Stochastic Dynamic Model of Urban Traffic and Optimum Management of Its Flow and Congestion

by

Shi'an Wang

Thesis submitted to the
Faculty of Engineering, Graduate Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

There are a lot more roads being built periodically in most of the countries with the advancement of modern society. In order to promote the overall traffic flow quality within different cities, city traffic management has been playing a more and more essential role during the last few decades. In recent years, a significantly increasing attention has been paid to the management of traffic flow in major cities all over the world.

In this thesis, we develop a stochastic dynamic model for urban traffic along with physical constraints characteristic of intersections equipped with traffic light. We assume that the incoming traffic to each stream in an intersection is amenable to the Poisson random process with variable intensity (mean). We introduce expressions for traffic throughput, congestion as well as operator's waiting time for the typical intersection in a city and hereafter define an appropriate objective functional. Afterwards, we formulate an optimization problem and propose the sequential (or recursive) algorithm based on the principle of optimality (dynamic programming) due to Bellman. The solution if implemented is expected to improve throughput, reduce congestion, and promote driver's satisfaction. Because the dynamic programming method is computationally quite intensive, we consider the scenario that one unit traffic stream stands for a specific number of vehicles which actually depends on the volume of traffic flow through the intersection.

The system is simulated with inputs described by several distinct nonhomogeneous Poisson processes. For example, we apply the typical traffic arrival rate in Canada with morning peak hour at around 7 : 30 AM and afternoon peak hour at around 4 : 30 PM whilst it is also applied with morning rush hour at about 8 : 00 AM and afternoon rush hour at about 6 : 00 PM like in China. In the meanwhile, we also present a group of numerical results for the traffic arrival rates that have shorter morning peak-hour period but longer afternoon rush hour period. This may occasionally happen when there are some social activities or big events in the afternoon. In addition, another series of experiments are carried out to illustrate the feasibility of the proposed dynamic model based on the traffic arrival rates with only one peak-hour throughout the whole day. The system is simulated with a series of experiments and the optimization problem is solved by dynamic programming based on the proposed algorithm which gives us the optimal feedback control law. More specifically, the results show that both the optimal traffic light timing allocated for each stream and the congestion broadcast level (CBL) of each road segment during each time segment are found. Accordingly, the corresponding optimal cost can be found for any given initial condition. It is reasonably believed that this stochastic dynamic model would be potentially applicable for real time adaptive traffic control system.

Acknowledgements

I would like to thank my supervisors, Prof. N.U. Ahmed and Prof. Tet Yeap, for their extraordinary understanding, guidance and support during the Master program.

Prof. N.U. Ahmed's guidance, patience, and especially attitude towards both research and life really motivated me quite a lot during the last two years at the University of Ottawa. I really would like to say sincere thanks to him for leading me to the path of academic research. More importantly, it is his edification that has made me realize that research is a significant life-long journey full of joy and inspiration. In the meanwhile, Prof. Tet Yeap pointed out how the academic research could be associated with real industry so that I might be able to apply what I have learnt to the industry in the future.

I also would like to thank Miss. Tazrin Ahmed and Mr. Xuesong Liu for their help and suggestions during the graduate study.

Last but not least, this research work would not have been possible without the financial support from my supervisor and the University of Ottawa. I would like to say thanks to them again.

Dedication

To my parents, and my sister Man Wang, I wish to express my sincere gratitude for their remarkable understanding, support, and encouragement.

Table of Contents

List of Tables

List of Figures

Nomenclature

J_k	Time interval $(t_k, t_{k+1}]$ from time t_k to t_{k+1}
τ_{cik}	Time allocated for cross stream on Road i during J_k
τ_{sik}	Time allocated for straight stream on Road i during J_k
$x_{1,i}$	Number of vehicles accumulated on stream i of Road 1
$x_{2,i}$	Number of vehicles accumulated on stream i of Road 2
$\gamma_i(t_k)$	Fraction of time allocated for traffic flows during J_k
$\lambda_i(t_k)$	Traffic arrival rate of i-th stream during J_k
$\ell_i(t_k)$	CBL (congestion broadcast level) for i-th stream during J_k
θ_i	Reduced arrival ratio for i-th stream under congestion
β_i	Service capacity for i-th stream
C_i	Lane capacity of i-th road segment
γ	Vector of traffic light timing fractions
ℓ	Vector of CBLs
$J_{TP}(\gamma)$	Total throughput over the whole day
$J_c(\ell)$	Total measured congestion over the whole day
$J_w(\ell)$	Total measured waiting time over the whole day
$\hat{J}_{TP}(t_k)$	Throughput at stage k
$\hat{J}_c(t_k)$	Measured congestion at stage k
$\hat{J}_w(t_k)$	Measured waiting time at stage k
$J(u) = J(\gamma, \ell)$	Objective functional
$L(k, x_k, u_k)$	Running cost at stage k
W_i	The weight given to congestion for i-th stream
V_i	The weight given to waiting time for i-th stream
x_k	State vector of eight traffic streams at stage k
$\hat{\gamma}_k$	Vector of four traffic light timing fractions at stage k
$\hat{\ell}_k$	Vector of eight CBLs at stage k
G	Vector field
Γ	Constraint set of traffic light timing fraction
Λ	Constraint set of CBLs
CBL	Congestion broadcast level

Chapter 1

Introduction

This thesis studies optimum management of urban traffic flow and congestion. We develop a stochastic dynamic model for urban traffic and propose a methodology to solve the corresponding optimization problem.

1.1 Background

With the rapid growth of urban population (also known as urbanization), traffic management has become a problem of significant importance. Although new roads are being constructed periodically, it is still not enough to catch up with the speed of the growth of modern cities. This issue is quite protruding in many densely populated cities, which has a seriously negative impact on the development of those cities. For example, the automobile era of "urban disease" is severely spreading day by day with the rapid development of urban transportation [1], which causes a number of issues, such as traffic congestion, environmental pollution, frequent traffic accidents and enormous social consumption, etc.

A lot of issues related to traffic management have come out to modern cities, especially for bigger ones. For instance, traffic congestion is a prominent problem in Beijing (capital of China) with automobiles increasing by 10% annually and roads being extended by only 2% on average as reported by Zhang [2]. The rapid increase of automobiles seems to be overwhelming in Beijing as people's incomes increase while prices of automobiles decrease. Moreover, housing prices in Beijing are extremely high, especially near the center of the city. As a result, residents who have to move to the suburbs also buy cars as their means of transportation [2]. However, the development of city roads has not caught up with the speedy increase of automobiles. In [3], it was reported that for every one hour of commuting in Beijing, 30 minutes on average were wasted on traffic jams in 2015, which caused the loss of an average of 808 RMB ($127) in time-cost per month due to traffic congestion. This also occurs to one of the world's biggest cities - New York City. For example, congestion throughout New York City is clogging the streets, polluting the air and restricting the economy. It was reported in [4] that people in New York spent 35% extra travel time due to congestion in 2016 which doubled for commuters in the New York

City. As a consequence, the average New York commuter footed a $2,533 bill in terms of the value of fuel, time, freight and business fees. For the city as a whole, the economic loss was a whopping $16.9 billion in the year of 2016 [5].

It is clear that better (optimal) management of urban traffic is of great importance. In order to enhance the advancement of urban cities, great priorities should be given to urban traffic management. Therefore, there has been an increasing interest in research on traffic management and more attention has been given to urban traffic control in recent years. For example, Beijing, bogged down in severe traffic congestion like many other densely populated metropolitan cities around the world, has made a five-year plan (from 2016 to 2020) in an aim to resolve the problems regarding traffic management [6].

It is understandable that the larger the city is, the greater its complexity and the potential for disruptions are, particularly when this complexity is not effectively managed. Urban productivity is highly dependent on the efficiency of its transportation system to move labor, consumers and freight between multiple departures and destinations. The most important transportation problems are often related to urban areas and take place when transportation systems, for a variety of reasons, cannot satisfy the various requirements of urban mobility. There are several notable urban traffic problems, such as traffic congestion, road accidents and safety, and environmental impacts, etc. For example, it was reported that in 2016 there was 46% extra travel time for the public due to congestion in Beijing with an increase of 8% since the year of 2015 [7].

In order to address these practical problems associated with urban traffic management, more and more researches are being carried out on the basis of urban traffic control. Urban traffic control (UTC) is the method of coordinating traffic signals in a network by the use of traffic light timing plans loaded on a central computer. By applying the technique of UTC, it will be more efficient to manage and control urban traffic so as to achieve some certain objectives, such as increasing road safety, reducing exhaust emission and environmental pollution, and promoting the efficiency of traffic movement. With the advancement of modern technology, the intelligent transportation system (ITS) has already come into reality. ITS is the application of sensing, analysis, control and communications technologies to ground transportation in order to improve safety, mobility and efficiency [8]. It includes a wide range of applications that process and share information to reduce congestion, improve traffic management, minimize environmental impact and increase the benefits of transportation to commercial users and the public in general. However, the intelligent transportation system based on UTC should not be just about informing people where the traffic congestion is and how long one may get stuck in the waiting queue [35]. While such information is necessary, it could also allow vehicle operators to reroute in order to reduce traffic jams without reducing much of the traffic flow volume.

1.2 Related Work

In 1985, AL-KHALILI [13] introduced a general criterion for evaluating the performance of urban traffic system, in terms of delay, stopping time, fuel consumption and exhaust

emission rate. He also studied the relation among those performance factors. In the meanwhile, a general control policy was presented by means of hierarchical control theory. It In the late 90s, Spall *et al.* [28] put forward a control strategy without developing the model for traffic dynamics, which indeed eliminated the need for an open-loop model for the optimal traffic-signal timing. The approach was based on a neural network serving as the basis for the control law, with the weight estimation occurring in closed loop mode via the simultaneous perturbation stochastic approximation (SPSA) algorithm. Later on, an important traffic-adaptive control architecture RHODES was proposed by P. Mirchandani and L. Head [21], which decomposed the traffic control problem into several subproblems and predicted traffic flow at appropriate levels of resolution so as to enable pro-active control. The corresponding algorithm and analysis were also presented.

There were some different algorithms proposed for the control of an isolated intersection in the past two decades and they have made certain improvements to the existing adaptive traffic signal control system. For example, Sen and Head [25] proposed a methodology which allowed optimization of a variety of performance indices such as delay, stops and queue lengths. The presented purpose algorithm might be useful for determining optimal phase sequencing of traffic lights. In [26], Shelby identified some specific deficiencies in the arterial adaptive-control algorithms, such as the inability to optimize timing in general multiring configurations, limitations in real-time computational capabilities, etc. Besides, he compared the existing algorithmic techniques and clarified different situations where those algorithms were best suited for extension or reformulation to overcome the identified deficiencies. Yu and Recker [34] developed a traffic control model based on a discrete-time, stationary, Markov decision process. It is interesting that the proposed model incorporates probabilistic forecasts of individual vehicle actuations at downstream inductance loop detectors that are derived from a macroscopic link transfer function. In [17], an algorithm based on adaptive dynamic programming theory was proposed to adjust the signal time plan at an isolated intersection. However, they did not carry out research on dynamic adjustment to the time parameters, definition of the reward and a more flexible phase sequence, etc. Most recently, Chen and Sun [16] introduced a vehicle arrival estimation model so that vehicle arrival time can be approximately predicted based on the upstream detection information. However, this technique requires large amount of information of the upstream traffic flow.

Although researchers have proposed a number of different traffic control strategies, it appears from the literature that there is not much research work having been done to develop the stochastic dynamic model for urban traffic flow. In [24], a basic dynamic model for congestion status of an intersection and its corresponding cost function were reported by Rahman *et al.* without any attempt to optimize traffic flow (maximize traffic throughput and minimize congestion). In order to fully describe the dynamics of urban traffic flow, in this thesis we develop a discrete-time stochastic dynamic model for eight traffic streams in an intersection and carry out the optimization of its flow, congestion and waiting time.

1.3 Maximum Principle and Dynamic Programming

Given a dynamical process and the corresponding performance index, there are basically two ways of solving for the optimal control of the problem, one is the Pontryagin's maximum principle (MP) and the other is Bellman's dynamic programming (DP).

1.3.1 Maximum Principle

The maximum principle of optimal control gives the fundamental necessary conditions for a controlled trajectory to be optimal. It was first proposed in 1956 by a group of mathematicians under the leadership of L.S. Pontryagin, also including V.G. Boltyanskii, R.V. Gamkrelidze, and E.F. Mishchenko, and it is known as the Pontryagin's maximum principle [23]. The maximum principle is often used in optimal control theory to find the best possible control for taking a dynamical system from one state to another, especially in the presence of constraints for the state or control.

The principle was first known as Pontryagin's maximum principle and its proof is historically based on maximizing the Hamiltonian. The initial application of this principle was to maximize the terminal speed of a rocket [10]. However, as it was subsequently mostly used for minimization of a performance index, it has also been referred to as the minimum principle. There are two different proofs of the necessary conditions of optimality of the maximum principle given by Ahmed in [12].

1.3.2 Dynamic Programming

The term dynamic programming was originally used in 1957 by Richard Bellman to describe the process of solving problems where one needs to find the best decisions one after another [14]. Later on, he refined this to the modern meaning, referring specifically to nesting smaller decision problems inside larger decisions, and the field was thereafter recognized by the IEEE as a systems analysis and engineering topic [9]. Bellman's contribution is remembered in the name of the Bellman equation, a central result of dynamic programming which restates an optimization problem in recursive form.

Dynamic programming is both a mathematical optimization method and a computer programming method [9]. In both contexts it refers to simplifying a complicated problem by breaking it down into simpler sub-problems in a recursive manner. While some decision problems cannot be taken apart this way, decisions that span several points in time do often break apart recursively; Bellman called this the "Principle of Optimality". If sub-problems can be nested recursively inside larger problems, so that dynamic programming methods are applicable, then there is a relation between the value of the larger problem and the values of the sub-problems. In the optimization literature this relationship is called the Bellman equation.

The basic idea of dynamic programming is a discrete, multistage optimization problem in the sense that at each of the finite set of times, a decision is chosen from a finite

number of decisions based on some optimization criteria. Most importantly, it can provide feedback control laws, that is, a controller that monitors the system state and accordingly exercises control actions to achieve certain measures of performance. The central theme of dynamic programming is based on a simple intuitive concept called principle of optimality. It basically states the idea that an optimal path has the property that whatever the initial conditions and control variables (choices) over some initial period, the control (or decision variables) chosen over the remaining period must be optimal for the remaining problem, with the state resulting from the early decisions taken to be the initial condition. The proof of the principle of optimality can be found in [12].

1.4 Contribution

The level of congestion at any intersection is determined by the length of queue (waiting for service on each of the streams) in excess of a set congestion broadcast level (CBL). Once the queue length exceeds the setting level, congestion is radio broadcast so as to encourage drivers to take alternate routes. The CBL setting (maximum queue length) for each road segment is also considered as a control variable which can be adjusted according to the status of the traffic flow. For example, the CBL can be set higher to allow more traffic throughput; it can be also set lower to reduce the traffic congestion and operators waiting time. In order to formulate an appropriate optimization problem, we introduce an objective functional that includes traffic throughput, congestion and operators waiting time. Using this objective functional and the traffic flow dynamics, we carry out the optimization using the method of dynamic programming as proposed in our paper [29].

Based on the above philosophy, in this thesis we develop a stochastic dynamic model for urban traffic flow and propose a methodology for its optimum management. First, we consider any typical intersection of two major roads (2-way multiple lanes) and define the dynamics of traffic flow in eight directions, four directions for cross traffic and four for straight traffic. For simplification, it is fairly reasonable to omit the four right-turn traffic streams because they usually do not interfere with the time allocation of traffic light. The incoming traffic to each stream is assumed to be a nonhomogeneous Poisson random process with variable intensity (mean) according to reality.

Each segment of the road, where vehicles line up for service, is finite. As stated above, as soon as the traffic exceeds a specified CBL setting, congestion is declared and radio-broadcast thereby encouraging drivers to choose alternative routes so as to avoid potential traffic jams. This is used as one set of control variables. Another set of control variables is given by the set of fractions of time allocated to each one of the traffic streams during each cycle of the traffic light. The objective is to minimize possible traffic congestion, delay, service time and maximize the throughput.

More specifically, this thesis provides the following contributions:

1. We developed a comprehensive stochastic dynamic model for urban traffic, which has been published as "Dynamic Model of Urban Traffic and Optimum Management of

Its Flow and Congestion." in *Dynamic Systems and Applications* 26 (2017): 575-588 [29].

2. We proposed a performance criterion (objective functional) for the evaluation of traffic flow in the intersection and introduced congestion broadcast level (CBL) for better management of the intersection.

3. We proposed a methodology to optimize urban traffic flow so as to increase the throughput and reduce possible congestion. This work entitled "Optimum Management of Urban Traffic Flow Based on a Stochastic Dynamic Model." is under review of *IEEE Transactions on Intelligent Transportation Systems* [33].

Moreover, related work on traffic control, management and queueing theory can be found in [30] - [32].

1.5 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 reviews several well-known stochastic processes, including Poisson process, Gaussian process and Markov process. In this chapter, the Poisson process is discussed in detail and thereby the nonhomogeneous Poisson process is highlighted since it is this process that is applied to the stochastic dynamic model of urban traffic flow. Moreover, the traffic flow dynamics are introduced for eight traffic streams. In Chapter 3, we formally introduce the traffic flow optimization problem along with an appropriate objective functional that includes traffic throughput, congestion and waiting time involved in an intersection. Hereafter, the compact form of the optimization problem is also introduced in the same chapter. Besides, we make some discussions on the Monte Carlo methods which are generally used for the simulation of a wide range of mathematical and physical systems. In Chapter 4, we apply the dynamic programming technique to the optimization problem formulated in the preceding chapter. Then the sequential (or recursive) algorithm is developed based on the principle of optimality. In Chapter 5, we describe the implementation of the proposed algorithm, and then mainly demonstrate three different scenarios for a series of numerical simulations. Detailed analysis is also made in this chapter to illustrate the efficacy of the proposed stochastic dynamic model. Finally, conclusions are made in Chapter 6 with a summary of future research.

