]);
}
for(int i=0;i<3;i++)
{
		sprintf_s(file_name, "G:\\Research\\Test\\on %d.csv",i);
		ofstream fout(file_name, ios::out|ios::app);
		if(!fout)
			{
				cout<<"open failed!"<<endl;
```

```
                                    exit(1);
                              }
                        sprintf_s(astring," %f, %f, %f", ramp_queue_initial[i], on[i].get_ramp_metering_rate(),
on[i].get_ramp_inflow());
                        fout << astring << endl;
                        fout.close();
            }
            origin_queue_recovery=origin_queue;
            float *main=&origin_queue;
            TTS = TTS+metanet_revised(ramp_queue_initial,num_seg, seg,main, origin_demand, num_link, num_on_ramp,
num_off_ramp, sim_step, link_map, seg_map, link_para, seg_initial,  on, off);
            cout<<"TTS:"<<TTS<<"; mins:"<<k<<endl;
            //DP search
            if ((k+1)%2==0)
            {
                  stage group[6];
                  float starting_range[3][2];
                  float map[6];
                  for (int i=0; i<3; i++)
                  {
                        starting_range[i][0]=float(0.1);
                        starting_range[i][1]=float(0.9);
                  }
                  for(int i=0; i<6; i++)
                  {
                        group[i].initial_stage(9);
                        group[i].initial_stage_index(i);
                        group[i].initial_states_index();
                        group[i].initial_range(starting_range);
                        group[i].update_boundary();
                        group[i].update_combinations();
                        group[i].update_states_rm();
                  }

      group[0].initial_stage(origin_queue_recovery,ramp_queue_recovery[0],ramp_queue_recovery[1],ramp_queue_rec
overy[2],seg_recovery);
                  for(int i=0; i<5; i++)
                  {
                        if(i==0)
                        {
                              group[0].run_from_initial_stage(group[1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, 24, link_map, seg_map, link_para, on, off);
                              //group[1].print_tran_map();
                              //cout<<"group1"<<endl;
```

```
                        }
                        else
                        {
                              group[i].run_to_next_stage_revised(group[i+1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, 24, link_map, seg_map, link_para, on, off);
                              //cout<<"group["<<i<<"]"<<endl;
                        }
                  }
                  //group[10].print_tran_map();
                  mapping_path(group,map,solution);

            // Second search

                  for(int i=0; i<13; i++)
            {
                  seg_recovery[i][0]=seg_initial[i][0];
                  seg_recovery[i][1]=seg_initial[i][1];
                  seg_recovery[i][2]=seg_initial[i][2];
            }
            for(int i=0; i<3; i++)
            {
                  ramp_queue_recovery[i]=ramp_queue_initial[i];
                  //on[i].initial_ramp_metering_rate(solution[0][i]);
                  //on[i].initial_ramp_metering_rate(1);
                  on[i].initial_ramp_queue(ramp_queue_initial[i]);
            }
            for(int i=1; i<6; i++)
            {
                  group[i].update_range(map);
            }
            for(int i=0; i<6; i++)
            {
                  group[i].clear_stage();
                  group[i].initial_stage(5);
                  group[i].initial_stage_index(i);
                  group[i].initial_states_index();
                  //group[i].update_range(map);
                  group[i].update_boundary();
                  group[i].update_combinations();
                  group[i].update_states_rm();
            }

      group[0].initial_stage(origin_queue_recovery,ramp_queue_recovery[0],ramp_queue_recovery[1],ramp_queue_rec
overy[2],seg_recovery);
```

```
            //group[1].print_stats_rm();
            for(int i=0; i<5; i++)
                  {
                        if(i==0)
                        {
                              group[0].run_from_initial_stage(group[1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, sim_step/5, link_map, seg_map, link_para, on, off);
                              //group[1].print_tran_map();
                              //cout<<"group1"<<endl;
                        }
                        else
                        {
                              group[i].run_to_next_stage(group[i+1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, sim_step/5, link_map, seg_map, link_para, on, off);
                              //cout<<"group["<<i<<"]"<<endl;
                        }
                  }
            mapping_path(group,map,solution);
            // leave mark here for the first DP search
            }
            for(int i=0; i<5; i++)
                  for(int j=0; j<3; j++)
                        cout<<"solution["<<i<<"]["<<"["<<j<<"]:"<<solution[i][j]<<endl;
      //Leave mark here for no control case
      }
      cout<<"TTS:"<<TTS<<endl;
      sprintf_s(file_name, "G:\\Research\\Test\\TTS.txt");
      ofstream fout(file_name, ios::out|ios::app);
      if(!fout)
                              {
                                          cout<<"open failed!"<<endl;
                                          exit(1);
                              }
      sprintf_s(astring," %f", TTS);
      fout << astring << endl;
      fout.close();
      delete [] on;
      delete [] off;
      delete [] seg;
      //delete [] fitnessVector;
      //delete [] rmVector;
      //delete [] drmVector;
      return 0;
```

APPENDIX C: SOURCE CODE FOR AIMSUN API MODULE

```
// SOURCE CODE FOR CHAPTER 5
// "METANET_DP.H" INCLUDES CLASSES DEFINED FOR DP AND THE EMPLOYED MACROSCOPIC MODEL (APPENDIX D)
#include "AKIProxie.h"
#include "CIProxie.h"
#include "ANGConProxie.h"
#include "AAPI.h"
#include <stdio.h>
#include "METANET_DP.h"
int carPosition;
void seg_initialization(float seg_initial[][3], float seg_recovery[][3])
{
	for(int i=0; i<13; i++)
	{
		seg_initial[i][0]=seg_recovery[i][0];
		seg_initial[i][1]=seg_recovery[i][1];
		seg_initial[i][2]=seg_recovery[i][2];
	}
}
int AAPILoad()
{
	return 0;
}
int AAPIInit()
{
	carPosition=AKIVehGetVehTypePosition(AKIConvertFromAsciiString("car"));
	ANGConnEnableVehiclesInBatch(true);
	return 0;
}
int AAPIManage(double time, double timeSta, double timTrans, double acicle)
{
	return 0;
}
int AAPIPostManage(double time, double timeSta, double timTrans, double acicle)
{
	StructAkiEstadSection s[13];
	s[0]=AKIEstGetParcialStatisticsSection(691,timeSta,carPosition);
	s[1]=AKIEstGetParcialStatisticsSection(661,timeSta,carPosition);
	s[2]=AKIEstGetParcialStatisticsSection(1248,timeSta,carPosition);
	s[3]=AKIEstGetParcialStatisticsSection(649,timeSta,carPosition);
	s[4]=AKIEstGetParcialStatisticsSection(647,timeSta,carPosition);
	s[5]=AKIEstGetParcialStatisticsSection(567,timeSta,carPosition);
```

```
	s[6]=AKIEstGetParcialStatisticsSection(1297,timeSta,carPosition);
	s[7]=AKIEstGetParcialStatisticsSection(544,timeSta,carPosition);
	s[8]=AKIEstGetParcialStatisticsSection(1315,timeSta,carPosition);
	s[9]=AKIEstGetParcialStatisticsSection(1309,timeSta,carPosition);
	s[10]=AKIEstGetParcialStatisticsSection(270,timeSta,carPosition);
	s[11]=AKIEstGetParcialStatisticsSection(1318,timeSta,carPosition);
	s[12]=AKIEstGetParcialStatisticsSection(344,timeSta,carPosition);
	StructAkiEstadSection r[3];
	r[0]=AKIEstGetParcialStatisticsSection(695,timeSta,carPosition);
	r[1]=AKIEstGetParcialStatisticsSection(548,timeSta,carPosition);
	r[2]=AKIEstGetParcialStatisticsSection(316,timeSta,carPosition);
if(s[0].report==0)
		{
			int num_slice=AKIStateDemandGetNumSlices();
			int time_slice=0;
			for (int i=0; i<num_slice; i++)
			{
				if(timeSta>=AKIStateDemandGetIniTimeSlice(i)&&timeSta<=AKIStateDemandGetEndTimeSlice(i))
				time_slice=i;
			}
			double highway_demand=AKIStateDemandGetDemandSection(691,carPosition,time_slice);
			double ramp1_demand=AKIStateDemandGetDemandSection(695,carPosition,time_slice);
			double ramp2_demand=AKIStateDemandGetDemandSection(548,carPosition,time_slice);
			double ramp3_demand=AKIStateDemandGetDemandSection(316,carPosition,time_slice);
			float
link_map[8][2]={{1,0.4f},{1,0.3f},{1,0.5f},{2,0.55f},{2,0.55f},{3,0.5f},{2,0.45f},{1,0.4f}};
			int seg_map[13][5]={{0,3,0,0,1},
                                {1,3,2,1,0},
                                {2,2,0,0,0},
                                {3,2,1,0,0},{3,2,0,0,0},
                                {4,2,2,0,0},{4,2,0,0,0},
                                {5,2,1,0,0},{5,2,0,0,0},{5,2,0,0,0},
                                {6,2,2,0,0},{6,2,0,0,0},
                                {7,2,1,0,2}};
			float link_para[8][10]={{110,33,180,1.7f,0.0055f,34.75f,13,0.8f,2,2100},
                                    {110,33,180,1.7f,0.0058f,14.51f,13,0.8f,2,2100},
                                    {110,33,180,1.7f,0.0051f,2.12f,13,0.8f,2,2100},
                                    {110,33,180,1.7f,0.0049f,100.51f,13,1.5f,2,2100},
                                    {110,33,180,1.7f,0.0056f,15.13f,13,0.8f,2,2100},
                                    {110,33,180,1.7f,0.0042f,150.29f,13,1.8f,2,2100},
                                    {110,33,180,1.7f,0.0051f,20.31f,13,0.8f,2,2100},
                                    {110,33,180,1.7f,0.0053f,120.97f,13,2.5f,2,2100}};
			float seg_initial[13][3]={  {float(s[0].Flow/(s[0].SHa*3)),float(s[0].SHa),float(s[0].Flow)},
                                        {float(s[1].Flow/(s[1].SHa*3)),float(s[1].SHa),float(s[1].Flow)},
```