Chapter 2

Stochastic System Model

In this chapter, we introduce the stochastic system model for urban traffic management according to the nonhomogeneous Poisson process. We also present the traffic flow dynamics for eight different streams involved in a typical intersection. The following chapters are based on the preliminaries introduced in this chapter.

2.1 Definition

2.1.1 Indicator Function

Let S denote any logical (or mathematical) statement. Then the indicator function of this statement is defined as follows:

$$I(S) = \begin{cases} 1 & \text{if S is true,} \\ 0 & \text{otherwise.} \end{cases}$$

For different statements such as S_j we use indicators with a subscript such as $I_j, j \in \mathbb{Z}^+$.

2.1.2 Stochastic Processes

Suppose that $(\Omega, \mathcal{F}, P)$ is a probability space, and that $X : \Omega \to R$ is a random variable. This means that Ω is a space, $\mathcal{F}$ is a σ-algebra of subsets of Ω, P is a countably additive, non-negative measure on $(\Omega, \mathcal{F})$ with total mass $P(\Omega) = 1$, and X is a measurable function.

A stochastic process is simply a collection of random variables indexed by time. It will be useful to consider separately the cases of discrete time and continuous time. That is, a discrete time stochastic process $X = \{X_n, n = 0, 1, 2, \cdots\}$ is a countable collection of random variables indexed by the non-negative integers, and a continuous time stochastic process $X = \{X_t, 0 \leq t < \infty\}$ is an uncountable collection of random variables indexed by the non-negative real numbers. In general, we may consider any indexing set $S \subset R$

having infinite cardinality, so that calling $X = \{X_\alpha, \alpha \in S\}$ a stochastic process simply means that X_α is a random variable for each $\alpha \in S$.

There are a number of well-known stochastic processes, such as Gaussian process, Markov process and Poisson process, etc [19]. Gaussian process is a particular kind of stochastic process where observations occur in a continuous domain, such as time and space. In a Gaussian process, every point in some continuous input space is associated with a normally distributed random variable. Moreover, every finite collection of those random variables has a multivariate normal distribution. For example, every finite linear combination of these random variables is normally distributed. The distribution of a Gaussian process is the joint distribution of all these (infinitely many) random variables.

Gaussian processes are very useful in statistical modelling, benefiting from properties inherited from the normal. For instance, if a random process is modelled as a Gaussian process, the distributions of various derived quantities can be obtained explicitly. Such quantities include the average value of the process over a range of times and the error in estimating the average using sample values at a small set of times.

A Markov process is a stochastic process indexed by time. It has the important property that given the present the future is independent of the past, which is called the memoryless property in probability theory and related fields. A Markov chain is a well-known type of Markov process that has either discrete state space or discrete index set (often representing time). It is commonly regarded as a Markov process in either discrete or continuous time with a countable state space.

We use the nonhomogeneous Poisson process to model the traffic streams since the traffic arrival follows Poisson process. The basic Poisson process can be defined as follows: Let $\lambda > 0$ be fixed. The counting process $\{N(t), t \in [0, \infty)\}$ is called a Poisson process with rate λ if all the following conditions hold:

(1) $N(0) = 0$;
(2) $N(t)$ has independent increments;
(3) The number of arrivals in any interval of length $\Delta > 0$ has Poisson($\lambda\Delta$) distribution.

The Poisson process can be regarded as a counting process, which is a stochastic process that represents the random number of points or events up to some time. The number of points of the process that are located in the interval from zero to some given time is a Poisson random variable that depends on that time and some parameter. This process has the natural numbers as its state space and the non-negative numbers as its index set.

If a Poisson process is defined with a constant frequency of occurrence (like λ mentioned before), then the process is called a homogeneous Poisson process. The homogeneous Poisson process can be defined and generalized in different ways. It can be defined such that its index set is the real line, and this stochastic process is also called the stationary Poisson process. If the constant parameter of the Poisson process is replaced with some non-negative integrable function of t, the resulting process is called a nonhomogeneous Poisson process, where the average density of points of the process is no longer constant.

2.2 Stochastic Traffic Model

Traffic flow is a stochastic process. In other words, the number of vehicles crossing any intersection over any given interval of time is a random variable. A reasonable mathematical model can be constructed on the basis of counting process, in particular, the well-known Poisson process. Here in this section we present one such model. Let $(\Omega, \mathcal{F}, \mathcal{F}_{t\geq 0}, P)$ denote a filtered probability space, where Ω is the sample space, $\mathcal{F}$ is the sigma algebra of Borel subsets of the space Ω and $\mathcal{F}_t \subset \mathcal{F}$ is a family of nondecreasing subsigma algebras and P is the probability measure.

A stochastic process $\{\xi_t \equiv p(0,t), t \geq 0\}$, is called a Poisson random process or a counting process giving the number of events over the time period $[0,t]$. More precisely, for each $\omega \in \Omega$, p_w is a set function defined on the sigma algebra of Borel sets in $R_+ \equiv [0,\infty)$ and taking values from the set of nonnegative integers $\mathcal{N} \equiv \{0,1,2,\cdots\}$. For example, the number of cars arriving at the intersection during the time interval $(t_1,t_2]$ giving $p_w(t_1,t_2]$ is a random variable.

The process p is called a homogeneous Poisson process if the values of p over disjoint intervals of time are stochastically independent or equivalently the increments of ξ over disjoint intervals of time are statistically independent and there exists a nonnegative constant λ such that the probability that there are exactly n events over the time period $(t_1,t_2]$ is given by

$$P\{p_w(t_1,t_2] = n\} = P\{\xi_{t_2}(\omega) - \xi_{t_1}(\omega) = n\} = e^{-\lambda(t_2-t_1)}(\lambda(t_2-t_1))^n/n! \tag{2.1}$$

The constant λ is called the mean or average number of events per unit time or the frequency of occurrence of events. Indeed the reader can verify that the expected value of the random variable $p_\omega((t_1,t_2])$ is given by

$$\mathbf{E}\{p_\omega(t_1,t_2]\} = \sum_{n=0}^{\infty} n e^{-\lambda(t_2-t_1)}(\lambda(t_2-t_1))^n/n! = \lambda(t_2-t_1) \tag{2.2}$$

Throughout the rest of the thesis we shall omit the variable ω.

In the study of urban traffic flow, the mean flow rate is not constant. It varies with time. It is large at the rush hour in the morning when people go to work and again in the evening when they return home. These are the two main peak hours and there is also a moderate peak at lunch hour. So we need a nonhomogeneous Poisson process with time varying mean. Let $\xi_t \equiv p(0,t)$ be a nonhomogeneous Poisson process with the mean rate function $\lambda(t), t \geq 0$. Clearly, this is a nonnegative finite real valued function (deterministic). Then the probability that there are exactly n events during the time interval $J \equiv (t_1,t_2]$ is given by

$$P\{p(t_1,t_2] = n\} = P\{\xi_{t_2} - \xi_{t_1} = n\} = \exp\left(-\int_{t_1}^{t_2} \lambda(t)dt\right) \frac{\left(\int_{t_1}^{t_2} \lambda(t)dt\right)^n}{n!} \tag{2.3}$$

In practical applications the function $\lambda(t), t \geq 0$, can be estimated by survey of the traffic at any given major intersection. Let us assume that the function λ is available from midnight

to midnight. Suppose this time interval is partitioned into a finite number of subintervals like $(t_k, t_{k+1}]$, $k = 0, 1, 2, \cdots, N-1$ so that the entire day, denoted by $I \equiv (0, T]$, is given by $I = \cup_{k=0}^{N-1}(t_k, t_{k+1}]$. The function λ is then approximated by step functions in the sense that it is constant on each interval and given by $\lambda(t) = \lambda(t_k) \equiv \lambda_k$, for $t \in (t_k, t_{k+1}], k = 0, 1, 2, \cdots, N-1$. This is quite reasonable if the length of each time interval $(t_{k+1} - t_k)$ is sufficiently small, say, five minutes. Thus, the probability that there are exactly n events during the interval $(t_k, t_{k+1}]$ is given by

$$P\{\xi_{t_{k+1}} - \xi_{t_k} = n\} = \exp\big(-\lambda_k(t_{k+1} - t_k)\big)\frac{(\lambda_k(t_{k+1} - t_k))^n}{n!} \tag{2.4}$$

In the modeling process we will denote the corresponding Poisson process by

$$p((t_k, t_{k+1}], \lambda_k)$$

and rewrite equation (2.4) as

$$P\{p((t_k, t_{k+1}], \lambda_k) = n\} = \exp\big(-\lambda_k(t_{k+1} - t_k)\big)\frac{(\lambda_k(t_{k+1} - t_k))^n}{n!} \tag{2.5}$$

Note that the expected value of $p((t_k, t_{k+1}], \lambda_k)$ is given by

$$\mathbf{E}\{p((t_k, t_{k+1}], \lambda_k)\} = \lambda_k(t_{k+1} - t_k) \tag{2.6}$$

and the second moment is given by

$$\mathbf{E}\{(p((t_k, t_{k+1}], \lambda_k))^2\} = (\lambda_k(t_{k+1} - t_k))^2 + \lambda_k(t_{k+1} - t_k) \tag{2.7}$$

hence the variance is given by

$$\mathbf{E}\bigg(p((t_k, t_{k+1}], \lambda_k) - \lambda_k(t_{k+1} - t_k)\bigg)^2 = \lambda_k(t_{k+1} - t_k) \tag{2.8}$$

2.3 Traffic Flow Dynamics

2.3.1 Preliminaries of Traffic Flow

Now we are prepared to develop a dynamic model of urban traffic flow at an intersection. Clearly there are many different types of intersections, such as T intersection, L intersection, etc. However, in this thesis we consider the typical intersection of any two two-way roads having possibly multiple lanes and let us denote the two roads by $R1$ (road one) and $R2$ (road two). This isolated intersection is shown in Figure 2.1. It is clear that there are 12 streams of traffic flow at the intersection. However, as stated in section 1.2 the four right-turn traffic streams do not interfere with the traffic light timing. Therefore, we will disregard the (minor) flows that can take right turn from their right lane whenever it is assumed to be safe to do so. This will eliminate four simple streams. We are left with eight critically more important flows. These are 4 streams of straight traffic and 4 streams

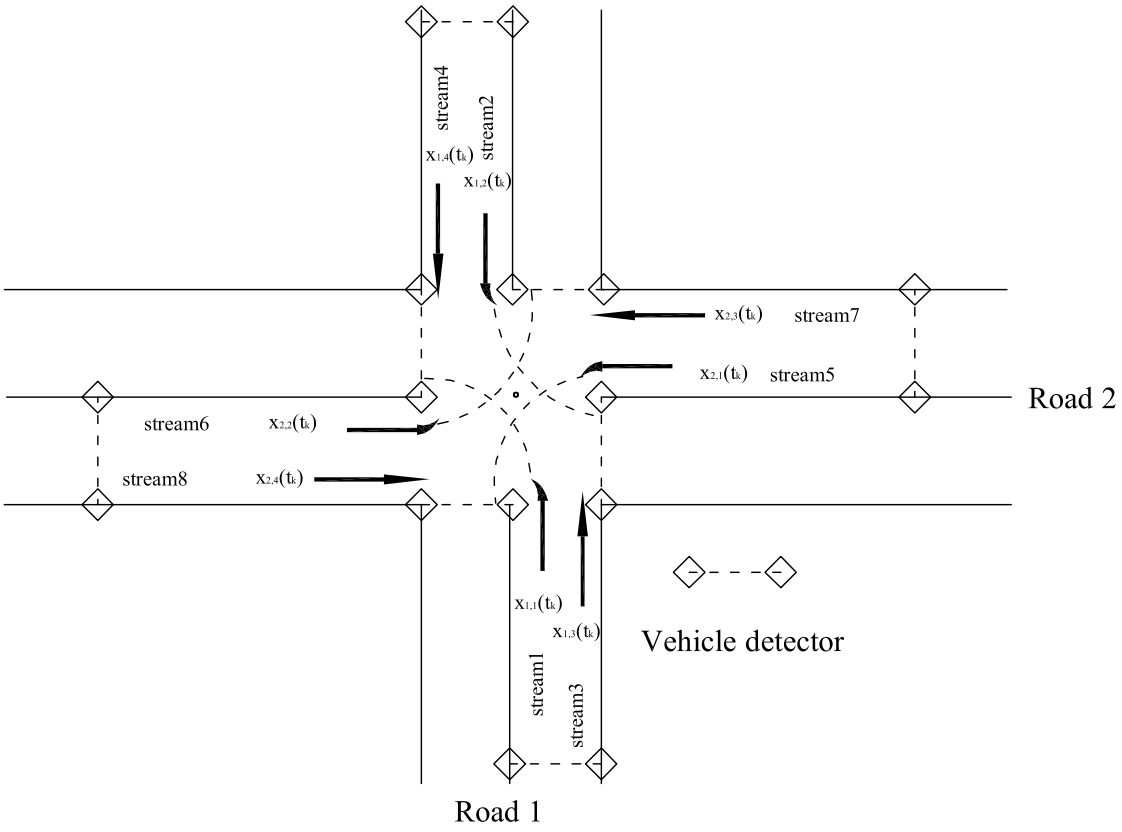


Figure 2.1: A typical isolated intersection.

of cross traffic. Thus, it suffices to consider only these 8 streams of flow and develop a discrete-time evolution of the traffic status at any intersection.

We consider the sequence of time intervals $J_k \equiv (t_k, t_{k+1}]$ starting with $k = 0$, $0 = t_0 < t_1 < t_2 < t_3 < \cdots < t_k < t_{k+1} < \cdots < t_{N-1}$, where N is the total number of time intervals covering the whole day. Each interval of time J_k is also considered as a cycle of traffic light sequence. Each interval is given by the union of 4 disjoint subintervals

$$J_k = \tau_{c1k} \cup \tau_{s1k} \cup \tau_{c2k} \cup \tau_{s2k}$$

where τ_{c1k} is the period of time allocated for cross traffic from $R1$ to $R2$ during the time segment J_k and τ_{s1k} is the period of time allocated for straight traffic on $R1$. Similarly τ_{c2k} and τ_{s2k} denote the time periods allocated for traffic with respect to $R2$ during the time slot J_k. Let $|J_k| \equiv (t_{k+1} - t_k)$ denote the length of one cycle, the time interval J_k, $|\tau_{c1k}| = \gamma_1(t_k)|J_k|$, $|\tau_{s1k}| = \gamma_2(t_k)|J_k|$, $|\tau_{c2k}| = \gamma_3(t_k)|J_k|$, $|\tau_{s2k}| = \gamma_4(t_k)|J_k|$ where $\{\gamma_i(\cdot)\}$ are positive fractions summing to one, that is $\sum \gamma_i(\cdot) = 1$. Clearly these numbers denote the fraction of time allocated to cross and straight flows at the intersection. The fractions $\{\gamma_i, i = 1, 2, 3, 4\}$ are variables that can be adjusted. We shall consider these as one set of control variables to be chosen to optimize the traffic flow. Throughout the rest of the

thesis we use the notation I for the indicator function as introduced before. For any two real numbers $\{a, b\}$ we use the notation

$$a \wedge b \equiv \min\{a, b\}$$

to denote the minimum of the two. Now we are prepared to introduce the traffic flow dynamics.

2.3.2 Complementary Cross Traffic from R1 to R2

The complementary cross traffic from $R1$ to $R2$ in two opposite directions is given by the following pair of equations:

$$\begin{aligned} x_{1,1}(t_{k+1}) &= x_{1,1}(t_k) + I_1(0 \le x_{1,1}(t_k) < \ell_1(t_k))p_1(J_k, \lambda_1(t_k)) \\ &\quad + I_2(\ell_1(t_k) \le x_{1,1}(t_k) < C_1)p_1(J_k, \theta_1\lambda_1(t_k)) - (\beta_1 \wedge x_{1,1}(t_k))\gamma_1(t_k)|J_k| \end{aligned} \quad (2.9)$$

$$\begin{aligned} x_{1,2}(t_{k+1}) &= x_{1,2}(t_k) + I_1(0 \le x_{1,2}(t_k) < \ell_2(t_k))p_2(J_k, \lambda_2(t_k)) \\ &\quad + I_2(\ell_2(t_k) \le x_{1,2}(t_k) < C_2)p_2(J_k, \theta_2\lambda_2(t_k)) - (\beta_2 \wedge x_{1,2}(t_k))\gamma_1(t_k)|J_k| \end{aligned} \quad (2.10)$$

Equation (2.9) represents the cross traffic from $R1$ to $R2$ in one direction and equation (2.10) represents the cross traffic in the opposite direction. The symbol $x_{1,1}(t_{k+1})$ stands for the number of vehicles accumulated on this stream at time t_{k+1}. This is given by the algebraic sum of 4 terms. The first term on the right gives the number of vehicles that was left on the stream from the previous cycle. The second term gives the random number of vehicles arriving during the time period $J_k \equiv (t_k, t_{k+1}]$. This is denoted by the Poisson random variable $p_1(J_k, \lambda_1(t_k))$ (Poisson random number) with the mean arrival rate $\lambda_1(t_k)$ provided that $x_{1,1}(t_k)$ is below the CBL setting $\ell_1(t_k)$. The number $\ell_1(t_k)$ denotes the congestion broadcast level explained further in the sequel. The third component on the right represents the number of vehicles arriving after congestion has been broadcast at a reduced rate $\theta_1\lambda_1(t_k)$ where $0 \le \theta_1 < 1$ provided that the lane capacity C_1 is not exceeded. The last term on the right represents the number of vehicles that leaves the stream during the time period $\gamma_1(t_k)J_k$ at the service capacity β_1. The number β_1 represents the service capacity of this stream which pretty much depends on the number of lanes. Thus the the departure rate is the smaller of the two $\{\beta_1, x_{1,1}(t_k)\}$ denoted by $\beta_1 \wedge x_{1,1}(t_k)$. Equation (2.10) represents the complementary cross traffic from $R1$ to $R2$ in the opposite direction and has identical interpretation as equation (2.9).

2.3.3 Complementary Straight Traffic on R1

The complementary straight traffic on $R1$ in two opposite directions is given by the following pair of equations:

$$\begin{aligned} x_{1,3}(t_{k+1}) &= x_{1,3}(t_k) + I_1(0 \le x_{1,3}(t_k) < \ell_3(t_k))p_3(J_k, \lambda_3(t_k)) \\ &\quad + I_2(\ell_3(t_k) \le x_{1,3}(t_k) < C_3)p_3(J_k, \theta_3\lambda_3(t_k)) - (\beta_3 \wedge x_{1,3}(t_k))\gamma_2(t_k)|J_k| \end{aligned} \quad (2.11)$$

$$x_{1,4}(t_{k+1}) = x_{1,4}(t_k) + I_1(0 \le x_{1,4}(t_k) < \ell_4(t_k))p_4(J_k, \lambda_4(t_k)) \\ + I_2(\ell_4(t_k) \le x_{1,4}(t_k) < C_4)p_4(J_k, \theta_4\lambda_4(t_k)) - (\beta_4 \wedge x_{1,4}(t_k))\gamma_2(t_k)|J_k| \quad (2.12)$$

Considering the complementary straight traffic on $R1$, equations (2.11) and (2.12) represent streams of straight traffic in opposite directions. Considering equation (2.11), the number of vehicles on this stream at time t_{k+1} denoted by $x_{1,3}(t_{k+1})$ is given by the sum of 4 terms. The first term represents the residue traffic from the previous cycle ending at time t_k. The second term represents the new arrivals during the time cycle J_k and it is given by the Poisson random variable $p_3(J_k, \lambda_3(t_k))$ corresponding to the mean intensity $\lambda_3(t_k)$ provided that the CBL setting $\ell_3(t_k)$ has not been reached. The third term represents the number of vehicles which arrive at a reduced rate after the congestion warning has been broadcast given that the road segment capacity C_3 is not reached. The fourth term on the righthand side represents the number of vehicles that left the stream during the time period J_k with departure rate β_3. Similar interpretation is valid for the complementary traffic (traffic in the opposite direction).

Section 2.3.2 and 2.3.3 take care of traffic on and from $R1$ (road one).

2.3.4 Complementary Cross Traffic from R2 to R1

Similarly the cross traffic from $R2$ to $R1$ is given by a set of 2 equations:

$$x_{2,1}(t_{k+1}) = x_{2,1}(t_k) + I_1(0 \le x_{2,1}(t_k) < \ell_5(t_k))p_5(J_k, \lambda_5(t_k)) \\ + I_2(\ell_5(t_k) \le x_{2,1}(t_k) < C_5)p_5(J_k, \theta_5\lambda_5(t_k)) - (\beta_5 \wedge x_{2,1}(t_k))\gamma_3(t_k)|J_k| \quad (2.13)$$

$$x_{2,2}(t_{k+1}) = x_{2,2}(t_k) + I_1(0 \le x_{2,2}(t_k) < \ell_6(t_k))p_6(J_k, \lambda_6(t_k)) \\ + I_2(\ell_6(t_k) \le x_{2,2}(t_k) < C_6)p_6(J_k, \theta_6\lambda_6(t_k)) - (\beta_6 \wedge x_{2,2}(t_k))\gamma_3(t_k)|J_k| \quad (2.14)$$

These equations describe the dynamics of complementary cross traffic from $R2$ to $R1$. Equation (2.13) represents the cross traffic from $R2$ to $R1$ in one direction and equation (2.14) represents the cross traffic in the opposite direction. The symbol $x_{2,1}(t_{k+1})$ stands for the number of vehicles accumulated on this stream at time t_{k+1}. This is given by the algebraic sum of 4 terms. The first term on the right gives the number of vehicles that was left on the stream from the previous cycle. The second term gives the random number of vehicles arriving during the time period $J_k \equiv (t_k, t_{k+1}]$. This is denoted by the Poisson random variable $p_5(J_k, \lambda_5(t_k))$ (Poisson random number) with the mean arrival rate $\lambda_5(t_k)$ provided that $x_{2,1}(t_k)$ is below the CBL setting $\ell_5(t_k)$. The number $\ell_5(t_k)$ denotes the congestion broadcast level explained further in the sequel. The third component on the right represents the number of vehicles arriving after congestion has been broadcast at a reduced rate $\theta_5\lambda_5(t_k)$ where $0 \le \theta_5 < 1$ provided that the lane capacity C_5 is not exceeded. The last term on the right represents the number of vehicles that leaves the stream during the time period $\gamma_3(t_k)J_k$ at the service capacity β_5. The number β_5 represents the service capacity of this stream which pretty much depends on the number of lanes. Thus the the departure rate is the smaller of the two $\{\beta_5, x_{2,1}(t_k)\}$ denoted by $\beta_5 \wedge x_{2,1}(t_k)$. Equation (2.14) represents the complementary cross traffic from $R2$ to $R1$ in the opposite direction and has identical interpretation as equation (2.13).

2.3.5 Complementary Straight Traffic on R2

The straight traffic on $R2$ is given by the following set of 2 equations:

$$x_{2,3}(t_{k+1}) = x_{2,3}(t_k) + I_1(0 \le x_{2,3}(t_k) < \ell_7(t_k))p_7(J_k, \lambda_7(t_k)) + I_2(\ell_7(t_k) \le x_{2,3}(t_k) < C_7)p_7(J_k, \theta_7\lambda_7(t_k)) - (\beta_7 \wedge x_{2,3}(t_k))\gamma_4(t_k)|J_k| \quad (2.15)$$

$$x_{2,4}(t_{k+1}) = x_{2,4}(t_k) + I_1(0 \le x_{2,4}(t_k) < \ell_8(t_k))p_8(J_k, \lambda_8(t_k)) + I_2(\ell_8(t_k) \le x_{2,4}(t_k) < C_8)p_8(J_k, \theta_8\lambda_8(t_k)) - (\beta_8 \wedge x_{2,4}(t_k))\gamma_4(t_k)|J_k| \quad (2.16)$$

These equations describe the dynamics of complementary straight traffic on $R2$. Considering the complementary straight traffic on $R2$, equations (2.15) and (2.16) represent streams of straight traffic in opposite directions. Considering equation (2.15), the number of vehicles on this stream at time t_{k+1} denoted by $x_{2,3}(t_{k+1})$ is given by the sum of 4 terms. The first term represents the residue traffic from the previous cycle ending at time t_k. The second term represents the new arrivals during the time cycle J_k and it is given by the Poisson random variable $p_7(J_k, \lambda_7(t_k))$ corresponding to the mean intensity $\lambda_7(t_k)$ provided that the CBL setting $\ell_7(t_k)$ has not been reached. The third term represents the number of vehicles which arrive at a reduced rate after the congestion warning has been broadcast given that the road segment capacity C_7 is not reached. The fourth term on the righthand side represents the number of vehicles that left the stream during the time period J_k with departure rate β_7. Equation (2.16) represents the complementary traffic on $R2$ in the opposite direction and has identical interpretation as equation (2.15).

Section 2.3.4 and 2.3.5 take care of traffic on and from $R2$ (road two).

2.4 Final Remarks

In this chapter, first we present the definition of indicator function and then introduce the stochastic processes, such as Gaussian process, Markov process and Poisson process. In the meanwhile, we have presented the work done in developing the stochastic dynamic model for urban traffic flow including the traffic dynamics for eight different streams, four for $R1$ and four for $R2$. It is clear that the traffic arrival rates follow nonhomogeneous Poisson random processes in reality and it is those variable intensity (mean) Poisson processes that are used in the later chapters for the optimization of the non-linear, multi-stage urban traffic control problem.

Chapter 3

Objective Functional and Optimization Problem Formulation

In this chapter, we introduce performance criteria of the dynamic system, such as traffic throughput, congestion and waiting time. Hereafter, we propose an appropriate objective functional with physical constraints. We also formulate the optimization problem in its compact form and introduce the Monte Carlo Method for the purpose of numerical simulation.

3.1 Traffic Flow Optimization Problem

The objective is to maximize throughput through the intersection and minimize congestion and waiting time. The lane capacity is fixed. Thus, the throughput depends on the lane capacity and the traffic present and more importantly on the distribution of the fraction of time $\{\gamma_i, i = 1, 2, 3, 4\}$ allocated to each of the streams which is a decision variable. Presently we denote this by the four dimensional vector

$$\gamma \equiv (\gamma_1, \gamma_2, \gamma_3, \gamma_4)' \in R_+^4,\ R_+ = [0, \infty)$$

Then the throughput during the period J_k (at stage k) is given by the following expression,

$$\begin{aligned}\hat{J}_{TP}(\gamma(t_k)) = &\sum_{i=1}^{2}(\beta_i \wedge x_{1,i}(t_k))I_3(t \in \tau_{c1k})\gamma_1(t_k)|J_k| \\ &+ \sum_{i=1}^{2}(\beta_{i+2} \wedge x_{1,i+2}(t_k))I_3(t \in \tau_{s1k})\gamma_2(t_k)|J_k| \\ &+ \sum_{i=1}^{2}(\beta_{i+4} \wedge x_{2,i}(t_k))I_3(t \in \tau_{c2k})\gamma_3(t_k)|J_k| \\ &+ \sum_{i=1}^{2}(\beta_{i+6} \wedge x_{2,i+2}(t_k))I_3(t \in \tau_{s2k})\gamma_4(t_k)|J_k| \end{aligned} \tag{3.1}$$

where the first and the second sum give respectively the number of vehicles served in the complementary cross flow and complementary straight flow from $R1$ during the time interval J_k. The third and the fourth sum represent the number of vehicles served from similar flows with reference to $R2$. Thus, the total throughput over the time period $[t_0, t_N]$ is given by the following expression,

$$J_{TP}(\gamma) \equiv \mathbf{E}\Big\{\sum_{k=0}^{N-1} \hat{J}_{TP}(\gamma(t_k))\Big\} \tag{3.2}$$

where $\mathbf{E}\{\cdot\}$ stands for the expected value of the random variable within the brace.

Next, we introduce a measure of congestion. It is clear that it depends on the size of the lane capacity $\{C_i, i = 1, 2, \cdots, 8\}$, the desired CBL for congestion warning, and the importance (or weight) given to the level of congestion. The CBL vector is denoted by $\ell = (\ell_1, \ell_2, \cdots, \ell_8)'$. So a reasonable measure of congestion at stage k is given by the following expression,

$$\begin{aligned}\hat{J}_c(\ell(t_k)) = &\sum_{i=1}^{4} W_i(x_{1,i}(t_k) - \ell_i(t_k))I_2(x_{1,i}(t_k) > \ell_i(t_k)) \\ &+ \sum_{i=1}^{4} W_{i+4}(x_{2,i}(t_k) - \ell_{i+4}(t_k))I_2(x_{2,i}(t_k) > \ell_{i+4}(t_k))\end{aligned} \tag{3.3}$$

where the first sum gives the weighted level of congestion on $R1$ during J_k, and the second sum gives the weighted congestion level on $R2$. Hence, the total measure of congestion is given by the following expression,

$$J_c(\ell) \equiv \mathbf{E}\Big\{\sum_{k=0}^{N-1} a(k)\hat{J}_c(\ell(t_k))\Big\}, \tag{3.4}$$

where $a(k)$ can be chosen as,

$$a(k) = \begin{cases} 1 & k = N - 240, N - 239, \cdots, N - 25, \\ 0 & \text{otherwise.} \end{cases}$$

where N is the number of time intervals covering the whole day. Since there is not much traffic around midnight, there is no need to control CBL and hence $a(k)$ is set to zero during midnight: from 0:00 AM to 4:00 AM and from 10:00 PM to 12:00 PM.

It is clear that waiting time for the drivers increases with the increase of CBL setting. In order to incorporate the measure of waiting time we include a third term to the objective functional. This is given by

$$J_w(\ell) \equiv \mathbf{E}\Big\{\sum_{k=0}^{N-1} a(k)\hat{J}_w(\ell(t_k))\Big\} \tag{3.5}$$

where $\hat{J}_w(\ell(t_k))$ is the waiting time at stage k, defined as

$$\hat{J}_w(\ell(t_k)) = \sum_{i=1}^{8} V_i \ \ell_i(t_k) \tag{3.6}$$

where $V_i \geq 0$ represents the weight (importance) given to the waiting time for the i-th traffic stream. This is a fairly good representation of the waiting time since it depends very much on the value of the CBL setting. The larger the CBL setting is, the longer is the waiting time on the queue. Finally, the objective is to choose the pair of decision variables (controls) (γ, ℓ), that minimizes the following cost functional

$$J(\gamma, \ell) \equiv J_c(\ell) - J_{TP}(\gamma) + J_w(\ell) \tag{3.7}$$

3.2 Compact Formulation of the Optimization Problem

3.2.1 Compact Form of the System Dynamics

We are going to use the principle of dynamic programming to solve the optimization problem stated in the preceding section. For this it is convenient to recast the problem using vector notation. For any $k \in \{0, 1, 2, \cdots, N-1\}$ let us denote $t_k \equiv k$ and introduce the following vectors:

$$\begin{aligned} x_k \equiv (&x_{1,1}(t_k), x_{1,2}(t_k), x_{1,3}(t_k), x_{1,4}(t_k), \\ &x_{2,1}(t_k), x_{2,2}(t_k), x_{2,3}(t_k), x_{2,4}(t_k))' \end{aligned} \tag{3.8}$$

$$\hat{\gamma}_k \equiv (\gamma_1(t_k), \gamma_2(t_k), \gamma_3(t_k), \gamma_4(t_k))' \tag{3.9}$$

$$\begin{aligned} \hat{\ell}_k \equiv (&\ell_1(t_k), \ell_2(t_k), \ell_3(t_k), \ell_4(t_k), \\ &\ell_5(t_k), \ell_6(t_k), \ell_7(t_k), \ell_8(t_k))' \end{aligned} \tag{3.10}$$

x_k, $\hat{\gamma}_k$, and $\hat{\ell}_k$ are column vectors in R^8, R^4, and R^8, respectively representing the state vector and the controls. Using these vector notations, the expressions on the righthand side of equations (2.9) - (2.16) are denoted by the column vector

$$x_k + F(k, x_k, \hat{\gamma}_k, \hat{\ell}_k) \equiv G(k, x_k, \hat{\gamma}_k, \hat{\ell}_k) \in R^8 \tag{3.11}$$

Similarly, using the vectors (3.8), (3.9), (3.10) and (3.11), equations (2.9) - (2.16) of the system can be written in the following compact form (as a vector difference equation),

$$x_{k+1} = G(k, x_k, \hat{\gamma}_k, \hat{\ell}_k), \ \ k \in \{0, 1, 2, \cdots, N-1\} \tag{3.12}$$

3.2.2 Parametric (or Control) Constraints

Define the set Γ by

$$\Gamma \equiv \Big\{\gamma \in R^4 : \gamma_i > 0, i = 1, 2, 3, 4 \ \text{ and } \ \sum_{i=1}^{4} \gamma_i = 1\Big\} \tag{3.13}$$

and the set Λ by

$$\Lambda \equiv \{\ell \in R^8 : \alpha C_i \leq \ell_i \leq C_i, i = 1, 2, \cdots, 8\} \tag{3.14}$$

where the fraction α can be chosen by traffic planner or city authority in any desirable range such as $0.8 \leq \alpha \leq 0.95$. The vector ℓ defines the CBL setting at which the intersection is declared to have reached congestion. This information is broadcast so that drivers may choose to avoid the particular direction of the intersection or the intersection itself. This can be done by use of microelectronic devices for monitoring the traffic status and transmitting the information to a central radio broadcasting station.

We denote the decision or control constraint set by $U \equiv \Gamma \times \Lambda$ and use $u = (\gamma, \ell)$ to denote any member of the admissible set U. Using this notation the dynamic system given by (3.12) can be written in the standard form as follows:

$$x_{k+1} = G(k, x_k, u_k), \ x_0 = \xi, \ k \in \mathcal{Z} \equiv \{0, 1, \cdots, N-1\} \tag{3.15}$$

where $x_0 = \xi$ is the initial state and $u_k \equiv (\hat{\gamma}_k, \hat{\ell}_k)$ is the control. Note that the vector G at any stage k (time) depends on the Poisson random vector $p \equiv (p_1, p_2, \cdots, p_8)^{'}$ with the intensity (or mean) functions $\lambda \equiv (\lambda_1, \lambda_2, \cdots, \lambda_8)^{'}$ at time k and this is an integral part of the model. This is what makes the system stochastic. As mentioned earlier, these functions (data) are available from statistical survey of the arrival history of the intersection under question.

3.2.3 Cost Functional

Using the expressions introduced before, we define and write the running cost at stage k as follows,

$$L(k, x_k, u_k) \equiv \hat{J}_c(\ell(t_k)) - \hat{J}_{TP}(\gamma(t_k)) + \hat{J}_w(\ell(t_k)) \tag{3.16}$$

Using this expression the objective (cost) functional (3.7) can be written compactly in the canonical form, as

$$J(u) \equiv \mathbf{E}\Big\{\sum_{k=0}^{N-1} L(k, x_k, u_k) + \Psi(x_N)\Big\} \tag{3.17}$$

where $\Psi(x_N)$ denotes the terminal cost, for example, penalty for deviation from a desired final state and $u \equiv \{u_0, u_1, u_2, \cdots, u_{N-1}\}$ denotes the decision or control policy over the time horizon. Note that there is no impact on the cost functional for any control chosen at the final stage N. The terminal cost $\Psi(x_N)$ depends only on the final state determined by the preceding state and the preceding control.

3.2.4 Optimization Problem

The objective functional is proposed in the preceding section. It is clear that the problem is to minimize the cost functional $J(u)$ given by (3.17) subject to the dynamic constraint (3.15) and the decision or control constraints $U \equiv \Gamma \times \Lambda$. The optimal control policy and optimal state trajectory can be determined by solving this optimization problem.

3.3 Monte Carlo Method

Monte Carlo methods (or Monte Carlo experiments) are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results [19]. They are often used in physical and mathematical problems and are most useful when it is difficult (or impossible) to apply other approaches. Monte Carlo methods are mainly used in three distinct problem classes: optimization, numerical integration, and generating draws from a probability distribution.