```
                                {float(s[2].Flow/(s[2].SHa*2)),float(s[2].SHa),float(s[2].Flow)},
                                {float(s[3].Flow/(s[3].SHa*2)),float(s[3].SHa),float(s[3].Flow)},
                                {float(s[4].Flow/(s[4].SHa*2)),float(s[4].SHa),float(s[4].Flow)},
                                {float(s[5].Flow/(s[5].SHa*2)),float(s[5].SHa),float(s[5].Flow)},
                                {float(s[6].Flow/(s[6].SHa*2)),float(s[6].SHa),float(s[6].Flow)},
                                {float(s[7].Flow/(s[7].SHa*2)),float(s[7].SHa),float(s[7].Flow)},
                                {float(s[8].Flow/(s[8].SHa*2)),float(s[8].SHa),float(s[8].Flow)},
                                {float(s[9].Flow/(s[9].SHa*2)),float(s[9].SHa),float(s[9].Flow)},
                           {float(s[10].Flow/(s[10].SHa*2)),float(s[10].SHa),float(s[10].Flow)},
                           {float(s[11].Flow/(s[11].SHa*2)),float(s[11].SHa),float(s[11].Flow)},
                           {float(s[12].Flow/(s[12].SHa*2)),float(s[12].SHa),float(s[12].Flow)}};
float seg_recovery[13][3]={ {float(s[0].Flow/(s[0].SHa*3)),float(s[0].SHa),float(s[0].Flow)},
                                {float(s[1].Flow/(s[1].SHa*3)),float(s[1].SHa),float(s[1].Flow)},
                                {float(s[2].Flow/(s[2].SHa*2)),float(s[2].SHa),float(s[2].Flow)},
                                {float(s[3].Flow/(s[3].SHa*2)),float(s[3].SHa),float(s[3].Flow)},
                                {float(s[4].Flow/(s[4].SHa*2)),float(s[4].SHa),float(s[4].Flow)},
                                {float(s[5].Flow/(s[5].SHa*2)),float(s[5].SHa),float(s[5].Flow)},
                                {float(s[6].Flow/(s[6].SHa*2)),float(s[6].SHa),float(s[6].Flow)},
                                {float(s[7].Flow/(s[7].SHa*2)),float(s[7].SHa),float(s[7].Flow)},
                                {float(s[8].Flow/(s[8].SHa*2)),float(s[8].SHa),float(s[8].Flow)},
                                {float(s[9].Flow/(s[9].SHa*2)),float(s[9].SHa),float(s[9].Flow)},
                           {float(s[10].Flow/(s[10].SHa*2)),float(s[10].SHa),float(s[10].Flow)},
                           {float(s[11].Flow/(s[11].SHa*2)),float(s[11].SHa),float(s[11].Flow)},
                           {float(s[12].Flow/(s[12].SHa*2)),float(s[12].SHa),float(s[12].Flow)}};
int num_link=8;
int sim_step=120;
int num_on_ramp=3;
int num_off_ramp=3;
float origin_demand=float(highway_demand);
float origin_queue=0;
int num_seg=0;
for(int i=0; i<num_link; i++)
num_seg=int(link_map[i][0])+num_seg;
//initialization
segment *seg = new segment[num_seg];
for(int i=0; i<num_seg; i++)
{
      seg[i].get_para(i,link_map,seg_map,link_para);
}
ramp *on= new ramp[num_on_ramp];
on[0].initial_ramp(2,900);
on[0].initial_ramp_demand(float(ramp1_demand));
on[0].initial_ramp_metering_rate(1);
on[0].initial_ramp_queue(float(r[0].LongQueueAvg*2));
```

```
on[0].initial_ramp_location(0,3);
on[1].initial_ramp(2,900);
on[1].initial_ramp_demand(float(ramp2_demand));
on[1].initial_ramp_metering_rate(1);
on[1].initial_ramp_queue(float(r[1].LongQueueAvg*2));
on[1].initial_ramp_location(1,7);
on[2].initial_ramp(2,900);
on[2].initial_ramp_demand(float(ramp3_demand));
on[2].initial_ramp_metering_rate(1);
on[2].initial_ramp_queue(float(r[2].LongQueueAvg*2));
on[2].initial_ramp_location(2,12);
off_ramp *off= new off_ramp[num_off_ramp];
off[0].initial_off_ramp_rate(0.3f);
off[0].initial_off_ramp_inflow(0);
off[0].initial_ramp_location(0,1);
off[1].initial_off_ramp_rate(0.3f);
off[1].initial_off_ramp_inflow(0);
off[1].initial_ramp_location(1,5);
off[2].initial_off_ramp_rate(0.3f);
off[2].initial_off_ramp_inflow(0);
off[2].initial_ramp_location(2,10);
float TTS=0;
TTS = metanet(num_seg, seg,origin_queue, origin_demand, num_link, num_on_ramp, num_off_ramp,
sim_step, link_map, seg_map, link_para, seg_initial, seg_recovery, on, off);
sprintf_s(astring,"\t\t TTS= %f",TTS);
AKIPrintString(astring);
//Dynamic programming
stage group[6];
float starting_range[3][2];
float map[6];
float solution[5][3];
for (int i=0; i<3; i++)
{
      starting_range[i][0]=float(0.1f);
      starting_range[i][1]=float(0.9f);
}
for(int i=0; i<6; i++)
{
      group[i].initial_stage(9);
      group[i].initial_stage_index(i);
      group[i].initial_states_index();
      group[i].initial_range(starting_range);
      group[i].update_boundary();
      group[i].update_combinations();
```

```
                group[i].update_states_rm();
            }
        group[0].initial_stage(origin_queue,on[0].get_ramp_queue(),on[1].get_ramp_queue(),on[2].get_ramp_queue(),
seg_initial);
            for(int i=0; i<5; i++)
            {
                if(i==0)
                {    group[0].run_from_initial_stage(group[1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, sim_step/5, link_map, seg_map, link_para, on, off);
                    //group[1].print_tran_map();
                    //cout<<"group1"<<endl;
                }
                else
                {
                    group[i].run_to_next_stage_revised(group[i+1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, sim_step/5, link_map, seg_map, link_para, on, off);
                    //cout<<"group["<<i<<"]"<<endl;
                }
            }
            //group[10].print_tran_map();
            mapping_path(group,map,solution);
            //cout<<"min_obj:"<<map[10]<<endl;
            sprintf_s(astring,"\t\t TTS= %f",map[5]);
            AKIPrintString(astring);
            for(int j=0; j<3; j++)
            {sprintf_s(astring,"\t\t r%d= %f",j,solution[0][j]);
            AKIPrintString(astring);}
            // the second search
            for(int i=1; i<6; i++)
            {
                group[i].update_range(map);
            }

            for(int i=0; i<6; i++)
            {
                group[i].clear_stage();
                group[i].initial_stage(5);
                group[i].initial_stage_index(i);
                group[i].initial_states_index();
                //group[i].update_range(map);
                group[i].update_boundary();
                group[i].update_combinations();
                group[i].update_states_rm();
            }
```

```
	group[0].initial_stage(origin_queue,on[0].get_ramp_queue(),on[1].get_ramp_queue(),on[2].get_ramp_queue(),
seg_initial);
		//group[1].print_stats_rm();
		for(int i=0; i<5; i++)
			{
				if(i==0)
				{
					group[0].run_from_initial_stage(group[1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, sim_step/5, link_map, seg_map, link_para, on, off);
					//group[1].print_tran_map();
					//cout<<"group1"<<endl;
				}
				else
				{
					group[i].run_to_next_stage(group[i+1],num_seg, seg, origin_demand, num_link,
num_on_ramp, num_off_ramp, sim_step/5, link_map, seg_map, link_para, on, off);
					//cout<<"group["<<i<<"]"<<endl;
				}
			}
		mapping_path(group,map,solution);
		cout<<"min_obj:"<<map[5]<<endl;
		for(int i=0; i<5; i++)
				for(int j=0; j<3; j++)
					cout<<"solution["<<i<<"]["<<"["<<j<<"]:"<<solution[i][j]<<endl;
	ECIChangeParametersFlowMeteringById(919,timeSta,solution[0][0]*1800,solution[0][0]*1800,solution[0][0]*18
00);
	ECIChangeParametersFlowMeteringById(908,timeSta,solution[0][1]*1800,solution[0][1]*1800,solution[0][1]*18
00);
	ECIChangeParametersFlowMeteringById(1026,timeSta,solution[0][2]*1800,solution[0][2]*1800,solution[0][2]*1
800);
		delete [] on;
		delete [] off;
		delete [] seg;
	}
	return 0;
}
int AAPIFinish()
{	return 0;
}
int AAPIUnLoad()
{	return 0;
}
```

APPENDIX D: C++ CLASSES DEFINED FOR DYNAMIC PROGRAMMING AND THE MACROSCOPIC TRAFFIC MODEL

D.1. Classes Defined for the Employed Macroscopic Traffic Model

```
// "METANET_DP.H"

#include <stdio.h>
#include <math.h>
#include <iostream>
#include <fstream>
using namespace std;
#ifndef MAX
# define MAX(x,y) ( (x) > (y) ? (x) : (y) )
#endif
#ifndef MIN
# define MIN(x,y) ( (x) < (y) ? (x) : (y) )
#endif
char astring[128];
char file_name[128];
class segment
{
      float L;
    const float T;
    int num_lane;
      int on_ramp_index;
      int off_ramp_index;
      enum ramp_type {on_ramp ,off_ramp, none} chk_ramp;
      enum weaving_effect {on, off} chk_weaving;
      enum origin {yes, no} chk_origin;
      enum destination {ye, not} chk_destination;
      int weaving_lanes;
      //inputs and outputs
      float q_in;
      float q_out;
      float density;
      float speed;
      //link parameters
      float q_max;
      float v_free;
      float p_crit;
      float p_jam;
      float a_m;
```