In physics-related problems, Monte Carlo methods are useful for simulating systems with many coupled degrees of freedom, such as fluids, disordered materials, strongly coupled solids, etc [11]. Other examples include modeling phenomena with significant uncertainty in inputs such as the calculation of risk in business and, in mathematics, evaluation of multi-dimensional definite integrals with complicated boundary conditions. In principle, Monte Carlo methods can be used to solve any problem having a probabilistic interpretation. By the law of large numbers, integrals described by the expected value of some random variable can be approximated by taking the empirical mean (the sample mean) of independent samples of the variable.

Monte Carlo methods are a generally used set of computational algorithms for the simulation of a wide range of mathematical and physical systems [27]. Since there are large number of computations involved in the repetition of the algorithms, Monte Carlo simulation is a method appropriate for the evaluation of multi-dimensional average on complex spaces by employing different techniques of computer simulation.

Theorem: Convergence in probability of the sample average [18]
Let $h(x)$ be a function of random variable with finite mean $M(h)$ and variance $V(h)$. Let $x_1, x_2, x_3, \cdots, x_n$ be a sample of identical independent random variables, and let

$$\tilde{h}_n = \frac{1}{n} \sum_{i=1}^{n} h(x_i)$$

be the sample average, then $\tilde{h}_n$ converges in probability to $M(h)$.

Note that the sample average $\tilde{h}_n$ gets closer and closer to the actual mean $M(h)$ with the increase of the number of samples n. This is a key point as the number of samples n is not expected to be infinite in the numerical simulation and it is not necessary to be infinite either. We compute the expected values of system performance criteria based on Monte Carlo method, thus it is of great importance to determine the number of sample paths

required for the simulation experiments. It is clear that the required number of samples is very much dependent on the variance of nonhomogeneous Poisson arrival rates. The larger the variance is, the larger is the number of samples required. For numerical experiments, it is reasonable to stop the simulation as long as the convergency of the optimal trajectories can be clearly seen. In the numerical simulation part, we have carried out a number of trials and the convergence of the corresponding results has been observed.

Chapter 4

Dynamic Programming and Recursive Algorithm

In this chapter, we prove the existence of solution to the optimization problem formulated in the preceding chapter. In order to address the problem, we develop the sequential or recursive algorithm based on the idea of dynamic programming technique.

4.1 Dynamic Programming

It is clear that maximum principle requires differentiability of the vector field G with respect to the state variable x. It is evident from the model that there are many indicator functions required for its construction and clearly they are not differentiable in the usual sense. However, dynamic programming method does not require this property and therefore DP is the most suitable technique for the optimization problem stated here.

The idea of dynamic programming is based on the simple fact that, given the present, nothing can alter the past but the future can be improved by appropriate actions starting from the present [20], [22]. Suppose $n-1$ stages have passed and the current stage is n so that the available stages are $\{n, n+1, \cdots, N-1\}$ and the decisions at these stages must be taken so as to minimize the "cost to go" to the final stage N. Let $x_n = \xi$ denote the present state and let $\Phi(n, x_n) = \Phi(n, \xi)$ denote the minimum future cost obtained by proper choice of the control policy $\{u_n^o, u_{n+1}^o, \cdots, u_{N-1}^o\}$ over the remaining time horizon. Thus, according to this philosophy combined with the Markov property (given the present the future is independent of the past) we obtain the following equations,

$$\Phi(n,\xi) = \inf_{u_n, u_{n+1}, \cdots, u_{N-1}} \mathbf{E}\left\{ \left(\sum_{k=n}^{N-1} L(k, x_k, u_k) + \Psi(x_N) \right) |_{x_n=\xi} \right\} \tag{4.1}$$

$$= \inf_{u_n} \mathbf{E}\left\{ L(n, \xi, u_n) + \Phi(n+1, x_{n+1}) \right\} \tag{4.2}$$

The first equation (4.1) gives the conditional expectation of the random variable within the parenthesis given that $x_n = \xi$. The last term in (4.2) is not explicitly dependent on

controls but the state x_{n+1} is dependent on the control u_n through the state equation

$$x_{n+1} = G(n, \xi, u_n)$$

and thus equation (4.2) takes the form

$$\Phi(n, \xi) = \inf_{u_n \in U} \mathbf{E}\Big\{ L(n, \xi, u_n) + \Phi(n+1, G(n, \xi, u_n)) \Big\} \tag{4.3}$$

This equation holds for all $n \in \{0, 1, 2, 3, \cdots, N-1\}$ and $\xi \in \mathcal{S} \subset R^8$ where $\mathcal{S}$ denotes the set of admissible states. In practice this may be a proper subset of R^8. According to the above expression, it is clear that the optimal control at the stage n depends on the state at this stage. Denoting the optimal policy by $u_n^o = v(n, \xi)$ for $n \in \mathcal{Z} \equiv \{0, 1, 2, 3, \cdots, N-1\}$ we arrive at the celebrated Bellman's functional equation,

$$\begin{aligned} \Phi(n, \xi) &= \mathbf{E}\Big\{ L(n, \xi, v(n, \xi)) + \Phi(n+1, G(n, \xi, v(n, \xi))) \Big\} \\ n &\in \mathcal{Z} \equiv \{0, 1, 2, 3, \cdots, N-1\} \quad \text{and} \quad \xi \in \mathcal{S} \subset R^8 \end{aligned} \tag{4.4}$$

where $v(n, \cdot)$ is the optimal decision (control) at the stage n. In other words, the optimal policy is given by

$$v(n, \xi) \equiv arg. \Big\{ \inf_{u_n \in U} \mathbf{E}\big\{ L(n, \xi, u_n) + \Phi(n+1, G(n, \xi, u_n)) \big\} \Big\} \tag{4.5}$$

which is always expected to be a function of the state, the system is at that stage. Before we discuss the algorithm, we present the following existence result.

Proposition 1 The dynamic programming problem consisting of equations (4.4) and (4.5) has at least one solution and hence an optimal feedback control law, $v^o(n, \xi)$, $(n, \xi) \in \mathcal{Z} \times \mathcal{S}$, exists in the class of bounded measurable functions with values in U.

Proof. The proof follows from the simple facts that the set U is compact and that, for every fixed stage $n \in \{0, 1, 2, \cdots, N-1\}$ and state $\xi \in \mathcal{S}$, the functions $L(n, \xi, \cdot)$ and $G(n, \xi, \cdot)$ are continuous.

Using the Bellman equation (4.4) and the equation for the optimal policy (4.5) one can develop an algorithm whereby one can compute the optimal policy and the optimal cost. The functional equation (4.4) can be solved only backward because it is the terminal condition $\Psi(\cdot)$ (not the initial condition) that is given. The backward sweep determines the optimal control policy. Once the optimal policy is determined the state equation is solved forward in time giving the optimal state. Once we are able to construct the function Φ, all other necessary information such as the optimal feedback control law, the optimal cost for the given initial state and the optimal trajectory can be determined. The task of constructing this function using the Bellman equation (4.4) is computationally intensive, which is the so-called curse of dimensionality.

4.2 Sequential or Recursive Algorithm

4.2.1 Backward Sweep

Let $\mathcal{S} \subset R^8$ denote the set of admissible states. Examining the original state equations (2.9) - (2.16) it is clear that this set is bounded (and closed). Similarly the control set $U \equiv \Gamma \times \Lambda \subset R^{12}$ is also a bounded set. By construction of the Bellman function it is meant that for each $n \in \{0, 1, 2, \cdots, N-1\}$ we have the function $\Phi(n, \cdot) = \{\Phi(n, \xi), \xi \in \mathcal{S}\}$. If $\mathcal{S}$ consists of only a finite set of points, the problem is computationally feasible. However if $\mathcal{S}$ is a continuum, the problem is computationally formidable.

In any case the procedure is as described below. Starting at the stage $N-1$ with any state $x_{N-1} \in \mathcal{S}$ we have

$$\begin{aligned}\Phi(N-1, x_{N-1}) &= \inf_{u_{N-1} \in U} \mathbf{E}\Big\{ L(N-1, x_{N-1}, u_{N-1}) + \Psi(x_N) \Big\} \\ &= \inf_{u_{N-1} \in U} \mathbf{E}\Big\{ L(N-1, x_{N-1}, u_{N-1}) + \Psi(G(N-1, x_{N-1}, u_{N-1})) \Big\} \end{aligned} \tag{4.6}$$

By following (4.5) we have the corresponding optimal decision given by

$$u^o_{N-1} = v(N-1, x_{N-1}) \tag{4.7}$$

where v is a suitable function of the current stage $N-1$ and current state x_{N-1}. Note that this gives us only a pair of real numbers for (Φ, v) corresponding to the stage and state pair $(N-1, x_{N-1}) \in \mathcal{Z} \times \mathcal{S}$, that is, $(\Phi(N-1, x_{N-1}), v(N-1, x_{N-1}))$. Keeping $n = N-1$ fixed one must solve the minimization problem (4.6) for a dense set of values for $x_{N-1} \in \mathcal{S}$. One can choose a finite number $(m < \infty)$ of representative points $\{\xi_1, \xi_2, \cdots, \xi_m\}$ from $\mathcal{S}$ and carry out the above procedure to obtain the set

$$\{(\Phi(N-1, \xi_i), v(N-1, \xi_i)),\ i = 1, 2, 3, \cdots, m\}$$

Then one can use interpolation to construct the functions

$$(\Phi(N-1, \cdot), v(N-1, \cdot)) \equiv \{(\Phi(N-1, \xi), v(N-1, \xi)), \xi \in \mathcal{S}\} \tag{4.8}$$

for the given stage $N-1$. This completes only one stage. To continue we go (backward in time) to the next stage and state. Then we have, for any $x_{N-2} \in \mathcal{S}$,

$$\begin{aligned}\Phi(N-2, x_{N-2}) &= \inf_{u_{N-2} \in U} \mathbf{E}\Big\{ L(N-2, x_{N-2}, u_{N-2}) + \Phi(N-1, x_{N-1}) \Big\} \\ &= \inf_{u_{N-2} \in U} \mathbf{E}\Big\{ L(N-2, x_{N-2}, u_{N-2}) + \Phi(N-1, G(N-2, x_{N-2}, u_{N-2})) \Big\} \end{aligned} \tag{4.9}$$

Again following equation (4.5) we obtain the optimal decision at the stage $N-2$ given by

$$u^o_{N-2} = v(N-2, x_{N-2}) \tag{4.10}$$

Finally following the same procedure as in stage $N-1$, we arrive at the following pair of functions

$$(\Phi(N-2,\cdot), v(N-2,\cdot)) \equiv \{(\Phi(N-2,\xi), v(N-2,\xi)), \xi \in \mathcal{S}\} \tag{4.11}$$

for the stage $N-2$. Continuing this process step by step backward till the initial stage $k=0$ is reached, we arrive at the following expression

$$\Phi(0,x_0) = \inf_{u_0 \in U} \mathbf{E}\Big\{ L(0,x_0,u_0) + \Phi(1, G(0,x_0,u_o)) \Big\} \tag{4.12}$$

Again following equation (4.5) we obtain the optimal decision $u_0^o = v(0,x_0)$ at stage $k=0$ and state x_0 and finally the pair of functions

$$(\Phi(0,\cdot), v(0,\cdot)) \equiv \{(\Phi(0,\xi), v(0,\xi)), \xi \in \mathcal{S}\} \tag{4.13}$$

4.2.2 Forward Sweep

Using the optimal feedback control law $u_k^o \equiv v(k,x_k)$ determined from the previous backward sweep, one can solve for the optimal state trajectory using the state dynamics

$$x_{k+1} = G(k, x_k, v(k,x_k)), \quad x_0 = \xi, \quad k = 0,1,2,\cdots,N-1. \tag{4.14}$$

This gives the optimal state trajectory $\{x_k^o, k \in \mathcal{Z}\}$.

4.2.3 Backward-Forward Sweep Combined

Combining the results from the backward and forward sweep, we obtain (i): the value function (Bellman function) Φ (ii): the optimal feedback control law v (iii): the optimal cost $J(u^o) = J(v)$ and (iv): the optimal state trajectory as follows:

$$(i) : \Phi(k,x), \quad k \in \mathcal{Z} \equiv \{0,1,2,\cdots,N-1\}, x \in \mathcal{S} \tag{4.15}$$
$$(ii) : v(k,x), k \in \mathcal{Z}, x \in \mathcal{S} \subset R^8 \tag{4.16}$$
$$(iii) : J(u^o) = J(v) = \Phi(0,x_0) \tag{4.17}$$
$$(iv) : x_{k+1}^o = G(k, x_k^o, u_k^o) = G(k, x_k^o, v(k,x_k^o)), \ x_0^o = x_0, \ k \in \mathcal{Z} \tag{4.18}$$

Based on the above philosophy, we develop the following algorithm whereby the optimal control policy and optimal cost can be determined. In the algorithm, Block 1 is the backward sweep process which determines the optimal feedback control law $u_k^o \equiv v(k,x_k)$. Block 2 describes the forward sweep process solving for the optimal state trajectory $\{x_k^o, k \in \mathcal{Z}\}$ by using the feedback control law determined from Block 1.

Algorithm 1 Sequential or Recursive Algorithm

Initialization:

Choose an array of initial states x_0;

Set system parameters θ, β, W, V, C, $k = N$;

Output:

Optimal cost J^o;

Optimal control vector u^o.

1: Set the terminal cost $J(N) = \Psi(x_N) = 0$ and save it to the table named as "T".
2: Let $k = N - 1$.

Block 1. (Backward Sweep)

3: Read traffic arrival rate at stage k.
4: Read the first element of all combinations in x_k.
5: Find the admissible $u_k \in U$.
6: Compute $\Phi(k, x_k)$ by (4.3).
7: Find the optimal control $u_k^o(x_k)$ that corresponds to $\Phi(k, x_k)$.
8: Let $J(u_k^o) = \Phi(k, x_k)$.
9: Save $J(u_k^o)$, x_k, $u_k^o(x_k)$ to the table "T".
10: If there are more combinations of x_k, then goto Step 11. Otherwise, goto Step 12.
11: Read the next element in the combinations of x_k, then goto Step 5.
12: If $k \neq 0$, then set $k = k - 1$ and goto Step 3. Otherwise goto Step 13.

Block 2. (Forward Sweep)

13: Let $x_0^o = x_0$ which is the given initial state.
14: Find $u_k^o = u_k(x_k^o)$ (starting with $k = 0$) corresponding to the optimal state x_k^o in table "T".
15: Save u_k^o.
16: Compute $x_{k+1}^o = G(k, x_k^o, u_k^o)$ by (3.15).
17: If $k \neq N - 1$, then let $k = k + 1$ and goto Step 14. Otherwise goto Step 18.
18: **return** optimal cost J^o and optimal control vector u^o.

Chapter 5

Numerical Results and Analysis

The implementation of the algorithm proposed in the preceding chapter is done with Matlab program. Numerical results are obtained based on the simulation and optimization programs. Experiments in this thesis are conducted on the platform of Intel(R) i7 2.6G and Windows 7 professional 64-bit operating environment.

In this chapter, we apply the algorithm introduced in Chapter 4 and the implementation scheme presented above to numerical simulation experiments. For this purpose, we assume that the road $R2$ is busier than $R1$. Since the dynamic programming is computationally intensive [15], to reduce the computational burden, we consider a scenario for which one unit of traffic stream stands for a cluster of traffic, say five. This number depends on the actual volume of traffic flow through the intersection. To reduce the computational burden further, we fix the CBL settings for the cross streams as these streams have much less impact on the optimality of the traffic flow. For optimization, we only keep the CBLs (ℓ_3, ℓ_4, ℓ_7 and ℓ_8) corresponding to the straight traffic. In the following parts, we present simulation results for three different cases. There are two scenarios in the first case. One scenario has earlier peak hours while the other has later peak hours. For the second case, we present numerical results for the scenario where the morning rush hour period is shorter while the afternoon rush hour period is longer. In order to illustrate the efficacy of the technique developed in the thesis, we present simulation results for two atypical scenarios in the third case study where the traffic arrival processes have only one peak in the morning for scenario 1 and one in the afternoon for scenario 2.

5.1 Numerical Method Used

In the numerical experiments, the urban traffic control system developed here is modeled as a discrete-time feedback control system. For the entire day ($24h$), the period $I \equiv [0, T]$ is discretized into $N = 288$ time intervals with 5 mins for each. The sequence of time intervals $J_k \equiv (t_k, t_{k+1}]$ starts with $k = 0$, $0 = t_0 < t_1 < t_2 < \cdots < t_k < t_{k+1} < \cdots < t_{N-1}$ and the length of each time interval is $\Delta = t_1 - t_0 = t_2 - t_1 = \cdots = t_{k+1} - t_k = \cdots = t_{N-1} - t_{N-2}$. During each sufficiently small time interval J_k, the mean arrival rates do not change. The

corresponding nonhomogeneous Poisson processes are applied to the experiments on the basis of the given mean arrival rates during each small time interval.

5.2 Simulation Environment Setup

We have carried out a series of numerical experiments for three different cases based on the same relevant parameters, however the nonhomogeneous Poisson arrival rates applied to the simulations have their own characteristics for those cases under consideration. As we have stated before, the choice of the length of each time interval $J_k \equiv (t_k, t_{k+1}]$ has to be sufficiently small so that the statistical mean of the Poisson arrival rate does not change during each time interval. Here we choose 5 minutes for each time interval which appears to be fairly reasonable. Accordingly, we have 288 stages (time periods) throughout the whole day. The parameters chosen for numerical experiments are given in the following Table 5.1.

Table 5.1: Parameters for Simulation

Road 1	Road 2
$C_1 = 1, C_2 = 2$ $C_3 = 3, C_4 = 4$	$C_5 = 2, C_6 = 3$ $C_7 = 5, C_8 = 6$
$\ell_1 = 1, \ell_2 = 2$ $\ell_3 \in \{2, 3\}, \ell_4 \in \{3, 4\}$	$\ell_5 = 2, \ell_6 = 3$ $\ell_7 \in \{4, 5\}, \ell_8 \in \{5, 6\}$
$W_1 = 50, W_2 = 100$ $W_3 = 300, W_4 = 350$	$W_5 = 100, W_6 = 150$ $W_7 = 360, W_8 = 400$
$V_1 = 10, V_2 = 15$ $V_3 = 30, V_4 = 40$	$V_5 = 15, V_6 = 20$ $V_7 = 50, V_8 = 60$
$\beta_1 = 1, \beta_2 = 1$ $\beta_3 = 2, \beta_4 = 2$	$\beta_5 = 1, \beta_6 = 1$ $\beta_7 = 2, \beta_8 = 2$
$\theta_1 = 0.2, \theta_2 = 0.2$ $\theta_3 = 0.1, \theta_4 = 0.1$	$\theta_5 = 0.2, \theta_6 = 0.2$ $\theta_7 = 0.1, \theta_8 = 0.1$

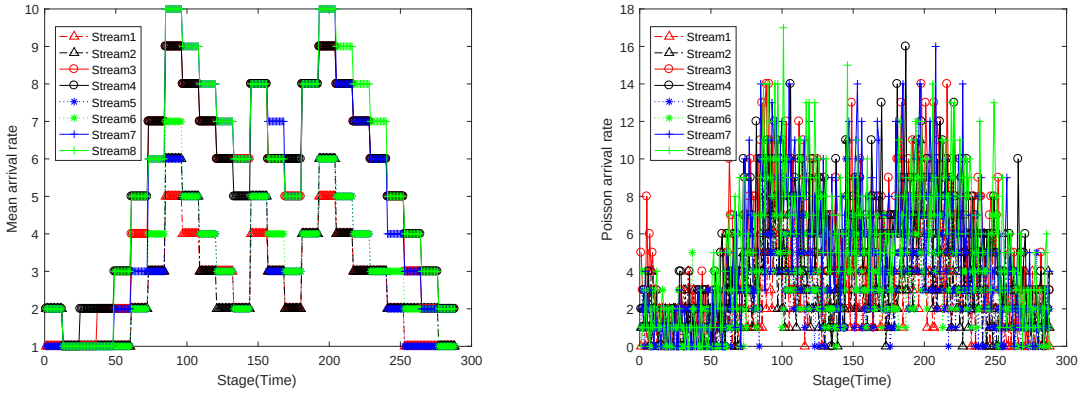
First of all, it can be seen from the table that we choose larger lane capacity C_i for both straight and cross traffic on and from the busier road $R2$. Based on the chosen lane capacity C_i, the corresponding CBL setting ℓ_i has to be chosen accordingly. As we have mentioned before, we only keep the CBL settings (ℓ_3, ℓ_4, ℓ_7 and ℓ_8) corresponding to the straight traffic for optimization purpose while keeping those CBLs fixed for cross traffic. In addition, given the scenarios considered in the thesis for simulation experiments, we assign more weight (importance) to the parameters for the busier road $R2$. More specifically, the weights (importance) W_i given for the congestion level and V_i for waiting time for

the busier road $R2$ are larger than that for $R1$. It is clear that there are more vehicles served in the straight stream than in the corresponding cross stream over the same time period. Thus, the service capacity β_i for straight traffic is relatively larger than that for cross traffic. Besides, when the i-th stream is under congestion, the reduced arrival ratio θ_i for straight traffic is smaller than that for cross traffic since there are relatively more alternatives for vehicles involving in the straight streams. Note that the initial condition used in our experiments for x_0 at midnight is $(0, 1, 1, 2, 0, 1, 1, 2)'$. It is understood that, depending on the data and statistics of a specific intersection simulated in a particular setup, the relative parameters will have to be set up accordingly.

5.3 Specification of Traffic Arrival Rates

5.3.1 Traffic Arrival Rates of the First Case Study

For the two different scenarios in the first case study, we apply traffic arrival rates with two main peak-hour periods, one in the morning and another one in the afternoon. Note that there is also a moderate peak around noon for the numerical experiments. As we have already assumed that $R2$ is busier than $R1$, the straight streams on $R2$ are heavier than that on $R1$ and so are the cross streams. The mean arrival rates (rate functions) $\lambda_7(\cdot)$ and $\lambda_8(\cdot)$ of straight traffic on $R2$ are relatively larger than $\lambda_3(\cdot)$ and $\lambda_4(\cdot)$ on $R1$ during most of the time over the whole day. This is also true for the mean arrival rates (rate functions) $\lambda_5(\cdot)$ and $\lambda_6(\cdot)$ of cross traffic from $R2$, and $\lambda_1(\cdot)$ and $\lambda_2(\cdot)$ from $R1$. However, these experiments include two different scenarios of traffic arrival rates: one has earlier peak-hour period and another one has later peak-hour period.

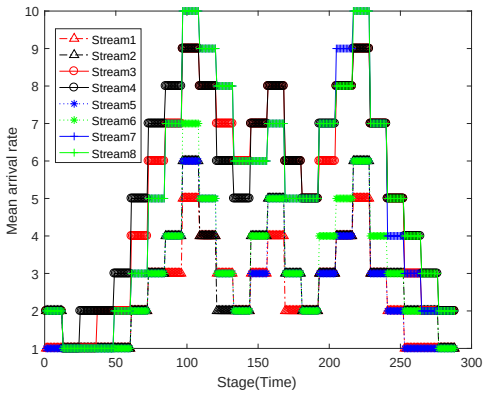


(a) Mean arrival rate

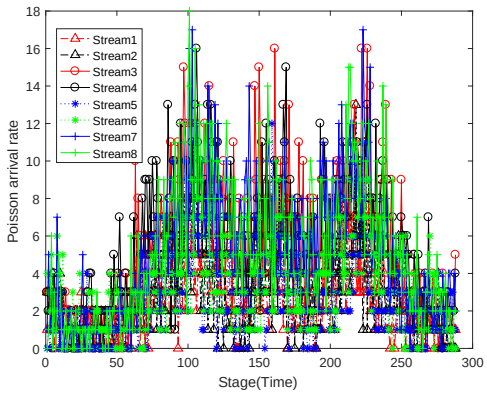
(b) Poisson arrival rate

Figure 5.1: Arrival rates of eight streams in scenario 1 (Case 1).

Figure 5.1a shows the mean arrival rates of eight traffic streams over the whole day in scenario 1, in which there is a morning peak-hour period around 7 : 30 AM and another afternoon peak-hour period close to 4 : 30 PM. This is a common phenomenon in Canada.



(a) Mean arrival rate

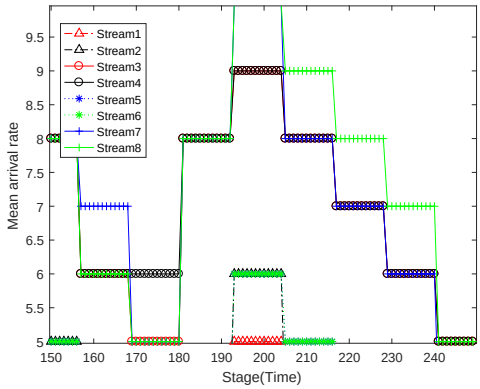


(b) Poisson arrival rate

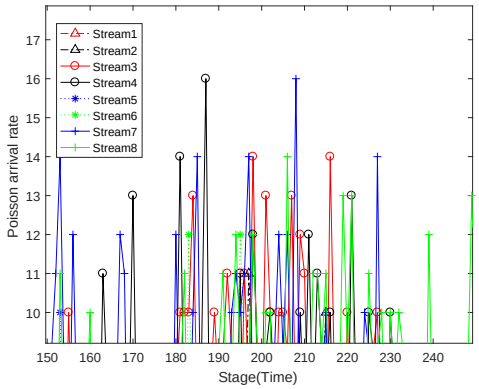
Figure 5.2: Arrival rates of eight streams in scenario 2 (Case 1).

However, as shown in Figure 5.2a for scenario 2, the mean arrival rates of eight traffic streams during the whole day have a morning rush hour period around 8 : 30 AM and approximate afternoon rush hour period near 6 : 00 PM, which is the usual case in China. However, it should be noted that the actual number of vehicles arriving in an intersection is random and it follows the nonhomogeneous Poisson process corresponding to the mean intensities shown in Figure 5.1a and Figure 5.2a. It is these random processes as shown in Figure 5.1b for scenario 1 and Figure 5.2b for scenario 2, that are used for the numerical experiments.

For illustration, two segments (approximate 8 hours period) of both the mean arrival rates and the corresponding Poisson arrival rates of scenario 1 are shown in Figure 5.3a and 5.3b, respectively.



(a) Mean arrival rate

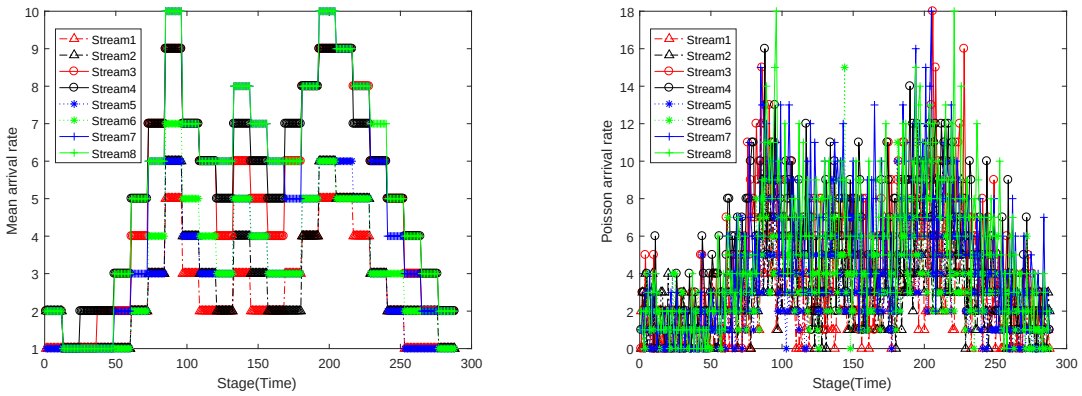


(b) Poisson arrival rate

Figure 5.3: Segment of the arrival rates in scenario 1 (Case 1).

5.3.2 Traffic Arrival Rates of the Second Case Study

For the numerical experiments of the second case study, the traffic arrival rates applied in the simulation also have two main peak-hour periods, one in the morning and another one in the afternoon. However, there is only one scenario where the morning peak-hour period (roughly from 7 : 00 AM to 8 : 00 AM) is shorter than the afternoon peak-hour period (roughly from 4 : 00 PM to 7 : 00 PM). This may occasionally happen when there are some social activities or big events in the afternoon. There is a moderate peak around noon for the traffic arrival rates as well. As it has been mentioned that $R2$ is busier than $R1$, both the straight streams and the cross streams on $R2$ are relatively heavier than that on $R1$. The mean arrival rates (rate functions) $\lambda_7(\cdot)$ and $\lambda_8(\cdot)$ of straight traffic on $R2$ are generally larger than $\lambda_3(\cdot)$ and $\lambda_4(\cdot)$ on $R1$. This also holds for the mean arrival rates (rate functions) $\lambda_5(\cdot)$ and $\lambda_6(\cdot)$ of cross traffic from $R2$, and $\lambda_1(\cdot)$ and $\lambda_2(\cdot)$ from $R1$.



(a) Mean arrival rate

(b) Poisson arrival rate

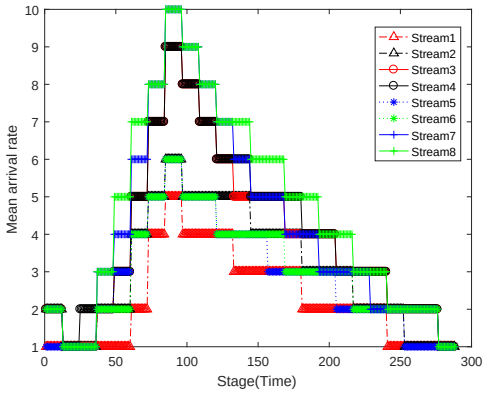
Figure 5.4: Arrival rates of eight streams (Case 2).

It is shown in Figure 5.4a that the mean arrival rates of eight traffic streams over the whole day have a shorter morning rush hour period and longer afternoon rush hour period. As mentioned above, this may occur occasionally in reality. Figure 5.4b shows the corresponding Poisson arrival rates which are actually used in the simulation experiments.

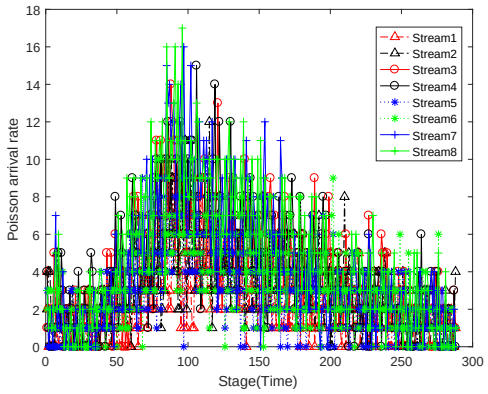
5.3.3 Traffic Arrival Rates of the Third Case Study

For the numerical simulations in the third case study, we carry out the experiments where the traffic arrival rates have only one peak-hour period. This is the untypical case, however it is helpful for validating the feasibility of the dynamic model proposed in the thesis. As has been assumed that $R2$ is busier than $R1$, there are more vehicles going straight on $R2$ than on $R1$ and so are the vehicles in the cross streams. Thus, the mean arrival rates (rate functions) $\lambda_7(\cdot)$ and $\lambda_8(\cdot)$ of straight traffic on $R2$ are relatively larger than $\lambda_3(\cdot)$ and $\lambda_4(\cdot)$ on $R1$ during most of the time over the entire day. This is also true for the mean arrival rates (rate functions) $\lambda_5(\cdot)$ and $\lambda_6(\cdot)$ of cross traffic from $R2$, and $\lambda_1(\cdot)$ and $\lambda_2(\cdot)$

from $R1$. However, these are two different scenarios of traffic arrival rates included in the numerical experiments: one has only one peak-hour period in the morning and the other one has only one peak-hour period in the afternoon.

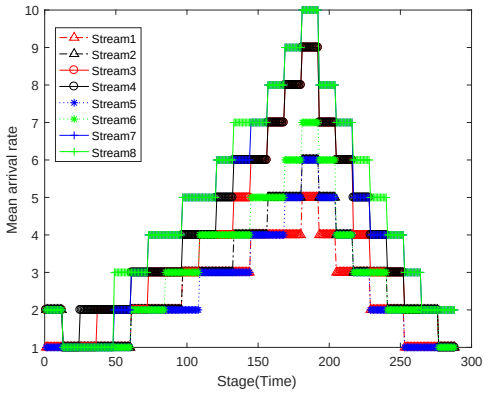


(a) Mean arrival rate

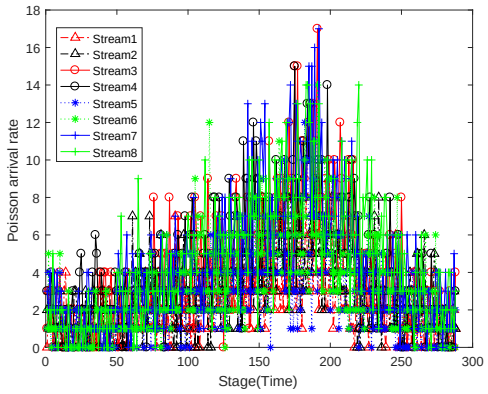


(b) Poisson arrival rate

Figure 5.5: Arrival rates of eight streams in scenario 1 (Case 3).



(a) Mean arrival rate



(b) Poisson arrival rate

Figure 5.6: Arrival rates of eight streams in scenario 2 (Case 3).

The mean arrival rates of eight different traffic streams throughout the entire day for scenario 1 are shown in Figure 5.5a, in which there is only a morning peak-hour period around $7:30$ AM. In the meanwhile, for scenario 2 there is only a afternoon peak-hour period around $4:00$ PM as shown in Figure 5.6a. Note that the actual number of vehicles arriving in an intersection is stochastic and it follows the nonhomogeneous Poisson process corresponding to the mean intensities shown in Figure 5.5a and Figure 5.6a for scenario 1 and scenario 2, respectively. It is these random processes as shown in Figure 5.5b for scenario 1 and Figure 5.6b for scenario 2, that are used for the numerical simulations.

5.4 Experimental Results and Interpretations

We have conducted a series of numerical experiments to determine the optimal paths of control variables from the sets $\{\gamma_i, i = 1, 2, 3, 4\}$ and $\{\ell_i, i = 3, 4, 7, 8\}$. Based on our previous assumption that $R2$ is busier than $R1$, we allocate 70% of the time in each interval to the four straight streams, which is to say $\gamma_2 + \gamma_4 = 0.7$. This yields $\gamma_1 + \gamma_3 = 0.3$ for the four cross streams. In order to discretize those control variables, we choose γ_3 and γ_4 from sets $\{0.15, 0.20\}$ and $\{0.35, 0.40, 0.45, 0.50\}$ respectively so that γ_3 is not less than γ_1, and γ_4 is greater than or equal to γ_2 ($R2$ is generally busier than $R1$). Note that it is not necessarily required to have 70% of the time in each interval allocated to the four straight streams. The distribution of time allocated to straight and cross streams depends on the long-term statistics (data) which can be collected from sensors and detectors deployed in the intersection. Numerical results of this specific scenario are shown in following sections based on three hundred (Monte Carlo) simulation experiments.

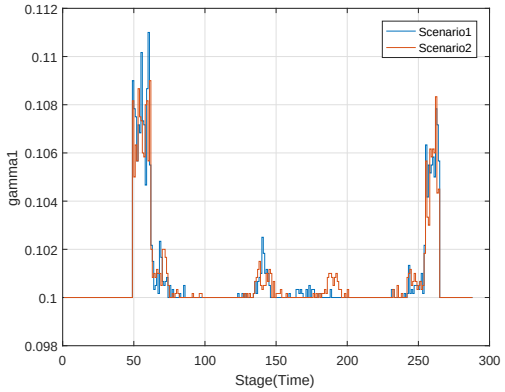
5.4.1 Experiments of the First Case Study

Figure 5.7 shows the expected optimal values for each of the control variables $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$ throughout the entire day (starting from midnight to midnight). This is based on three hundred (Monte Carlo) simulation experiments involving the cost functional given by (3.7). In this case, the arrival rate of straight streams on $R2$ is closer to that on $R1$ around midnight. Thus, during this period, γ_4 has slightly larger value than that of γ_2. Considering the rush hours, the difference between the arrival rates of straight streams on $R2$ and $R1$ reaches the largest value around early peak hour and late rush hour. Therefore, the algorithm allocates more time to straight streams on $R2$ while much less for the same on $R1$. This means that, around these two peak-hour periods, it would be the peak of γ_4 and bottom of γ_2. For illustration, two segments (approximate 4 hours period) of the values of γ_2 and γ_4 around afternoon rush hours are shown in Figure 5.8a and 5.8b, respectively.