```
	float t_m;
	float v_m;
	float k_m;
	float ramp_m;
	float weaving_m;
	//speed parameters
	float equilibrium_speed;
	float relaxation;
	float convection;
	float anticipation;
	float merging_term_on_ramp;
	float weaving_term;
public :
	segment();
	void get_para(int , float link_map [][2], int seg_map[/*seg_index*/][5], float link_para[][10]);
	void initial(float , float );
	void initial_q_out(float );
	void initial_q_in(float );
	float get_origin();
	float get_destination();
	float get_on_ramp();
	float get_off_ramp();
	float get_weave();
	float get_q_in();
	float get_q_out();
	float get_density();
	float get_speed();
	int get_num_lane();
	float get_lane_L();
	float get_q_max();
	float get_p_crit();
	float next_density();
	float next_speed(float , float );
	float next_speed_weave(float , float );
	float next_speed_origin(float );
	float next_speed_destination(float );
	float next_speed_on_ramp(float , float , int , float );
	float next_speed_on_ramp_destination(float  , int , float );
};
segment::segment():T(0.00139f)
{}
void segment :: get_para(int seg_index, float link_map [][2], int seg_map[/*seg_index*/][5], float link_para[][10])
	{
		//Get the parameters of the link
```

```
int link_index;
link_index = seg_map[seg_index][0];
L=link_map[link_index][1];
v_free = link_para[link_index][0];
p_crit = link_para[link_index][1];
p_jam = link_para[link_index][2];
a_m = link_para[link_index][3];
t_m = link_para[link_index][4];
v_m = link_para[link_index][5];
k_m = link_para[link_index][6];
ramp_m = link_para[link_index][7];
weaving_m = link_para[link_index][8];
q_max = link_para[link_index][9];
on_ramp_index=0;
off_ramp_index=0;
num_lane=seg_map[seg_index][1];
if(seg_map[seg_index][2]==0)
      chk_ramp = none;
else if (seg_map[seg_index][2]==1)
      {
            chk_ramp = on_ramp;
      }
else
      {
            chk_ramp = off_ramp;
      }
if(seg_map[seg_index][3]==0)
      {
            chk_weaving = off;
            weaving_lanes=0;
      }
else
      {
            chk_weaving = on;
            weaving_lanes=seg_map[seg_index][3];
      }
if(seg_map[seg_index][4]==0)
      {
            chk_origin=no;
            chk_destination=not;
      }
else if (seg_map[seg_index][4]==1)
      {
            chk_origin=yes;
```

```
                    chk_destination=not;
                }
            else
                {
                    chk_destination=ye;
                    chk_origin=no;
                }
      }
void segment :: initial(float init_density, float init_speed)
{
            density= init_density;
            speed= init_speed;
}
void segment :: initial_q_out(float next_segment_density)
{
            if(next_segment_density<180)
            q_out=density*speed*num_lane;
            else
            q_out=0;
}
void segment :: initial_q_in(float init_q_in)
{
            q_in=init_q_in;
}
float segment :: get_origin()
{
            if(chk_origin==yes)
                return(1);
            else
                return(0);
}
float segment::get_destination()
{
            if(chk_destination==ye)
                return(1);
            else
                return(0);
}
float segment:: get_on_ramp()
      {
            if(chk_ramp==on_ramp)
                return(1);
            else
                return(0);
```

```
}
float segment::get_off_ramp()
{
    if(chk_ramp==off_ramp)
        return(1);
    else
        return(0);
}
float segment:: get_weave()
{
    if(chk_weaving==on)
        return(1);
    else
        return(0);
}
float segment::get_q_in()
{return(q_in);}
float segment::get_q_out()
{ return(q_out);}
float segment::get_density()
{ return(density);}
float segment::get_speed()
{ return(speed);}
int segment::get_num_lane()
{ return(num_lane);}
float segment::get_lane_L()
{ return(L);}
float segment::get_q_max()
{ return(q_max);}
float segment::get_p_crit()
{ return(p_crit);}
float segment::next_density()
{
    float den=density+(T*(q_in-q_out)/(num_lane*L));
    if(den>180)
    return(180);
    else if(den<0)
    return(0);
    else
    return(den);
}
float segment::next_speed(float last_seg_speed, float next_seg_density)
{
    equilibrium_speed = v_free * exp((-1)*(1/a_m)*pow((density/p_crit),a_m));
```

```
            relaxation=T*(equilibrium_speed-speed)/t_m;
            convection=T*speed*(last_seg_speed-speed)/L;
            anticipation=(-1)*(T*v_m*(next_seg_density-density))/(t_m*L*(density+k_m));
            float temp=0;
            temp=speed+relaxation+convection+anticipation;
            return(temp);
      }
float segment::next_speed_weave(float last_seg_speed, float next_seg_density)
      {

            weaving_term =(-1)*weaving_m*T*weaving_lanes*density*speed*speed/(L*num_lane*p_crit);
            equilibrium_speed = v_free * exp((-1)*(1/a_m)*pow((density/p_crit),a_m));
            relaxation=T*(equilibrium_speed-speed)/t_m;
            convection=T*speed*(last_seg_speed-speed)/L;
            anticipation=(-1)*(T*v_m*(next_seg_density-density))/(t_m*L*(density+k_m));
            float temp=0;
            temp=speed+relaxation+convection+anticipation+weaving_term;
            return(temp);
      }
float segment::next_speed_origin(float next_seg_density)
      {
            equilibrium_speed = v_free * exp((-1)*(1/a_m)*pow((density/p_crit),a_m));
            relaxation=T*(equilibrium_speed-speed)/t_m;
            //convection=T*speed*(last_seg_speed-speed)/L;
            anticipation=(-1)*(T*v_m*(next_seg_density-density))/(t_m*L*(density+k_m));
            float temp=0;
            temp=speed+relaxation+anticipation;
            return(temp);
      }
float segment::next_speed_destination(float last_seg_speed)
      {
            equilibrium_speed = v_free * exp((-1)*(1/a_m)*pow((density/p_crit),a_m));
            relaxation=T*(equilibrium_speed-speed)/t_m;
            convection=T*speed*(last_seg_speed-speed)/L;
            if(density<p_crit)
            anticipation=(-1)*(T*v_m*(density-density))/(t_m*L*(density+k_m));
            else
            anticipation=(-1)*(T*v_m*(p_crit-density))/(t_m*L*(density+k_m));
            float temp=0;
            temp=speed+relaxation+convection+anticipation;
            return(temp);
      }
float segment::next_speed_on_ramp(float last_seg_speed, float next_seg_density, int last_seg_num_lane, float
q_ramp_in)
```

```
	{
		equilibrium_speed = v_free * exp((-1)*(1/a_m)*pow((density/p_crit),a_m));
		relaxation=T*(equilibrium_speed-speed)/t_m;
		convection=T*speed*(last_seg_speed-speed)/L;
		anticipation=(-1)*(T*v_m*(next_seg_density-density))/(t_m*L*(density+k_m));
		float temp=0;
		if(num_lane<=last_seg_num_lane)
		merging_term_on_ramp=(-1)*(ramp_m*T*q_ramp_in*speed)/((L*num_lane)*(density+k_m));
		else if((num_lane>last_seg_num_lane)&&((q_max*num_lane)<q_ramp_in))
		merging_term_on_ramp=(-1)*(ramp_m*T*(q_ramp_in-q_max*num_lane)*speed)/((L*num_lane)*(density+k_m));
		else
		merging_term_on_ramp=0;
		if(chk_destination==not)
		{
			temp=speed+relaxation+convection+anticipation+merging_term_on_ramp;
		}
		else
		{
			temp=speed+relaxation+convection+merging_term_on_ramp;
		}
		return(temp);
}
float segment::next_speed_on_ramp_destination(float last_seg_speed,  int last_seg_num_lane, float q_ramp_in )
{
		equilibrium_speed = v_free * exp((-1)*(1/a_m)*pow((density/p_crit),a_m));
		relaxation=T*(equilibrium_speed-speed)/t_m;
		convection=T*speed*(last_seg_speed-speed)/L;
		if(density<p_crit)
		anticipation=(-1)*(T*v_m*(density-density))/(t_m*L*(density+k_m));
		else
		anticipation=(-1)*(T*v_m*(p_crit-density))/(t_m*L*(density+k_m));

		float temp=0;
		if(num_lane<=last_seg_num_lane)
		merging_term_on_ramp=(-1)*(ramp_m*T*q_ramp_in*speed)/((L*num_lane)*(density+k_m));
		else if((num_lane>last_seg_num_lane)&&((q_max*num_lane)<q_ramp_in))
		merging_term_on_ramp=(-1)*(ramp_m*T*(q_ramp_in-q_max*num_lane)*speed)/((L*num_lane)*(density+k_m));
		else
		merging_term_on_ramp=0;
		temp=speed+relaxation+convection+anticipation+merging_term_on_ramp;
		return(temp);
	}

class off_ramp
```

```
{
      float turning_rate;
      float off_ramp_inflow;
      int off_ramp_index;
      int segment_index;
public:
      off_ramp();
      void initial_off_ramp_rate(float );
      void initial_off_ramp_inflow(float );
      void initial_ramp_location(int , int );
      int get_off_ramp_location();
      float get_off_ramp_rate();
      float get_off_ramp_inflow();
};
off_ramp::off_ramp()
{}
void off_ramp::initial_off_ramp_rate(float rate)
{
            turning_rate=rate;
}
void off_ramp::initial_off_ramp_inflow(float inflow)
{
            off_ramp_inflow=inflow;
}
void off_ramp::initial_ramp_location(int ramp, int segment)
{
            off_ramp_index=ramp;
            segment_index=segment;
}
int off_ramp::get_off_ramp_location()
{
            return(segment_index);

}
float off_ramp::get_off_ramp_rate()
      {
            return(turning_rate);
      }
float off_ramp::get_off_ramp_inflow()
      {
            return(off_ramp_inflow);
      }
class ramp
{
```

```
    float ramp_demand;
    float ramp_capacity;
    float ramp_num_lane;
    float ramp_queue;
    float ramp_metering_rate;
    float ramp_inflow;
    int ramp_index;
    int segment_index;
public:
    ramp();
    void initial_ramp(float , float );
    void initial_ramp_demand(float );
    void initial_ramp_queue(float );
    void initial_ramp_metering_rate(float );
    void intial_ramp_inflow(float );
    void initial_ramp_location(int , int );
    int get_ramp_location();
    float get_ramp_queue();
    float get_ramp_capacity();
    float get_ramp_demand();
    float get_ramp_num_lane();
    float get_ramp_metering_rate();
    float get_ramp_inflow();
};
ramp::ramp()
{ }
void ramp::initial_ramp(float num_lane, float capacity)
    {
        ramp_num_lane=num_lane;
        ramp_capacity=capacity;
    }
void ramp::initial_ramp_demand(float demand)
    {ramp_demand=demand;}
void ramp::initial_ramp_queue(float queue)
    {ramp_queue=queue;}
void ramp::initial_ramp_metering_rate(float rate)
    {ramp_metering_rate=rate;}
void ramp::intial_ramp_inflow(float inflow)
    {ramp_inflow = inflow;}
void ramp::initial_ramp_location(int ramp, int segment)
    {
        ramp_index=ramp;
        segment_index=segment;
    }
```

```
int ramp::get_ramp_location()
	{
		return(segment_index);

	}
float ramp::get_ramp_queue()
	{return(ramp_queue);}
float ramp::get_ramp_capacity()
	{return(ramp_capacity);}
float ramp::get_ramp_demand()
	{return(ramp_demand);}
float ramp::get_ramp_num_lane()
	{return(ramp_num_lane);}
float ramp::get_ramp_metering_rate()
	{return(ramp_metering_rate);}
float ramp::get_ramp_inflow()
	{return(ramp_inflow);}
class nod
{
	const float T;
	float flow_in;
	float flow_out;
	float next_origin_queue;
	float next_ramp_queue;
public:
	nod();
	//float q_out(segment &);
	//float q_out_destination(segment &);
	float q_out_origin(float , float , segment &);
	float q_out_on_ramp(ramp &, segment &, segment &);
	float q_out_off_ramp(off_ramp &, segment &, segment &);
	float get_flow_in();
	float get_next_origin_queue();
	float get_next_ramp_queue();
};
nod::nod():T(0.00139f)
{}
float nod::q_out_origin(float q_in_demand, float cur_origin_queue, segment &output)
	{
		float q_temp;
		if(output.get_density()<output.get_p_crit())
			q_temp=output.get_q_max()*output.get_num_lane();
		else
	q_temp=output.get_q_max()*output.get_num_lane()*(1-(output.get_density()-output.get_p_crit())/(180-output
```

```
.get_p_crit()));
            flow_in=MIN((q_in_demand+cur_origin_queue/T), q_temp);
            flow_out=flow_in;
            next_origin_queue=cur_origin_queue+T*(q_in_demand-flow_out);
            return(flow_out);
      }
float nod::q_out_on_ramp(ramp &on_ramp, segment &input, segment &output)
      {
            float q_out_input_seg;
            if (output.get_density()>=180)
            q_out_input_seg=0;
            else
            q_out_input_seg=input.get_density()*input.get_speed()*input.get_num_lane();
                        float q_ramp_in, q_temp, p;
            p=1-(output.get_density()-output.get_p_crit())/(180-output.get_p_crit());
            if(output.get_density()<=output.get_p_crit())
      q_temp=on_ramp.get_ramp_capacity()*on_ramp.get_ramp_num_lane()*on_ramp.get_ramp_metering_rate();
            else
                  {
                        if(on_ramp.get_ramp_metering_rate()<p)
      q_temp=on_ramp.get_ramp_capacity()*on_ramp.get_ramp_num_lane()*on_ramp.get_ramp_metering_rate();
                        else
                        q_temp=on_ramp.get_ramp_capacity()*on_ramp.get_ramp_num_lane()*p;
                  }
            q_ramp_in=MIN((on_ramp.get_ramp_demand()+on_ramp.get_ramp_queue()/T), q_temp);
            on_ramp.intial_ramp_inflow(q_ramp_in);
            next_ramp_queue=on_ramp.get_ramp_queue()+T*(on_ramp.get_ramp_demand()-q_ramp_in);
            on_ramp.initial_ramp_queue(next_ramp_queue);
                  flow_in=q_out_input_seg+q_ramp_in;
                  flow_out=flow_in;
            return(flow_out);
      }
float nod::q_out_off_ramp(off_ramp &off_ramp, segment &input, segment &output)
      {
            if (output.get_density()>=180)
            flow_in=0;
            else
            flow_in=input.get_density()*input.get_speed()*input.get_num_lane();
            float q_ramp_in;
            q_ramp_in=flow_in*off_ramp.get_off_ramp_rate();
            off_ramp.initial_off_ramp_inflow(q_ramp_in);
            flow_out=flow_in-off_ramp.get_off_ramp_inflow();
            return(flow_out);
      }
```

```
float nod::get_flow_in()
	{
		return(flow_in);
	}
float nod::get_next_origin_queue()
	{
		return(next_origin_queue);
	}

float nod::get_next_ramp_queue()
	{
		return(next_ramp_queue);
	}
float metanet(int num_seg, segment seg[], const float origin_queue, float origin_demand, int num_link, int
num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float link_para[][10], float
seg_initial[][3], float seg_recovery[][3],ramp on[], off_ramp off[])
{
	nod n1;
	float TTT_N=0;
	float TTT_O=0;
	float TTT_origin=0;
	float queue_penalty=0;
	float origin_queue_temp=origin_queue;
	float ramp_queue[3]={on[0].get_ramp_queue(),on[1].get_ramp_queue(),on[2].get_ramp_queue()};
    for(int j=0; j<sim_step; j++)
	{
		for(int i=0; i<num_seg; i++)
		{
			seg[i].initial(seg_initial[i][0],seg_initial[i][1]);
		}
		for(int i=0; i<num_seg; i++)
		{
			TTT_N = TTT_N + seg[i].get_density()*seg[i].get_lane_L()*seg[i].get_num_lane()*(0.00139f);
		}
		for(int i=0; i<num_on_ramp; i++)
		{
			TTT_O = TTT_O + on[i].get_ramp_queue()*(0.00139f);
		}
		TTT_origin=TTT_origin+origin_queue_temp*(0.00139f);
		for(int i=0; i<num_seg; i++)
		{
				if(seg[i].get_destination())
					seg[i].initial_q_out(0);
				else
```

```
                seg[i].initial_q_out(seg[i+1].get_density());
}
for(int i=0; i<num_seg; i++)
{
      if(seg[i].get_origin())
            {
                  float q_temp = n1.q_out_origin(origin_demand,origin_queue_temp,seg[i]);
                  //cout<<"q_origin_in"<<q_temp<<endl;
                  seg[i].initial_q_in(q_temp);
                  origin_queue_temp=n1.get_next_origin_queue();
            }
      else if(seg[i].get_off_ramp())
            {
                  for(int k=0; k<num_off_ramp; k++)
                  {
                        if(off[k].get_off_ramp_location()==i)
                        {
                              float q_temp = n1.q_out_off_ramp(off[k],seg[i-1],seg[i]);
                              seg[i].initial_q_in(q_temp);
                        }
                  }
            }
      else if(seg[i].get_on_ramp())
            {
                  for(int k=0; k<num_on_ramp; k++)
                  {
                        if(on[k].get_ramp_location()==i)
                        {
                              float q_temp = n1.q_out_on_ramp(on[k],seg[i-1],seg[i]);
                              seg[i].initial_q_in(q_temp);
                        }
                  }
            }
      else
            seg[i].initial_q_in(seg[i-1].get_q_out());
}
for(int i=0; i<num_seg; i++)
{
      seg_initial[i][2]=seg[i].get_q_out();
}

for(int i=0; i<num_seg; i++)
{
      if(seg[i].get_origin())
```

```
                {
                      seg_initial[i][0]=seg[i].next_density();
                      seg_initial[i][1]=seg[i].next_speed_origin(seg[i+1].get_density());
                }
                else if(seg[i].get_destination()&&(seg[i].get_on_ramp()!=1))
                {
                      seg_initial[i][0]=seg[i].next_density();
                      seg_initial[i][1]=seg[i].next_speed_destination(seg[i-1].get_speed());
                }
                else if(seg[i].get_weave())
                {
                      seg_initial[i][0]=seg[i].next_density();
                      seg_initial[i][1]=seg[i].next_speed_weave(seg[i-1].get_speed(), seg[i+1].get_density());
                }
                else if(seg[i].get_on_ramp()&&(seg[i].get_destination()!=1))
                {
                      for(int k=0; k<num_on_ramp; k++)
                      {
                            if(on[k].get_ramp_location()==i)
                            {
                                        seg_initial[i][0]=seg[i].next_density();
                                        seg_initial[i][1]=seg[i].next_speed_on_ramp(seg[i-1].get_speed(),
seg[i+1].get_density(),seg[i-1].get_num_lane(),on[k].get_ramp_inflow());
                            }
                      }
                }
                else if(seg[i].get_on_ramp()&&(seg[i].get_destination()==1))
                {
                      for(int k=0; k<num_on_ramp; k++)
                      {
                            if(on[k].get_ramp_location()==i)
                            {
                                        seg_initial[i][0]=seg[i].next_density();
     seg_initial[i][1]=seg[i].next_speed_on_ramp_destination(seg[i-1].get_speed(),
seg[i-1].get_num_lane(),on[k].get_ramp_inflow());
                            }
                      }
                }
                else
                {
                      seg_initial[i][0]=seg[i].next_density();
                      seg_initial[i][1]=seg[i].next_speed(seg[i-1].get_speed(), seg[i+1].get_density());
                }
          }
```