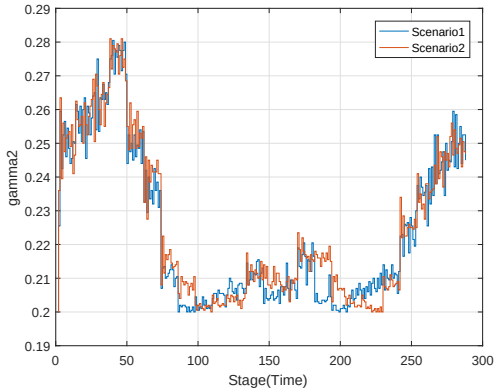
As for the cross traffic, there is no congestion around midnight (from 0:00 AM to 4:00 AM and from 10:00 PM to 12:00 PM), thus the algorithm allocates as much time as possible to the busier road $R2$. Therefore, γ_3 reaches the maximum value while γ_1 stays at the minimum value during this period. However, in order to incorporate the congestion, waiting time and throughput except for the lean period, more time would be allocated to the cross traffic from $R1$ to $R2$ (corresponding to γ_1) when it gets close to the cross traffic from $R2$ to $R1$ (corresponding to γ_3). This is indicated as the peak of γ_1 in Figure 5.7a and bottom of γ_3 in Figure 5.7c. However, during most of the day time the cross traffic from $R2$ to $R1$ is much heavier than that from $R1$ to $R2$, hence γ_3 stays at almost the largest value while γ_1 is near the minimum value.

Since this is the adaptive distribution of time for the dynamic system, it is evident that generally more time would be allocated to γ_4 while less to γ_2 when the straight streams on $R2$ are heavier than that on $R1$. This also holds for the four cross streams including two from $R1$ to $R2$ (corresponding to γ_1) and other two from $R2$ to $R1$ (corresponding to

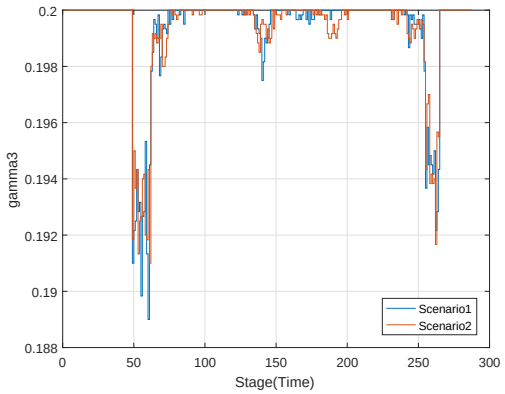
γ_3). During most of the day time, there would be more time allocated to γ_3 while less to γ_1 since the cross streams from $R2$ to $R1$ are generally heavier than that from $R1$ to $R2$.



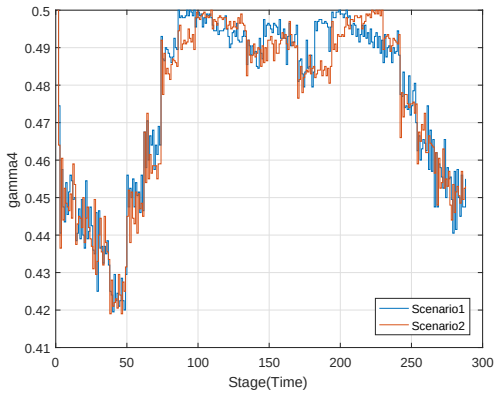
(a) Optimal path for γ_1



(b) Optimal path for γ_2

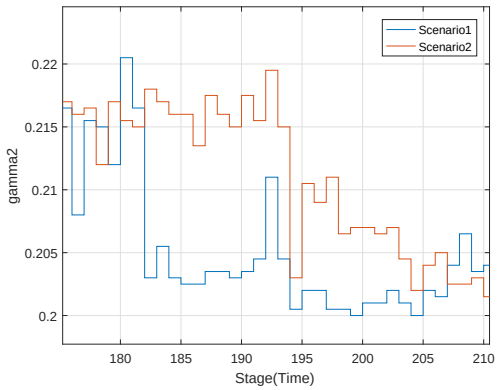


(c) Optimal path for γ_3

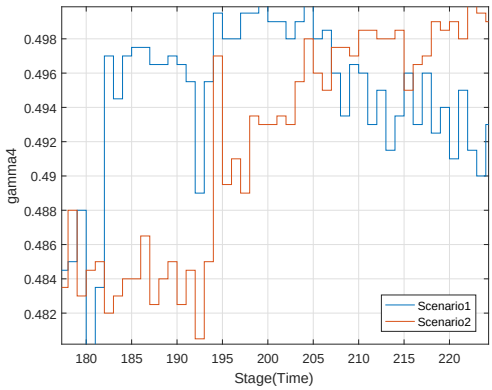


(d) Optimal path for γ_4

Figure 5.7: Optimal values for γ_1, γ_2, γ_3 and γ_4 (Case 1).

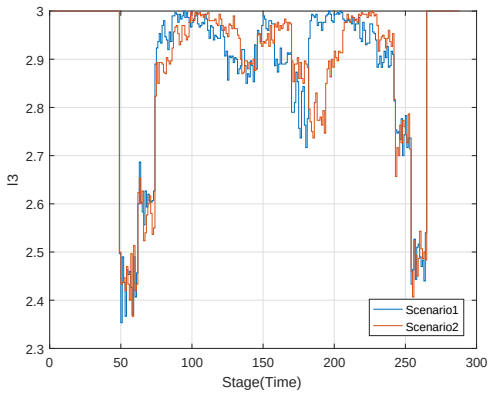


(a) γ_2

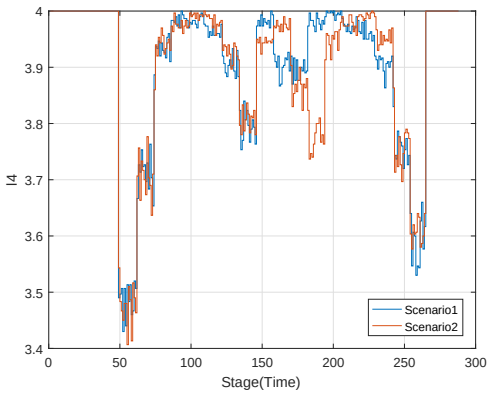


(b) γ_4

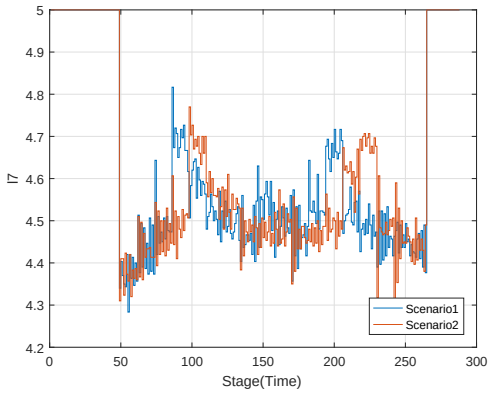
Figure 5.8: γ_2 and γ_4 in the afternoon peak (Case 1).



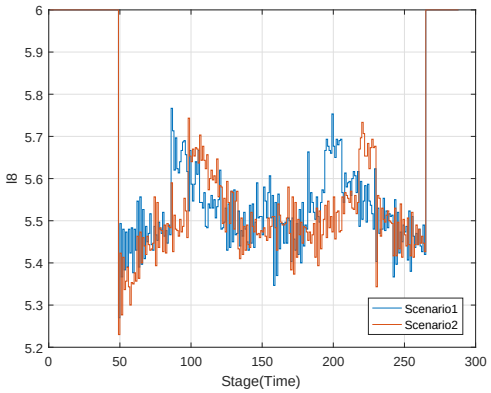
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4



(c) Optimal path for ℓ_7



(d) Optimal path for ℓ_8

Figure 5.9: Optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (Case 1).

Figure 5.9 depicts the expected optimal values of the four CBL settings (starting from midnight to midnight) for the two straight streams on $R1$, and another two on $R2$. It is reasonable to consider only the four straight streams for the optimization of CBL settings because there is usually much more congestion of those straight traffic than that of cross traffic. It is not necessarily required to take into account the CBL settings for cross traffic since there is usually very low volume of traffic in the cross streams. It can be observed from Figure 5.9 that the CBL settings stay at a very much high level in the very early morning and late evening hours so that there is no radio broadcast as there is usually no congestion during these lean periods. Except for those lean periods, however, CBLs will be increased during the day time with the increase of traffic flow and reach the peak around morning and afternoon rush hours. This is understandable because throughput plays a more important role during the day time, especially around the peak-hour period. In this way the throughput could be increased during the day time at the relatively low expense of the increase of waiting time. For simplicity, one can also choose the average of CBL settings during $24h$ as the one for the whole day.

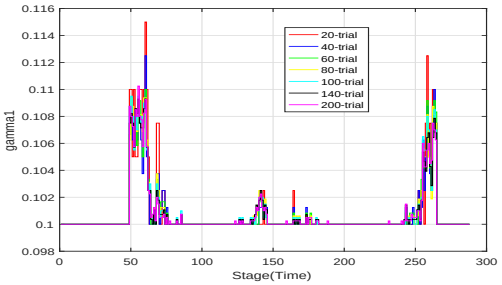
It can be also clearly seen from both Figure 5.7 and Figure 5.9 that there is similar

scheme of the simulation results for both scenario 1 and scenario 2, however, the peaks or bottoms of the graphs for scenario 2 appear later compared with that of scenario 1. This is understood because the rush hours in scenario 2 occur later than that in scenario 1.

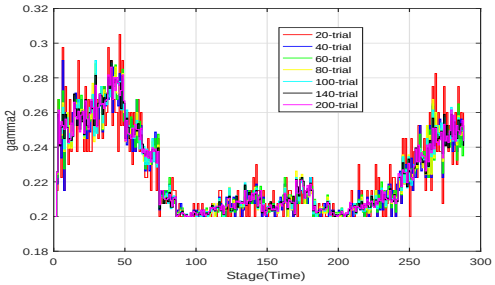
For scenario 1, we have carried out a number of Monte Carlo simulation experiments (20, 40, 60, 80, 100, 140, 200). The corresponding optimal trajectories of control variables $\{\gamma_i, i = 1, 2, 3, 4\}$ and $\{\ell_i, i = 3, 4, 7, 8\}$, are shown in Figure 5.10 and Figure 5.11, respectively. Recall that the intensity (or mean) function $\lambda \equiv (\lambda_1, \lambda_2, \cdots, \lambda_8)$ is an integral part of the dynamic model and it makes the system stochastic. Hence, we can find the convergency of the optimal trajectories with the increase of the number of Monte Carlo simulation experiments. A careful scrutiny of any one of the graphs will clearly show that the respective optimal trajectories converge to a specific path as the number of experiments (or samples) increases. In the meanwhile, we have also conducted a number of Monte Carlo simulations for scenario 2 and observed similar convergency results that are shown in Figure 5.12 and Figure 5.13.

It is well-known that according to dynamic programming the optimal cost is dependent on the initial state. Table 5.2 shows different optimal costs corresponding to a number of various initial states. It is clear from these results that the costs do not change much for similar initial states. This is expected from the theory of dynamic programming asserting spatial continuity of the value function $\xi \longrightarrow \Phi(n, \xi)$. It can be seen from Table 5.2 that the optimal cost varies dramatically with significant change of the initial state.

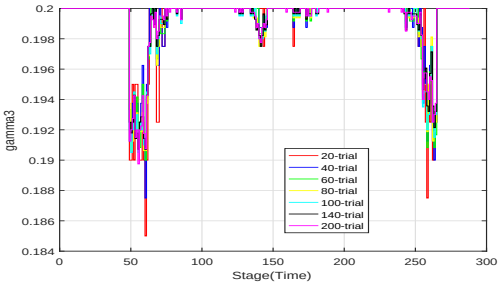
In order to compare the performance of the proposed dynamic model with heuristic traffic control policies, we carry out corresponding computation without any optimization. More specifically, we allocate constant time for each of the traffic streams, for example $\gamma_1 = 0.1$, $\gamma_2 = 0.25$, $\gamma_3 = 0.2$ and $\gamma_4 = 0.45$. Moreover, the CBLs for straight streams are also fixed as $\ell_3 = 3$, $\ell_4 = 4$, $\ell_7 = 4$ and $\ell_8 = 5$. In this case, the costs corresponding to different initial conditions are shown in Table 5.3. It can be clearly observed that the optimal cost (Table 5.2) given by the proposed dynamic model is much less than the corresponding cost (Table 5.3) given by a heuristic policy for the same initial condition.



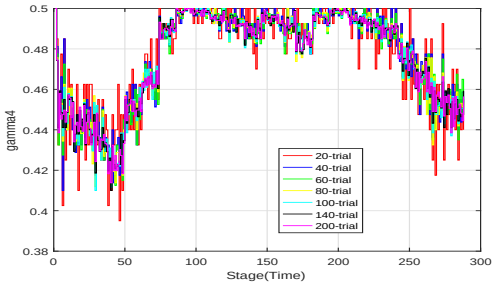
(a) Optimal path for γ_1



(b) Optimal path for γ_2

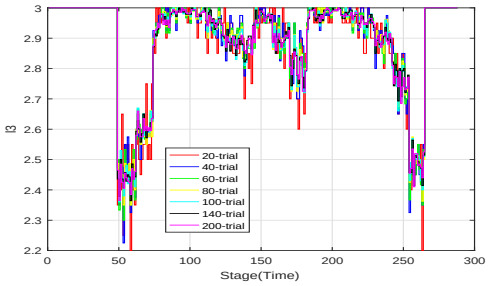


(c) Optimal path for γ_3

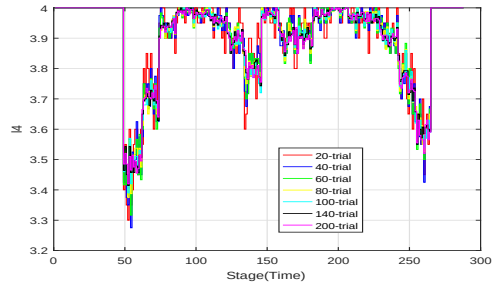


(d) Optimal path for γ_4

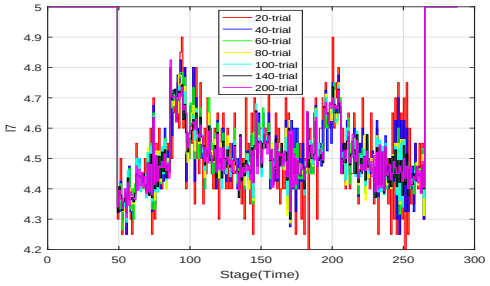
Figure 5.10: A set of optimal values for γ_1, γ_2, γ_3 and γ_4 (Scenario 1).



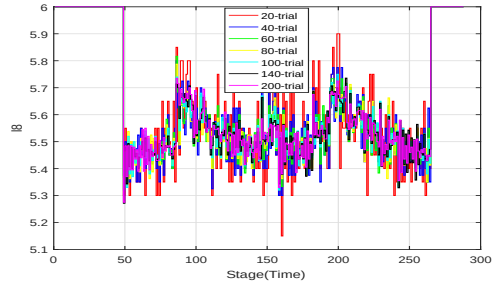
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4

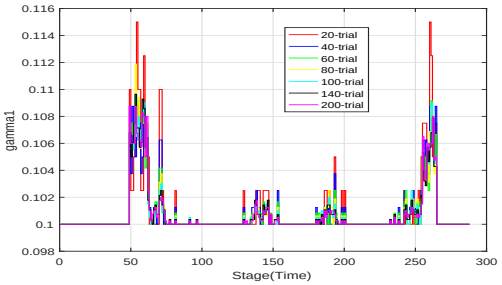


(c) Optimal path for ℓ_7

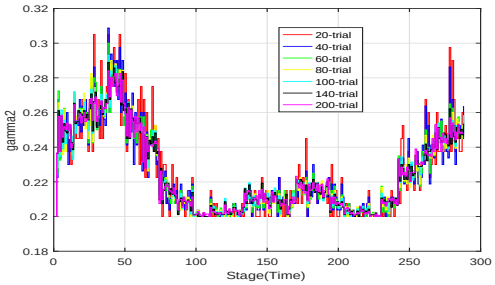


(d) Optimal path for ℓ_8

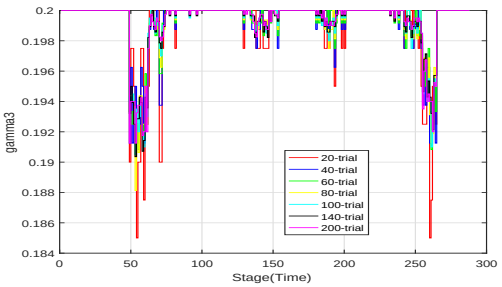
Figure 5.11: A set of optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (Scenario 1).