```
	}
	on[0].initial_ramp_queue(ramp_queue[0]);
	on[1].initial_ramp_queue(ramp_queue[1]);
	on[2].initial_ramp_queue(ramp_queue[2]);
	for(int i=0; i<13; i++)
	{
		seg_initial[i][0]=seg_recovery[i][0];
		seg_initial[i][1]=seg_recovery[i][1];
		seg_initial[i][2]=seg_recovery[i][2];
	}
	return(TTT_N+TTT_O+TTT_origin);
}
float metanet_revised(float *temp_ramp_queue, int num_seg, segment seg[], float *origin_queue, float origin_demand,
int num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float
link_para[][10], float seg_initial[][3], ramp on[], off_ramp off[])
{
	nod n1;
	float TTT_N=0;
	float TTT_O=0;
	float TTT_origin=0;
	float queue_penalty=0;
	 for(int j=0; j<sim_step; j++)
	{
		for(int i=0; i<num_seg; i++)
		{
			seg[i].initial(seg_initial[i][0],seg_initial[i][1]);
		}
		for(int i=0; i<num_seg; i++)
		{
			TTT_N = TTT_N + seg[i].get_density()*seg[i].get_lane_L()*seg[i].get_num_lane()*(0.00139f);
		}
		for(int i=0; i<num_on_ramp; i++)
		{
			TTT_O = TTT_O + on[i].get_ramp_queue()*(0.00139f);
		}
		TTT_origin=TTT_origin+*origin_queue*(0.00139f);
		for(int i=0; i<num_seg; i++)
		{
			if(seg[i].get_destination())
				seg[i].initial_q_out(0);
			else
				seg[i].initial_q_out(seg[i+1].get_density());
		}
		for(int i=0; i<num_seg; i++)
```

```
{
	if(seg[i].get_origin())
		{
			float q_temp = n1.q_out_origin(origin_demand,*origin_queue,seg[i]);
			seg[i].initial_q_in(q_temp);
			*origin_queue=n1.get_next_origin_queue();
		}
	else if(seg[i].get_off_ramp())
		{
			for(int k=0; k<num_off_ramp; k++)
			{
				if(off[k].get_off_ramp_location()==i)
				{
					float q_temp = n1.q_out_off_ramp(off[k],seg[i-1],seg[i]);
					seg[i].initial_q_in(q_temp);
				}
			}
		}
	else if(seg[i].get_on_ramp())
		{
			for(int k=0; k<num_on_ramp; k++)
			{
				if(on[k].get_ramp_location()==i)
				{
					float q_temp = n1.q_out_on_ramp(on[k],seg[i-1],seg[i]);
					seg[i].initial_q_in(q_temp);
					if(j==sim_step-1)
					temp_ramp_queue[k]=on[k].get_ramp_queue();
				}
			}
		}
	else
		seg[i].initial_q_in(seg[i-1].get_q_out());
}
for(int i=0; i<num_seg; i++)
{
	seg_initial[i][2]=seg[i].get_q_out();
}
for(int i=0; i<num_seg; i++)
	{
		if(seg[i].get_origin())
		{
			seg_initial[i][0]=seg[i].next_density();
			seg_initial[i][1]=seg[i].next_speed_origin(seg[i+1].get_density());
```

```
                        }
                        else if(seg[i].get_destination()&&(seg[i].get_on_ramp()!=1))
                        {
                              seg_initial[i][0]=seg[i].next_density();
                              seg_initial[i][1]=seg[i].next_speed_destination(seg[i-1].get_speed());
                        }
                        else if(seg[i].get_weave())
                        {
                              seg_initial[i][0]=seg[i].next_density();
                              seg_initial[i][1]=seg[i].next_speed_weave(seg[i-1].get_speed(),
seg[i+1].get_density());
                        }
                        else if(seg[i].get_on_ramp()&&(seg[i].get_destination()!=1))
                        {
                              for(int k=0; k<num_on_ramp; k++)
                              {
                                    if(on[k].get_ramp_location()==i)
                                    {
                                                seg_initial[i][0]=seg[i].next_density();
                                                seg_initial[i][1]=seg[i].next_speed_on_ramp(seg[i-1].get_speed(),
seg[i+1].get_density(),seg[i-1].get_num_lane(),on[k].get_ramp_inflow());
                                    }
                              }
                        }
                        else if(seg[i].get_on_ramp()&&(seg[i].get_destination()==1))
                        {
                              for(int k=0; k<num_on_ramp; k++)
                              {
                                    if(on[k].get_ramp_location()==i)
                                    {
                                                seg_initial[i][0]=seg[i].next_density();
                        seg_initial[i][1]=seg[i].next_speed_on_ramp_destination(seg[i-1].get_speed(),
seg[i-1].get_num_lane(),on[k].get_ramp_inflow());
                                    }
                              }
                        }
                        else
                        {
                              seg_initial[i][0]=seg[i].next_density();
                              seg_initial[i][1]=seg[i].next_speed(seg[i-1].get_speed(), seg[i+1].get_density());
                        }
                  }
      }
      return(TTT_N+TTT_O+TTT_origin);
```

```
}
float metanet_dynamic(float *temp_ramp_queue, int num_seg, segment seg[], float *origin_queue, float origin_demand,
int num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float
link_para[][10], float seg_initial[][3], ramp on[], off_ramp off[])
{
	nod n1;
	float TTT_N=0;
	float TTT_O=0;
	float TTT_origin=0;
	float queue_penalty=0;
	for(int j=0; j<sim_step; j++)
	{
		for(int i=0; i<num_seg; i++)
		{
			seg[i].initial(seg_initial[i][0],seg_initial[i][1]);
		}
		for(int i=0; i<num_seg; i++)
		{
			TTT_N = TTT_N + seg[i].get_density()*seg[i].get_lane_L()*seg[i].get_num_lane()*(0.00139f);
		}
		for(int i=0; i<num_on_ramp; i++)
		{
			TTT_O = TTT_O + on[i].get_ramp_queue()*(0.00139f);
		}
		for(int i=0; i<num_on_ramp; i++)
		{
			if(i==0)
	queue_penalty=queue_penalty+MAX(0,on[i].get_ramp_queue()-180)*MAX(0,on[i].get_ramp_queue()-180);
			else if (i==1)
	queue_penalty=queue_penalty+100*MAX(0,on[i].get_ramp_queue()-300)*MAX(0,on[i].get_ramp_queue()-300);
			else
	queue_penalty=queue_penalty+MAX(0,on[i].get_ramp_queue()-100)*MAX(0,on[i].get_ramp_queue()-100);
		}
		TTT_origin=TTT_origin+*origin_queue*(0.00139f);
		for(int i=0; i<num_seg; i++)
		{
				if(seg[i].get_destination())
					seg[i].initial_q_out(0);
				else
					seg[i].initial_q_out(seg[i+1].get_density());
		}
		for(int i=0; i<num_seg; i++)
		{
			if(seg[i].get_origin())
```

```
            {
                  float q_temp = n1.q_out_origin(origin_demand,*origin_queue,seg[i]);
                  //cout<<"q_origin_in"<<q_temp<<endl;
                  seg[i].initial_q_in(q_temp);
                  *origin_queue=n1.get_next_origin_queue();
            }
      else if(seg[i].get_off_ramp())
            {
                  for(int k=0; k<num_off_ramp; k++)
                  {
                        if(off[k].get_off_ramp_location()==i)
                        {
                              float q_temp = n1.q_out_off_ramp(off[k],seg[i-1],seg[i]);
                              seg[i].initial_q_in(q_temp);
                        }
                  }
            }
      else if(seg[i].get_on_ramp())
            {
                  for(int k=0; k<num_on_ramp; k++)
                  {
                        if(on[k].get_ramp_location()==i)
                        {
                              float q_temp = n1.q_out_on_ramp(on[k],seg[i-1],seg[i]);
                              seg[i].initial_q_in(q_temp);
                              if(j==sim_step-1)
                              temp_ramp_queue[k]=on[k].get_ramp_queue();
                        }
                  }
            }
      else
            seg[i].initial_q_in(seg[i-1].get_q_out());
}
for(int i=0; i<num_seg; i++)
{
      seg_initial[i][2]=seg[i].get_q_out();
}
for(int i=0; i<num_seg; i++)
      {
            if(seg[i].get_origin())
            {
                  seg_initial[i][0]=seg[i].next_density();
                  seg_initial[i][1]=seg[i].next_speed_origin(seg[i+1].get_density());
            }
```

```
                        else if(seg[i].get_destination()&&(seg[i].get_on_ramp()!=1))
                        {
                              seg_initial[i][0]=seg[i].next_density();
                              seg_initial[i][1]=seg[i].next_speed_destination(seg[i-1].get_speed());
                        }
                        else if(seg[i].get_weave())
                        {
                              seg_initial[i][0]=seg[i].next_density();
                              seg_initial[i][1]=seg[i].next_speed_weave(seg[i-1].get_speed(),
seg[i+1].get_density());
                        }
                        else if(seg[i].get_on_ramp()&&(seg[i].get_destination()!=1))
                        {
                              for(int k=0; k<num_on_ramp; k++)
                              {
                                    if(on[k].get_ramp_location()==i)
                                    {
                                                seg_initial[i][0]=seg[i].next_density();
                                                seg_initial[i][1]=seg[i].next_speed_on_ramp(seg[i-1].get_speed(),
seg[i+1].get_density(),seg[i-1].get_num_lane(),on[k].get_ramp_inflow());
                                    }
                              }
                        }
                        else if(seg[i].get_on_ramp()&&(seg[i].get_destination()==1))
                        {
                              for(int k=0; k<num_on_ramp; k++)
                              {
                                    if(on[k].get_ramp_location()==i)
                                    {
                                                seg_initial[i][0]=seg[i].next_density();

      seg_initial[i][1]=seg[i].next_speed_on_ramp_destination(seg[i-1].get_speed(),
seg[i-1].get_num_lane(),on[k].get_ramp_inflow());
                                    }
                              }
                        }
                        else
                        {
                              seg_initial[i][0]=seg[i].next_density();
                              seg_initial[i][1]=seg[i].next_speed(seg[i-1].get_speed(), seg[i+1].get_density());
                        }
                  }
      }
return(TTT_N+TTT_O+TTT_origin+queue_penalty);
```

```
}
```

D.2. Classes Defined for Dynamic Programming

```
class state
{
     float cur_main_queue;float cur_r1_queue;
     float cur_r2_queue;
     float cur_r3_queue;
     float cur_state[13][3];
     double rm[3];
     int state_index;
     float next_main_queue;
     float next_r1_queue;
     float next_r2_queue;
     float next_r3_queue;
     float next_state[13][3];
     public:
     state();
     void initial_state_index(int );
     void initial_rm(double drm[] );
     double get_rm1();
     double get_rm2();
     double get_rm3();
     float get_next_main_queue();
     float get_next_r1_queue();
     float get_next_r2_queue();
     float get_next_r3_queue();
     void get_next_state(float n_state[][3]);
     void update_cur_state(state&);
     void initial_cur_state(float , float , float , float , float initial_state[][3]);
     float run_to_next_state(state& ,int, segment seg[],float , int , int , int , float link_map[][2],
int seg_map[][5], float link_para[][10],  ramp on[], off_ramp off[]);
     void print_state();
};
state::state(){}
void state::initial_state_index(int index)
{    state_index=index;}
void state::initial_rm(double drm[])
{
     for(int i=0; i<3; i++)
     {
          rm[i]=drm[i];
     }
}
double state::get_rm1()
```

```
{ return(rm[0]);}
double state::get_rm2()
{ return(rm[1]);}
double state::get_rm3()
{ return(rm[2]);}
float state::get_next_main_queue()
{ return(next_main_queue);}
float state::get_next_r1_queue()
{ return(next_r1_queue);}
float state::get_next_r2_queue()
{ return(next_r2_queue);}
float state::get_next_r3_queue()
{ return(next_r3_queue);}
void state::get_next_state(float n_state[][3])
{
     for(int i=0; i<13; i++)
     {
          n_state[i][0]=next_state[i][0];
          n_state[i][1]=next_state[i][1];
          n_state[i][2]=next_state[i][2];;
     }
}
void state::update_cur_state(state &last_state)
{
     cur_main_queue =  last_state.get_next_main_queue();
     cur_r1_queue= last_state.get_next_r1_queue();
     cur_r2_queue= last_state.get_next_r2_queue();
     cur_r3_queue= last_state.get_next_r3_queue();
     //float n_state[13][3];
     //last_state.get_next_state(n_state);
     for(int i=0; i<13; i++)
     {
          cur_state[i][0]=last_state.next_state[i][0];
          cur_state[i][1]=last_state.next_state[i][1];
          cur_state[i][2]=last_state.next_state[i][2];
     }
}
void state::initial_cur_state(float main_queue, float r1_queue, float r2_queue, float r3_queue, float
initial_state[][3])
{
     cur_main_queue =  main_queue;
     cur_r1_queue= r1_queue;
     cur_r2_queue= r2_queue;
     cur_r3_queue= r3_queue;
```

```
	for(int i=0; i<13; i++)
	{
		cur_state[i][0]=initial_state[i][0];
		cur_state[i][1]=initial_state[i][1];
		cur_state[i][2]=initial_state[i][2];
	}
}
float state::run_to_next_state(state &n_state,int num_seg, segment seg[],float origin_demand, int num_link,
int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float link_para[][10],
ramp on[], off_ramp off[])
{
	float temp_ramp_queue[3];
	next_main_queue=cur_main_queue;
	float *main=&next_main_queue;
	for(int i=0; i<13; i++)
	{
		next_state[i][0]=cur_state[i][0];
		next_state[i][1]=cur_state[i][1];
		next_state[i][2]=cur_state[i][2];
	}
	on[0].initial_ramp_metering_rate(float(n_state.get_rm1()));
	on[1].initial_ramp_metering_rate(float(n_state.get_rm2()));
	on[2].initial_ramp_metering_rate(float(n_state.get_rm3()));
	on[0].initial_ramp_queue(cur_r1_queue);
	on[1].initial_ramp_queue(cur_r2_queue);
	on[2].initial_ramp_queue(cur_r3_queue);
	float TTS = metanet_dynamic(temp_ramp_queue,num_seg, seg, main, origin_demand, num_link, num_on_ramp,
num_off_ramp, sim_step, link_map, seg_map, link_para,  next_state, on, off);
	next_r1_queue= temp_ramp_queue[0];
	next_r2_queue= temp_ramp_queue[1];
	next_r3_queue= temp_ramp_queue[2];
	//cout<<"next_main_queue"<<next_main_queue<<endl;
	return(TTS);
}
class stage
{
	int stage_index;
	float range[3][2];
	int num_nods;
	float **boundary/*[3][18]*/;
	double **drm/*[5832][3]*/;
	float **tran_map/*[5832][2]*/;
	state *s/*[5832]*/;
public:
```

```
	stage();
	void clear_stage();
	void initial_stage(int num);
	void initial_stage_index(int );
	void initial_range(float temp_range[][2]);
	void initial_states_index();
	void update_states_rm();
	void update_boundary();
	void update_combinations();
	void update_range(float map[]);
	void initial_stage(float , float , float , float , float initial_state[][3]);
	void run_from_initial_stage(stage &next_stage,int num_seg, segment seg[],float origin_demand, int num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float link_para[][10],  ramp on[], off_ramp off[]);
	void run_to_next_stage(stage &next_stage,int num_seg, segment seg[],float origin_demand, int num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float link_para[][10], ramp on[], off_ramp off[]);
	void testing(stage &next_stage, int i);
	void run_to_next_stage_revised(stage &next_stage,int num_seg, segment seg[],float origin_demand, int num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float link_para[][10],  ramp on[], off_ramp off[]);
    void run_to_next_stage_revised_31min(stage &next_stage,int num_seg, segment seg[],float origin_demand, int num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float link_para[][10],  ramp on[], off_ramp off[]);
	friend void mapping_path(stage group[], float map[], float solution[][3]);
	void print_rm();
	void print_tran_map();
	void print_stats_rm();
	void print_state_info(int );
};
stage::stage()
{}
void stage::initial_stage(int num)
{
	num_nods=num;
	boundary= new float*[3];
	for(int i=0; i<3; i++)
	{boundary[i]=new float[num_nods];}
	drm= new double*[num_nods*num_nods*num_nods];
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{drm[i]=new double[3];}
	tran_map= new float*[num_nods*num_nods*num_nods];
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{tran_map[i]=new float[2];}
```

```
	s= new state[num_nods*num_nods*num_nods];
}
void stage::clear_stage()
{
	for(int i=0; i<3; i++)
	{delete[] boundary[i];}
	delete[] boundary;
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{delete[] drm[i];}
	delete[] drm;
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{delete[] tran_map[i];}
	delete[] tran_map;
	delete[] s;
}
void stage::initial_stage_index(int index)
{ stage_index=index;}
void stage::initial_range(float temp_range[][2])
{
	for(int i=0; i<3; i++)
	{
		range[i][0]=temp_range[i][0];
		range[i][1]=temp_range[i][1];
	}
}
void stage::initial_states_index()
{
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{
		s[i].initial_state_index(i);
	}
}
void stage::update_range(float map[])
{
	int selected_state=int(map[stage_index-1]);
	for(int i=0; i<3; i++)
		for(int j=0; j<num_nods; j++)
		{
			if(drm[selected_state][i]==boundary[i][j])
			{
				if(j==0)
					{
						range[i][0]=boundary[i][0]-0.05f;
						range[i][1]=boundary[i][0]+0.05f;
```

```
                    }
                else if(j==(num_nods-1))
                    {
                        range[i][0]=boundary[i][num_nods-1]-0.05f;
                        range[i][1]=boundary[i][num_nods-1]+0.05f;
                    }
                else
                    {
    range[i][0]=boundary[i][j]-0.05f;//(boundary[i][num_nods-1]-boundary[i][0])/(num_nods-1);
    range[i][1]=boundary[i][j]+0.05f;//(boundary[i][num_nods-1]-boundary[i][0])/(num_nods-1);
                }
            }
    }
}
void stage::update_boundary()
{
    for(int i=0; i<3; i++)
    {
        for(int j=0; j<num_nods; j++)
        {
            if(j==0)
            boundary[i][j]=range[i][0];
            else
            boundary[i][j]=boundary[i][j-1]+(range[i][1]-range[i][0])/(num_nods-1);
        }
    }
}
void stage::update_combinations()
{
        for(int j=0; j<num_nods*num_nods*num_nods; j++)
    {
        int n1, n2, n3;
        n1=int(j/(num_nods*num_nods));
        if(j<num_nods*num_nods)
        n2=int(j/num_nods);
        else
        n2=(int)((j%(num_nods*num_nods))/num_nods);
        n3=j%num_nods;
        drm[j][0]=boundary[0][n1];
        drm[j][1]=boundary[1][n2];
        drm[j][2]=boundary[2][n3];
    }
}
void stage::update_states_rm()
```