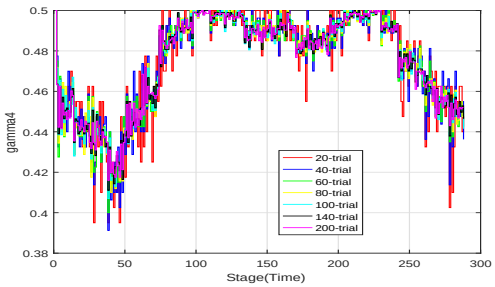
(a) Optimal path for γ_1



(b) Optimal path for γ_2

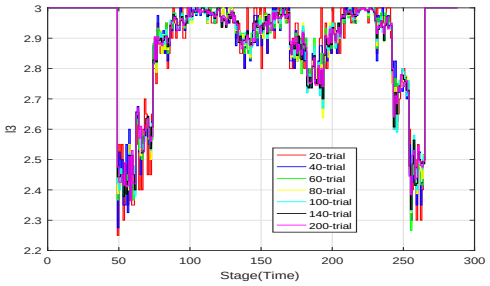


(c) Optimal path for γ_3

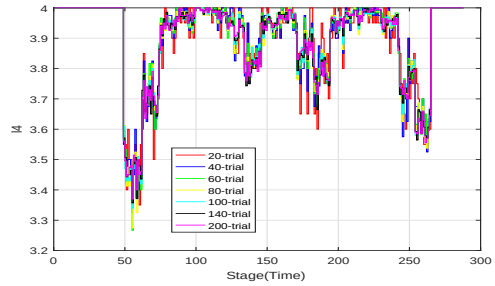


(d) Optimal path for γ_4

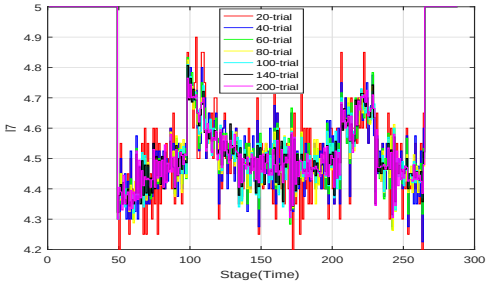
Figure 5.12: A set of optimal values for γ_1, γ_2, γ_3 and γ_4 (Scenario 2).



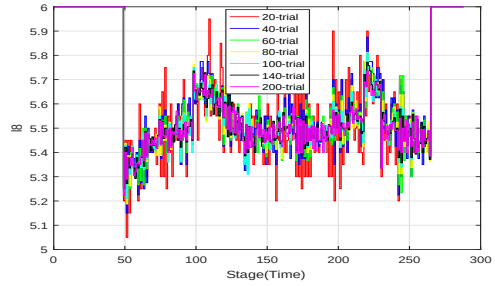
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4



(c) Optimal path for ℓ_7



(d) Optimal path for ℓ_8

Figure 5.13: A set of optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (Scenario 2).

Table 5.2: Optimal Cost Versus Initial Condition for Two Scenarios

Initial Condition	Optimal Cost (Scenario 1)	Optimal Cost (Scenario 2)
(0, 0, 0, 0, 0, 0, 0, 0)	377650	380314
(0, 0, 0, 0, 0, 0, 0, 1)	377647	380312
(0, 0, 0, 1, 0, 0, 0, 1)	377646	380311
(0, 0, 1, 0, 0, 0, 1, 1)	377645	380308
(0, 0, 1, 1, 0, 0, 1, 1)	377644	380307
(0, 0, 1, 1, 1, 0, 1, 1)	377643	380306
(1, 2, 3, 4, 2, 3, 5, 6)	-13	-10

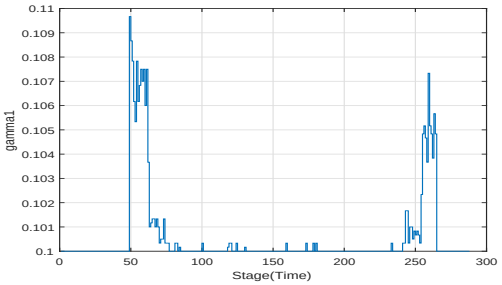
Table 5.3: Cost Versus Initial Condition Without Optimization

Initial Condition	Cost (Scenario 1)	Cost (Scenario 2)
(0, 0, 0, 0, 0, 0, 0, 0)	2685200	2637100
(0, 0, 0, 0, 0, 0, 0, 1)	2687600	264900
(0, 0, 0, 1, 0, 0, 0, 1)	2692100	2645300
(0, 0, 1, 0, 0, 0, 1, 1)	2679300	2640700
(0, 0, 1, 1, 0, 0, 1, 1)	2683400	2640900
(0, 0, 1, 1, 1, 0, 1, 1)	2691700	2635200
(1, 2, 3, 4, 2, 3, 5, 6)	2690400	2645000

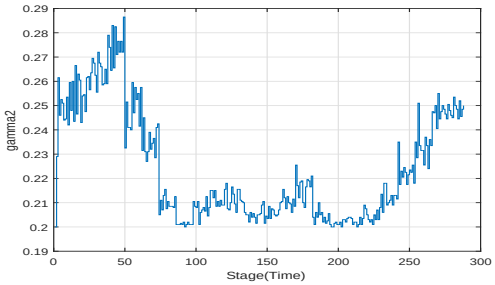
5.4.2 Experiments of the Second Case Study

Keeping the parameters the same as in the first case study, we carried out a number of numerical experiments for the second case where the traffic arrival rates have a shorter peak-hour period in the morning while a longer one in the afternoon. As we have mentioned before, this may happen occasionally in reality. In this case, both the mean arrival rates and the corresponding Poisson arrival processes are shown in Figure 5.4a and Figure 5.4b, respectively.

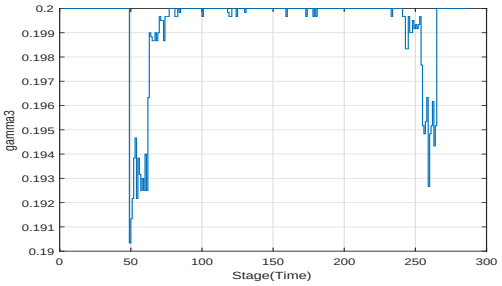
It can be seen from the graphs that the morning peak-hour period lasts for about 1 hour while the afternoon one lasts for around 3 hours. The expected optimum values for both time allocation (γ_i) and CBL settings (ℓ_i) were found for the whole day on the basis of three hundred (Monte Carlo) simulation experiments. They are shown in Figure 5.14 and Figure 5.15, respectively.



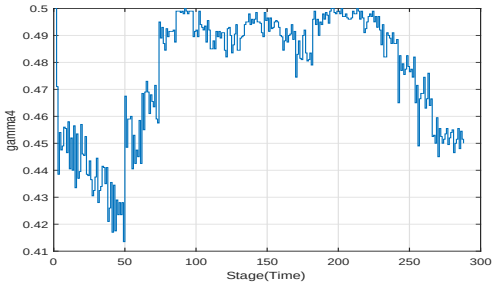
(a) Optimal path for γ_1



(b) Optimal path for γ_2

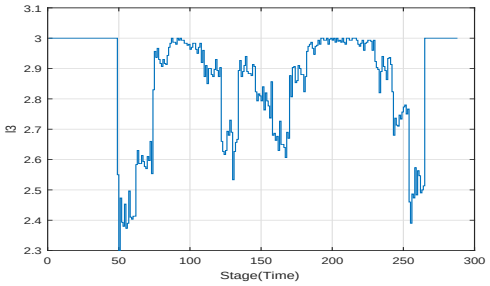


(c) Optimal path for γ_3

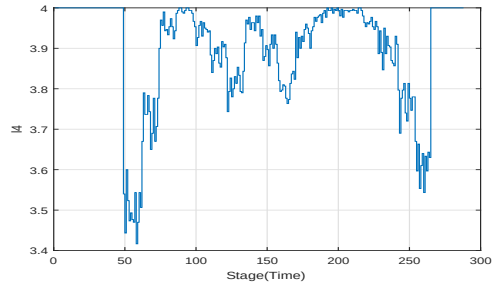


(d) Optimal path for γ_4

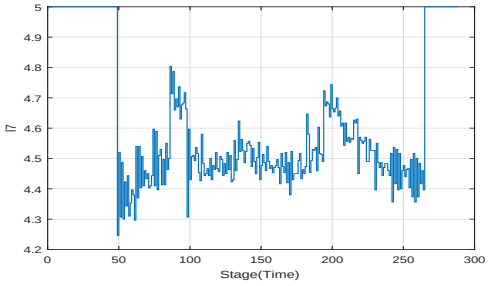
Figure 5.14: Optimal values for γ_1, γ_2, γ_3 and γ_4 (Case 2).



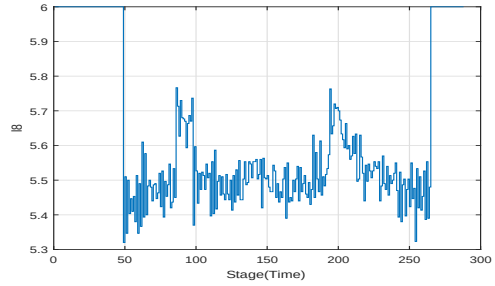
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4



(c) Optimal path for ℓ_7



(d) Optimal path for ℓ_8

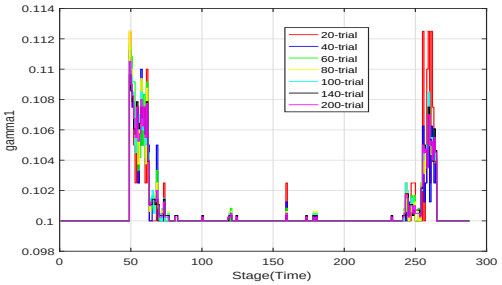
Figure 5.15: Optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (Case 2).

Figure 5.14 shows the optimization result for the control variable $\{\gamma_i, i = 1, 2, 3, 4\}$. As for the straight traffic, since the difference between $R2$ and $R1$ reaches the maximum during the two peak-hour periods, there will be much more time allocated to the busier road $R2$ during that time. This is the peak of γ_4 and bottom of γ_2 as shown in Figure 5.14d and Figure 5.14b, respectively. However, the duration of the peak-hour period in the afternoon is almost three times of that in the morning, hence it can be observed from the graphs that the duration of the peak of γ_4 in the afternoon is nearly three times of that in the morning. On the contrary, it can be also seen that the endurance of the bottom of γ_2 in the afternoon is approximately three times of that in the morning. There is similar interpretation for cross traffic associated with γ_1 and γ_3 as shown in Figure 5.14a and Figure 5.14c, respectively.

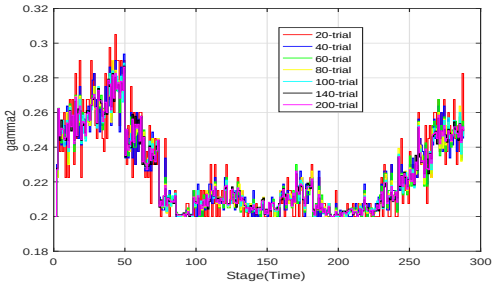
Figure 5.15 shows the optimization result for the control variable $\{\ell_i, i = 3, 4, 7, 8\}$. It is clear that the CBLs are set relatively higher during rush hours. In addition, it can be clearly observed from the graphs that the four CBLs are set higher for much longer time during the afternoon rush hours compared with the rush hours in the morning. This is also understood because the rush hour period lasts much longer in the afternoon.

In order to study the convergency of the expected optimal results, we have conducted a series of Monte Carlo simulation experiments. The corresponding optimal trajectories of control variables $\{\gamma_i, i = 1, 2, 3, 4\}$ and $\{\ell_i, i = 3, 4, 7, 8\}$, are shown in Figure 5.16 and Figure 5.17, respectively. It can be scrutinized from each of the graphs that the optimal trajectories converge to a particular path with the increase of the number of trials.

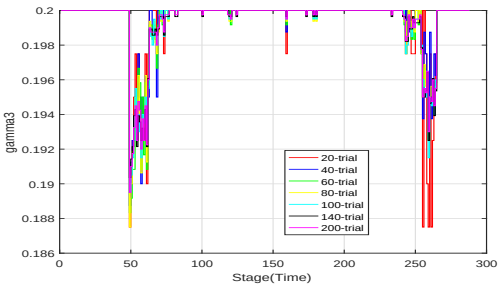
For this specific scenario, a group of optimal costs corresponding to a number of different initial states are shown in Table 5.4. We also compute the cost for this scenario with the same heuristic policy mentioned in the previous section. The corresponding result is shown in Table 5.5. It can be clearly seen that the optimal cost (Table 5.4) given by the proposed dynamic model is much less compared with the corresponding cost (Table 5.5) given by the heuristic policy for the same initial condition.



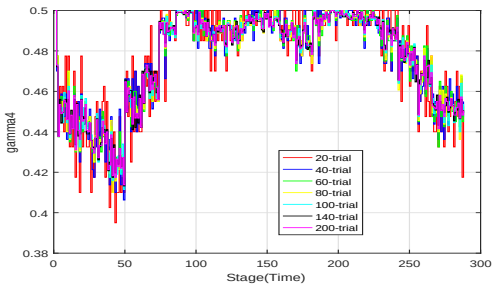
(a) Optimal path for γ_1



(b) Optimal path for γ_2

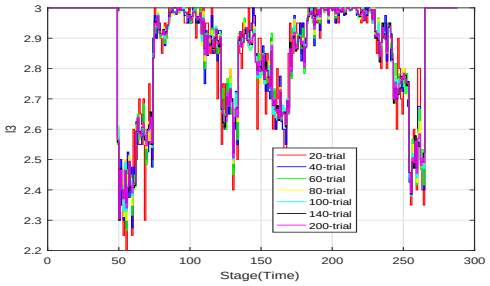


(c) Optimal path for γ_3

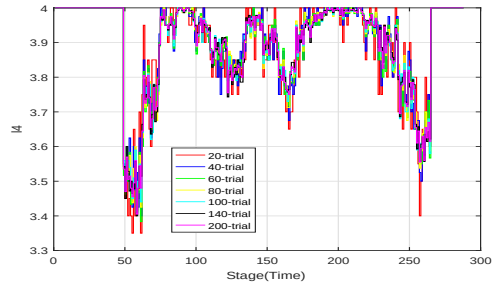


(d) Optimal path for γ_4

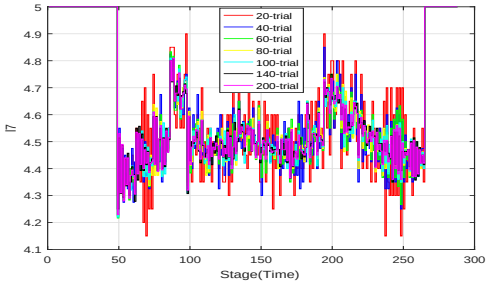
Figure 5.16: A set of optimal values for γ_1, γ_2, γ_3 and γ_4 (Case 2).



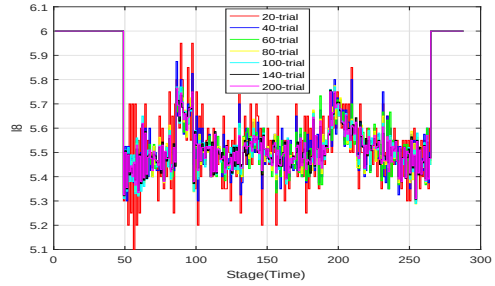
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4



(c) Optimal path for ℓ_7



(d) Optimal path for ℓ_8

Figure 5.17: A set of optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (Case 2).

Table 5.4: Optimal Cost Versus Initial Condition

Initial Condition	Optimal Cost
(0, 0, 0, 0, 0, 0, 0, 0)	378015
(0, 0, 0, 0, 0, 0, 0, 1)	378012
(0, 0, 0, 1, 0, 0, 0, 1)	378011
(0, 0, 1, 0, 0, 0, 1, 1)	378012
(0, 0, 1, 1, 0, 0, 1, 1)	378011
(0, 0, 1, 1, 1, 0, 1, 1)	378010
(1, 2, 3, 4, 2, 3, 5, 6)	-195

Table 5.5: Cost Versus Initial Condition Without Optimization

Initial Condition	Cost
(0, 0, 0, 0, 0, 0, 0, 0)	2728500
(0, 0, 0, 0, 0, 0, 0, 1)	2725400
(0, 0, 0, 1, 0, 0, 0, 1)	2716100
(0, 0, 1, 0, 0, 0, 1, 1)	2719000
(0, 0, 1, 1, 0, 0, 1, 1)	2708600
(0, 0, 1, 1, 1, 0, 1, 1)	2715800
(1, 2, 3, 4, 2, 3, 5, 6)	2721600

5.4.3 Experiments of the Third Case Study

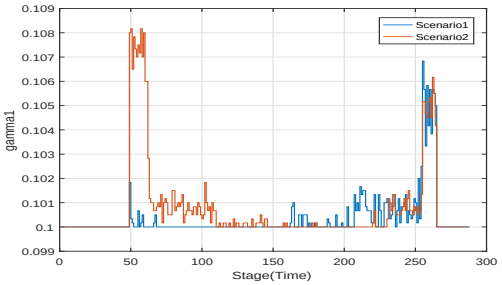
For further illustration, we also conducted another group of numerical experiments for the third case study while keeping relevant parameters the same as the previous two case studies. Specifically, the traffic arrival rates involved in the third case only have one peak-hour period over the whole day. We do this only for illustration to verify the efficacy of the proposed dynamic model. We assume that the traffic arrival rates have only one peak in the morning (around 7:30 AM) for scenario 1, while there is only one peak in the afternoon (around 4:00 PM) for scenario 2. In this case, the mean arrival rates along with the corresponding Poisson arrival processes for scenario 1 and scenario 2 are shown in Figure 5.5 and Figure 5.6, respectively.

Figure 5.18 shows the expected optimal values for the control variables $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$ throughout the whole day based on 300 (Monte Carlo) simulations. From the graphs, the distinction is very much evident between these two scenarios, especially for the more critical straight streams. For instance, the only peak of γ_4 arises during the morning peak-hour period in scenario 1 while it appears during the afternoon peak-hour period in scenario 2

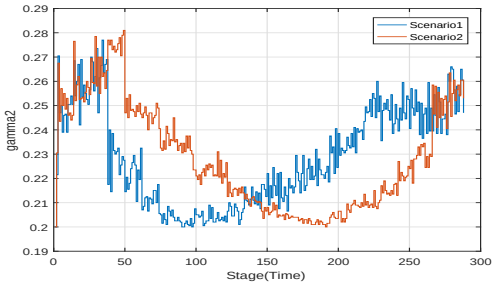
due to the different rush hour periods in those two scenarios. Similar results can be also observed for the cross traffic with respect to γ_1 and γ_3.