```
{
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{
		s[i].initial_rm(drm[i]);
	}
}
void stage::initial_stage(float main_queue, float r1_queue, float r2_queue, float r3_queue, float
initial_state[][3])
{
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{
		//s[i].initial_state_index(i);
		s[i].initial_cur_state(main_queue, r1_queue, r2_queue, r3_queue, initial_state);
	}
}
void stage::run_from_initial_stage(stage &next_stage,int num_seg, segment seg[],float origin_demand, int
num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float
link_para[][10],  ramp on[], off_ramp off[])
{
	float tts=0;
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{
		tts=s[i].run_to_next_state(next_stage.s[i], num_seg, seg, origin_demand, num_link, num_on_ramp,
num_off_ramp, sim_step, link_map, seg_map, link_para, on, off);
		next_stage.s[i].update_cur_state(s[i]);
		next_stage.tran_map[i][1]=tts;
	}
}

void stage::run_to_next_stage(stage &next_stage,int num_seg, segment seg[],float origin_demand, int
num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float
link_para[][10],  ramp on[], off_ramp off[])
{
	float *vector=new float [num_nods*num_nods*num_nods];
	float min_obj;
	int pos_state;
	for(int i=0; i<num_nods*num_nods*num_nods; i++)
	{
		for(int j=0; j<num_nods*num_nods*num_nods; j++)
		{
			vector[j]=tran_map[j][1]+s[j].run_to_next_state(next_stage.s[i], num_seg, seg,
origin_demand, num_link, num_on_ramp, num_off_ramp, sim_step, link_map, seg_map, link_para, on, off);
		}
		min_obj = vector[0];
```

```
            pos_state = 0;
            for(int k=1; k<num_nods*num_nods*num_nods; k++)
            {
                  if(vector[k]<min_obj)
                  {min_obj= vector[k];
                   pos_state=k;}
                  }
            tran_map[i][0]=float(pos_state);
            next_stage.tran_map[i][1]=min_obj;
            next_stage.s[i].update_cur_state(s[pos_state]);
      }
      delete [] vector;
}
void stage::run_to_next_stage_revised(stage &next_stage, int num_seg, segment seg[],float origin_demand,
int num_link, int num_on_ramp, int num_off_ramp, int sim_step, float link_map[][2], int seg_map[][5], float
link_para[][10],  ramp on[], off_ramp off[])
{
      float min_obj;
      int pos_state;
      for(int i=0; i<num_nods*num_nods*num_nods; i++)
      {
            int decision1, decision2, decision3;
            if((next_stage.s[i].get_rm1()>0.09f)&&(next_stage.s[i].get_rm1()<0.11f))
            {
                  decision1=2;
            }
            else if((next_stage.s[i].get_rm1()>0.89f)&&(next_stage.s[i].get_rm1()<0.91f))
            {
                  decision1=2;
            }
            else
            {
                  decision1=3;
            }
            if((next_stage.s[i].get_rm2()>0.09f)&&(next_stage.s[i].get_rm2()<0.11f))
            {
                  decision2=2;
            }
            else if((next_stage.s[i].get_rm2()>0.89f)&&(next_stage.s[i].get_rm2()<0.91f))
            {
                  decision2=2;
            }
            else
            {
```

```
      decision2=3;
}
if((next_stage.s[i].get_rm3()>0.09f)&&(next_stage.s[i].get_rm3()<0.11f))
{
      decision3=2;
}
else if((next_stage.s[i].get_rm3()>0.89f)&&(next_stage.s[i].get_rm3()<0.91f))
{
      decision3=2;
}
else
{
      decision3=3;
      //cout<<"ok"<<endl;
}
float *vector=new float [decision1*decision2*decision3];
int *index=new int [decision1*decision2*decision3];
int temp_index[27];
int m=0;
temp_index[0]=i;
temp_index[1]=i+1;
temp_index[2]=i-1;
temp_index[3]=i+9;
temp_index[4]=i+9+1;
temp_index[5]=i+9-1;
temp_index[6]=i-9;
temp_index[7]=i-9+1;
temp_index[8]=i-9-1;
temp_index[9]=i+81;
temp_index[10]=i+81+1;
temp_index[11]=i+81-1;
temp_index[12]=i+81+9;
temp_index[13]=i+81+9+1;
temp_index[14]=i+81+9-1;
temp_index[15]=i+81-9;
temp_index[16]=i+81-9+1;
temp_index[17]=i+81-9-1;
temp_index[18]=i-81;
temp_index[19]=i-81+1;
temp_index[20]=i-81-1;
temp_index[21]=i-81+9;
temp_index[22]=i-81+9+1;
temp_index[23]=i-81+9-1;
temp_index[24]=i-81-9;
```

```
temp_index[25]=i-81-9+1;
temp_index[26]=i-81-9-1;
      if((next_stage.s[i].get_rm1()>0.09f)&&(next_stage.s[i].get_rm1()<0.11f))
      {
            temp_index[18]=-1;
            temp_index[19]=-1;
            temp_index[20]=-1;
            temp_index[21]=-1;
            temp_index[22]=-1;
            temp_index[23]=-1;
            temp_index[24]=-1;
            temp_index[25]=-1;
            temp_index[26]=-1;
      }
      if((next_stage.s[i].get_rm1()>0.89f)&&(next_stage.s[i].get_rm1()<0.91f))
      {
            temp_index[9]=-1;
            temp_index[10]=-1;
            temp_index[11]=-1;
            temp_index[12]=-1;
            temp_index[13]=-1;
            temp_index[14]=-1;
            temp_index[15]=-1;
            temp_index[16]=-1;
            temp_index[17]=-1;
      }
      if((next_stage.s[i].get_rm2()>0.09f)&&(next_stage.s[i].get_rm2()<0.11f))
      {
            temp_index[6]=-1;
            temp_index[7]=-1;
            temp_index[8]=-1;
            temp_index[15]=-1;
            temp_index[16]=-1;
            temp_index[17]=-1;
            temp_index[24]=-1;
            temp_index[25]=-1;
            temp_index[26]=-1;
      }
      if((next_stage.s[i].get_rm2()>0.89f)&&(next_stage.s[i].get_rm2()<0.91f))
      {
            temp_index[3]=-1;
            temp_index[4]=-1;
            temp_index[5]=-1;
            temp_index[12]=-1;
```

```
                temp_index[13]=-1;
                temp_index[14]=-1;
                temp_index[21]=-1;
                temp_index[22]=-1;
                temp_index[23]=-1;
            }
            if((next_stage.s[i].get_rm3()>0.09f)&&(next_stage.s[i].get_rm3()<0.11f))
            {
                temp_index[2]=-1;
                temp_index[5]=-1;
                temp_index[8]=-1;
                temp_index[11]=-1;
                temp_index[14]=-1;
                temp_index[17]=-1;
                temp_index[20]=-1;
                temp_index[23]=-1;
                temp_index[26]=-1;
            }
            if((next_stage.s[i].get_rm3()>0.89f)&&(next_stage.s[i].get_rm3()<0.91f))
            {
                temp_index[1]=-1;
                temp_index[4]=-1;
                temp_index[7]=-1;
                temp_index[10]=-1;
                temp_index[13]=-1;
                temp_index[16]=-1;
                temp_index[19]=-1;
                temp_index[22]=-1;
                temp_index[25]=-1;
            }
        for(int l=0; l<27; l++)
        {
            if(temp_index[l]!=-1)
            {
                index[m]=temp_index[l];
                m=m+1;
            }
        }
        for(int j=0; j<decision1*decision2*decision3; j++)
        {
            vector[j]=tran_map[index[j]][1]+s[index[j]].run_to_next_state(next_stage.s[i], num_seg,
seg, origin_demand, num_link, num_on_ramp, num_off_ramp, sim_step, link_map, seg_map, link_para, on, off);
        }
        min_obj = vector[0];
```

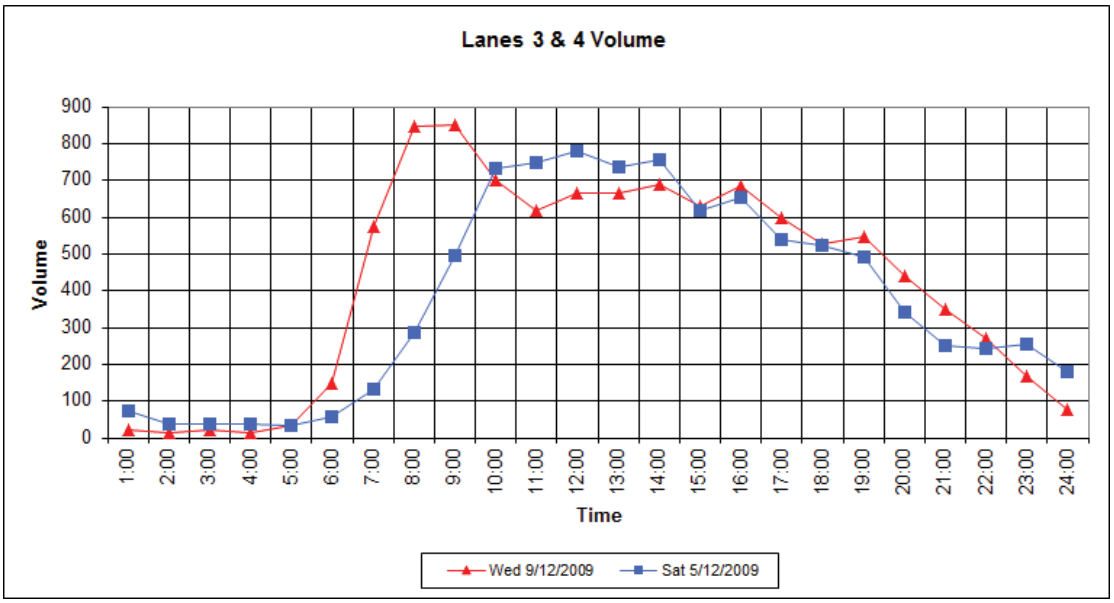
```
            pos_state = index[0];
            for(int k=1; k<decision1*decision2*decision3; k++)
            {
                  if(vector[k]<min_obj)
                  {min_obj= vector[k];
                   pos_state=index[k];}
            }
            tran_map[i][0]=float(pos_state);
            next_stage.tran_map[i][1]=min_obj;
            next_stage.s[i].update_cur_state(s[pos_state]);
            delete [] vector;
            delete [] index;
      }
}
void mapping_path(stage group[],float map[],float solution[][3])
{
      float min_obj = group[5].tran_map[0][1];
      int pos_state = 0;
      int num=group[0].num_nods;
      for(int k=1; k<num*num*num; k++)
            {
                  if(group[5].tran_map[k][1]<min_obj)
                  {
                        min_obj= group[5].tran_map[k][1];
                        pos_state=k;
                  }
            }
      map[5]=min_obj;
      for(int i=4; i>=0; i=i-1)
      {
            map[i]=float(pos_state);
            pos_state=int(group[i].tran_map[pos_state][0]);
      }
      for(int i=0; i<6; i++)
      {cout<<"map["<<i<<"]"<<map[i]<<endl;}
      for(int i=0; i<5; i++)
      {
            pos_state=int(map[i]);
            solution[i][0]=float(group[i+1].drm[pos_state][0]);
            solution[i][1]=float(group[i+1].drm[pos_state][1]);
            solution[i][2]=float(group[i+1].drm[pos_state][2]);
      }
}
```

APPENDIX E: TRAFFIC DATA FROM NZTA

E.1. Traffic Counts from On-ramps

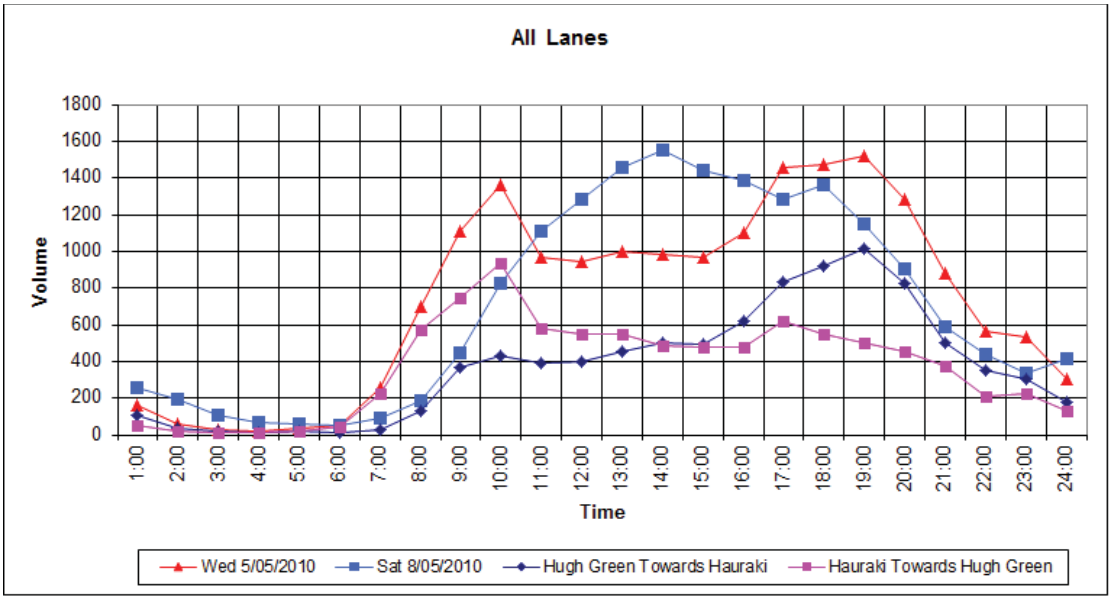
The traffic demands used in this thesis were adapted from the traffic counts for NZTA, which were collected in working days from 6:00am to 10:00am or from 7:00am to 11:00am. The following graphs show partial traffic volumes from on-ramps at the study site.

Tristram Ave



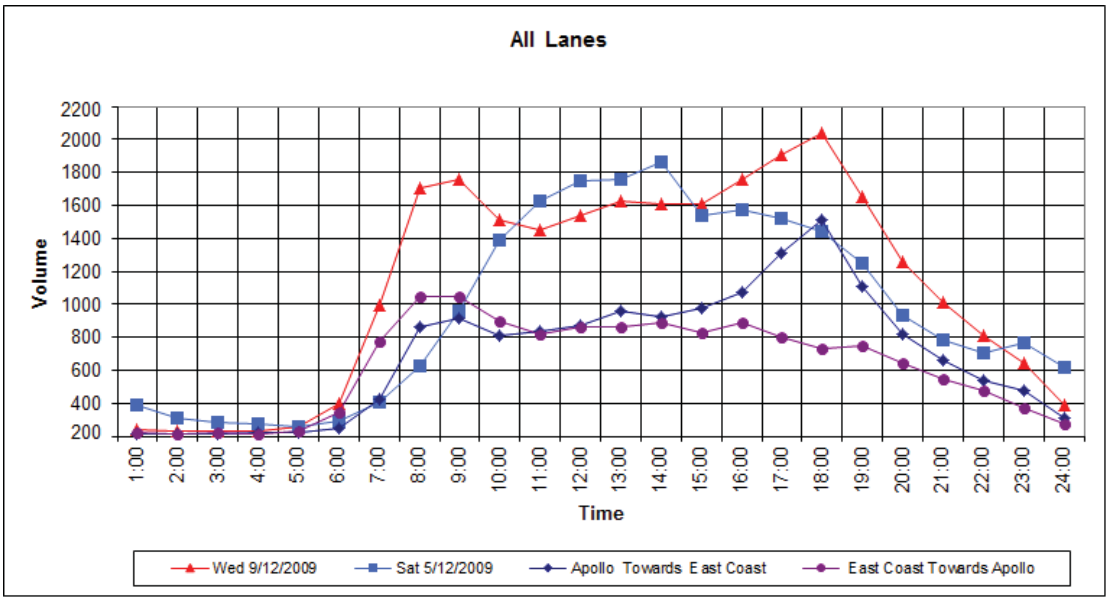
Summary For All Lanes				
Total Volume For Week	123387	Weekday AM Average (6-10am)	823	V/Hr/L
Average Daily Volume (7 Days)	17627	Weekday Midday Average (10am-3pm)	1021	V/Hr/L
Average Daily Volume (Mon - Fri)	17612	Weekday PM Average (3-9pm)	1235	V/Hr/L

Greville Road



Summary For All Lanes				
Total Volume For Week	116930	Weekday AM Average (6-10am)	844	V/Hr/L
Average Daily Volume (7 Days)	16704	Weekday Midday Average (10am-3pm)	950	V/Hr/L
Average Daily Volume (Mon - Fri)	16992	Weekday PM Average (3-9pm)	1228	V/Hr/L

Constellation Drive



Summary For All Lanes				
Total Volume For Week	140432	Weekday AM Average (6-10am)	976	V/Hr/L
Average Daily Volume (7 Days)	20062	Weekday Midday Average (10am-3pm)	1124	V/Hr/L
Average Daily Volume (Mon - Fri)	21200	Weekday PM Average (3-9pm)	1122	V/Hr/L

E.2. Correspondence to Request Traffic Data from NZTA

Dear Kevan,

This is Aaron from Massey University who is writing again to request for your kind support. It has been a while since I last wrote to request for your kind help. The traffic data you provided had played an essential role in our research over the last few months. I attached the paper we are currently working on with this email, and hopefully you may be interested in reading it and enlightening us with your expertise.

This request is made for the traffic data in 2010, during the morning peak period from 6:00am to 10:00am in any successive five working days of the year (the busiest month would be the best). The traffic information will be presented as traffic counts and speed (if possible) in the 15-min time slice.

The study locations remain the same and are listed as follows.

Site Ref: 01N30412 Description: SH1 Greville Rd On Ramp SB (N9)
Site Ref: 01N60412 Description: SH1 Greville Rd Off Ramp SB
Site Ref: 01N10412 Description: SH1 Greville Rd Interchange SB (N9)
Site Ref: 01N30414 Description: SH1 Upper Harbour Hwy On Ramp SB (N8)
Site Ref: 01N60414 Description: SH1 Upper Harbour Hwy Off Ramp SB
Site Ref: 01N10414 Description: SH1 Upper Harbour Hwy Interchange SB (N8)
Site Ref: 01N30417 Description: SH1 Tristram Ave On Ramp SB (N7)
Site Ref: 01N60417 Description: SH1 Tristram Ave Off Ramp SB
Site Ref: 01N10417 Description: SH1 Tristram Ave Interchange SB (N7)

Thanks again for your continued support, and I am looking forward to hearing from you.

Best regards

Aaron

PS: Since my last email account is not working recently, please reply me by this email (batiyu@msn.com)

Hi Aaron

Please find attached data for 2010 for the nine sites you requested.

The 01N60xxx counts are non-continuous, but the rest contain plenty of data.

I have attached hourly counts with daily totals for the whole of 2010 so that you may pick the weeks that you prefer.

Regards

Kevan

Kevan Fleckney
Project Manager (ATMS & ITS)
DDI 09 928 8718
M 021 244 2559
E kevan.fleckney@nzta.govt.nz