Figure 5.19 shows the expected optimal values for CBL settings (ℓ_3, ℓ_4, ℓ_7 and ℓ_8) corresponding to the four straight streams. It can be seen that each CBL stays at the high level around midnight because there is usually no congestion at that time. It is clear that each optimal path of these CBL settings has only one peak corresponding to the only rush hour period. For scenario 1, this occurs in the morning, while for scenario 2 it happens in the afternoon.

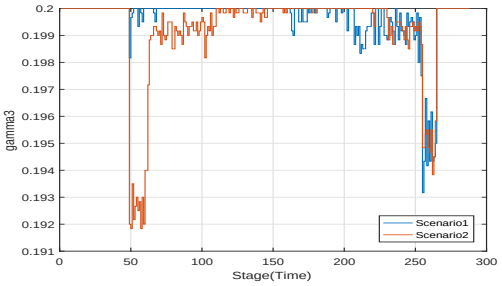
We also studied the convergency of the optimal trajectories of the decision variables $\{\gamma_1, \gamma_2, \gamma_3, \gamma_4\}$ and $\{\ell_3, \ell_4, \ell_7, \ell_8\}$. The results for scenario 1 are shown in Figure 5.20 and Figure 5.21 while in Figure 5.22 and Figure 5.23 for scenarios 2 based on three hundred Monte Carlo simulations. It can be clearly seen that the optimal paths are converging with the increase of the number of experiments (20, 40, 60, 80, 100, 140, 200).



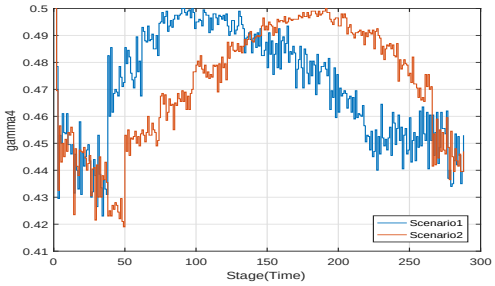
(a) Optimal path for γ_1



(b) Optimal path for γ_2

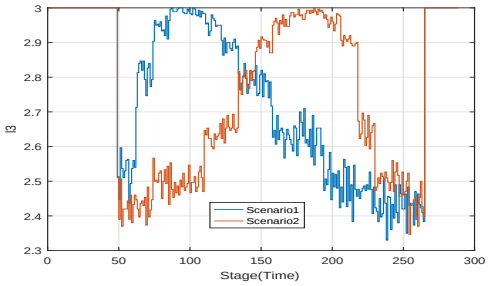


(c) Optimal path for γ_3

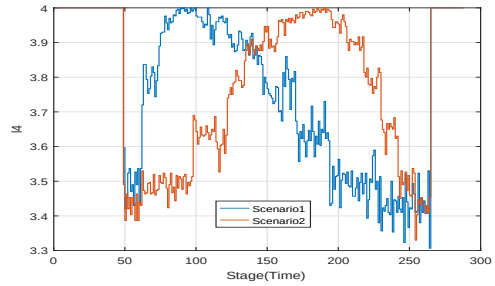


(d) Optimal path for γ_4

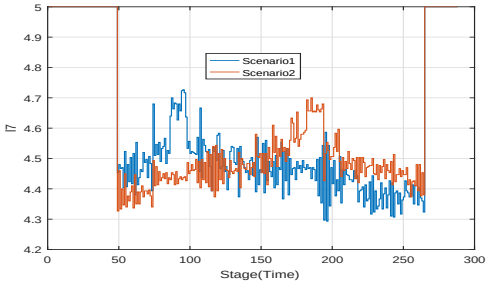
Figure 5.18: Optimal values for γ_1, γ_2, γ_3 and γ_4 (Case 3).



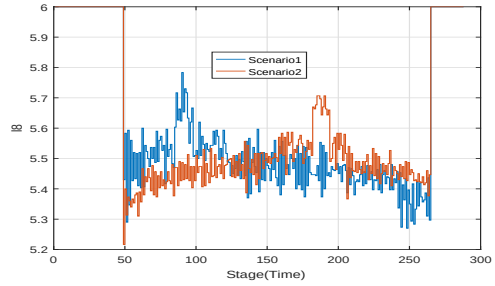
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4

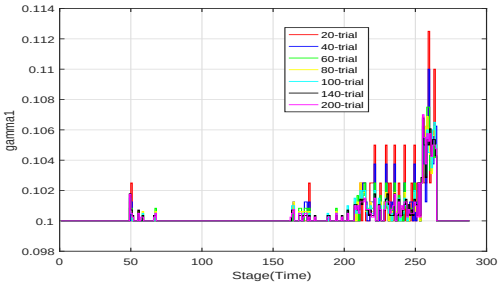


(c) Optimal path for ℓ_7

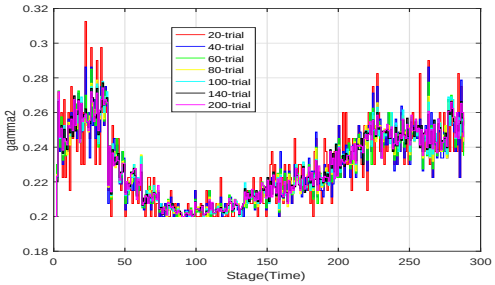


(d) Optimal path for ℓ_8

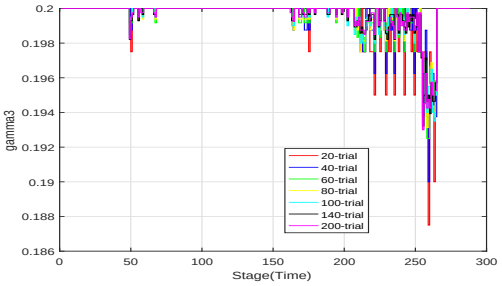
Figure 5.19: Optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (Case 3).



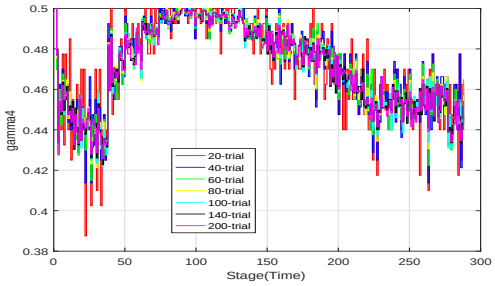
(a) Optimal path for γ_1



(b) Optimal path for γ_2

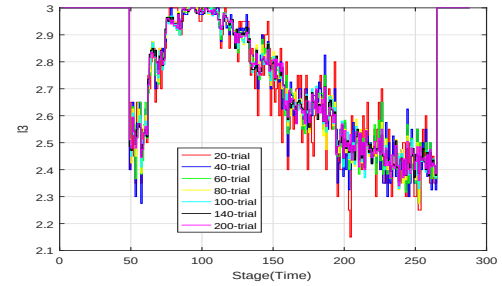


(c) Optimal path for γ_3

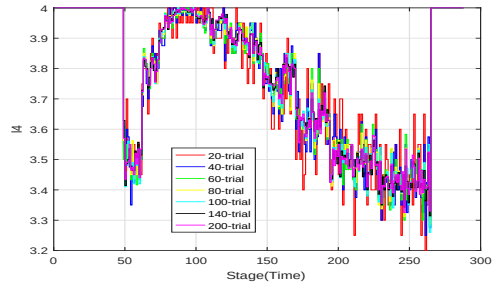


(d) Optimal path for γ_4

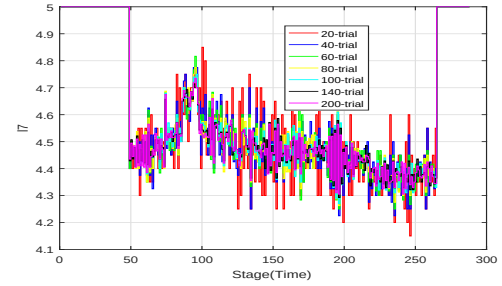
Figure 5.20: A set of optimal values for γ_1, γ_2, γ_3 and γ_4 (scenario 1).



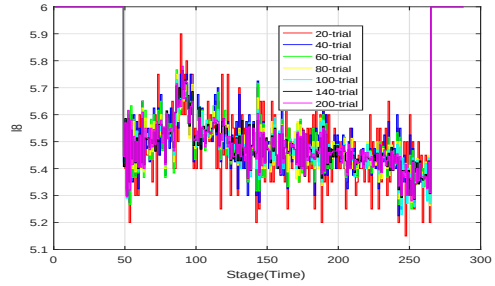
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4

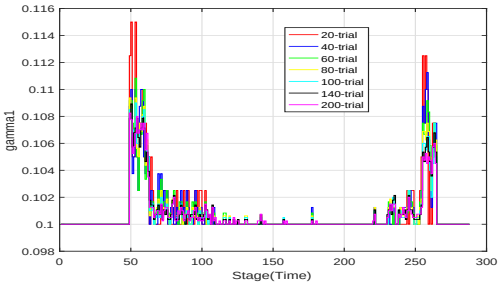


(c) Optimal path for ℓ_7

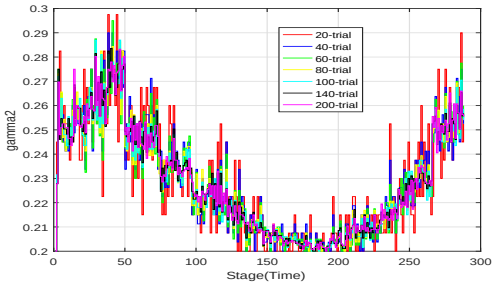


(d) Optimal path for ℓ_8

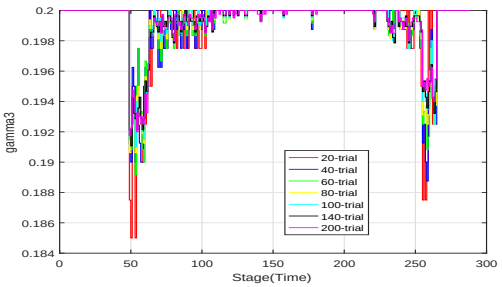
Figure 5.21: A set of optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (scenario 1).



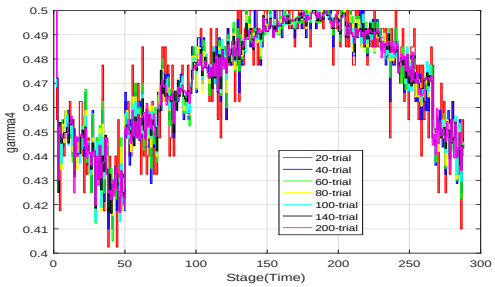
(a) Optimal path for γ_1



(b) Optimal path for γ_2

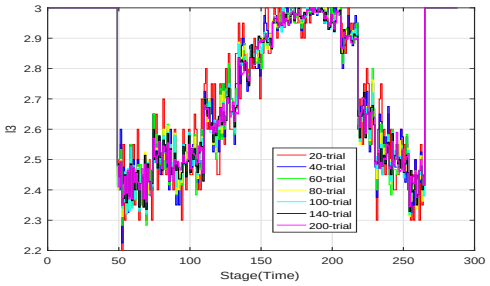


(c) Optimal path for γ_3

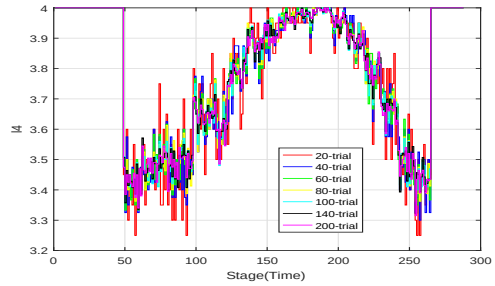


(d) Optimal path for γ_4

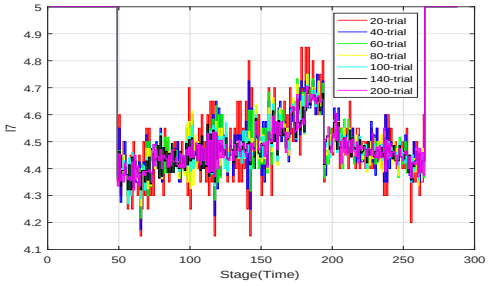
Figure 5.22: A set of optimal values for γ_1, γ_2, γ_3 and γ_4 (scenario 2).



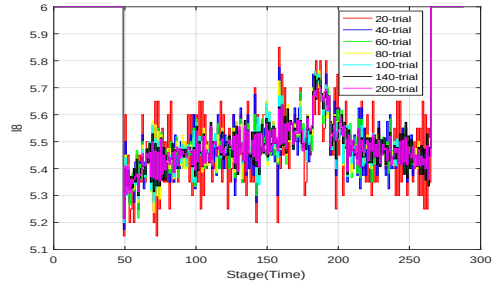
(a) Optimal path for ℓ_3



(b) Optimal path for ℓ_4



(c) Optimal path for ℓ_7



(d) Optimal path for ℓ_8

Figure 5.23: A set of optimal values for ℓ_3, ℓ_4, ℓ_7 and ℓ_8 (scenario 2).

Chapter 6

Conclusion

In this thesis, a stochastic dynamic (mathematical) model for urban traffic flow in a typical intersection has been proposed along with a methodology for determining the optimal policies designed to maximize throughput, minimize delay and avoid traffic congestions. The system is simulated with inputs described by several distinct nonhomogeneous Poisson processes. The optimization results for time allocation and CBL settings are found for three different cases by using the algorithm presented in Chapter 4. The algorithm presented in this thesis can be implemented on computer chips using microelectronic devices, loop detectors and video cameras etc. In order to apply this technique in any given city, one should select a set of major intersections and collect the necessary statistics (data) to implement the above algorithm. It is expected that the system proposed here will improve the overall traffic flow quality in the city.

6.1 Summary of Work

First of all, in Chapter 1 we introduce some fundamental concepts regarding urban traffic management, such as urban traffic control (UTC), intelligent transportation system (ITS), etc. By following this, the related works are discussed. In the meanwhile, we introduce two well-known techniques for solving optimal control and decision problems: maximum principle (MP) and dynamic programming (DP). Our contributions related to the thesis are summarized and organization of the thesis is also given at the end of this chapter.

In Chapter 2, we start with some discussions of stochastic processes. Then the stochastic traffic model is developed based on a specific stochastic process - nonhomogeneous Poisson random process. Hereafter, the traffic flow dynamics involved in an intersection is modeled giving four pairs of stochastic equations. There are two pairs of equations for straight traffic and another two for cross traffic.

We formulate the traffic flow optimization problem in Chapter 3 based on the dynamic model proposed in the preceding chapter. In order to apply the technique of dynamic programming, we convert the optimization problem into its compact form with the objective of increasing traffic throughput and reducing possible congestion as well as waiting time.

The optimization problem formulated in this chapter involves compact form of the system dynamics, parametric (or control) constraints and objective functional. In addition, the Monte Carlo method is also introduced here for the experimental simulations in Chapter 5.

In Chapter 4, we clarify the suitability of dynamic programming in addressing the optimization problem formulated in the thesis. It is clear that there are some indicator functions required for the construction of the dynamic model and they are not differentiable in the usual sense. Thus, the maximum principle is not suitable to solve the problem since it requires differentiability of the vector field G. However, dynamic programming method does not require this property and therefore DP is the more suitable technique for the optimization problem stated in the thesis. Based on the principle of optimality due to Bellman, we then put forward the sequential (or recursive) algorithm to address the formulated optimization problem.

In Chapter 5, the recursive algorithm is applied to solve the optimization problem proposed in the preceding chapter. The stochastic dynamic system is simulated with a series of numerical experiments having inputs described by several distinct nonhomogeneous Poisson processes. We have carried out experiments in terms of three different cases: traffic arrival rates with two ordinary peaks, traffic arrival rates with short morning and long afternoon peak, traffic arrival rates with only one peak. The corresponding optimization problems are solved using the proposed algorithm giving the optimal feedback control laws. Accordingly, for any given initial condition, the corresponding optimal cost can be found.

This thesis proposes a stochastic dynamic model for urban traffic management based on the typical intersection. We introduce the congestion broadcast level (CBL) for the straight traffic involved in the intersection and formulate the optimization problem based on the dynamics of eight traffic streams. Then the optimization problem is solved by use of dynamic programming giving the optimum strategies for time allocation and CBL settings for three different cases. In order to apply this technique in any given city, one should select a set of major intersections and collect the necessary statistics to implement the proposed algorithm. It is expected that this system will improve the overall traffic flow quality in the city.

6.2 Future Work

In this thesis, we did not focus on the size of each vehicle (assume they have the same average size) since we are more focused on the number of vehicles involved in the system dynamics. For simplicity, we did not take into account the CBL settings for the four cross streams. In the future, more research can be done to include the optimization of the CBL settings for both straight and cross traffic. Besides, pedestrians can be also taken into account in the future research. This requires the collection of historical data for the arrival rates of pedestrians in the intersection under consideration. In addition, we can also investigate some other schemes of the traffic light timing plan. Further more, using similar approach proposed here, it may be interesting to build a dynamic model of the traffic network with effective coordination among different intersections.

References

[1] CHINADAILY. Cities to suffer 'urban diseases'. http://www.chinadaily.com.cn/china/2012-02/09/content_14572500.htm, 2012. [Online; retrieved 19-May-2017].

[2] Y. Zhang. Traffic Congestion in Beijing: What to Do? http://www.china.org.cn/english/2002/Mar/28866.htm, 2002. [Online; retrieved 20-May-2017].

[3] CHINADAILY. Beijing commuters lose $127 a month on average due to congestion. http://www.chinadaily.com.cn/china/2015-11/25/content_22517778.htm, 2015. [Online; retrieved 22-May-2017].

[4] TomTom Traffic Index. Traffic congestion statistics for New York based on TomTom's historical database for 2016. https://www.tomtom.com/en_gb/trafficindex/city/new-york, 2016. [Online; retrieved 25-May-2017].

[5] G. Schifman. Congestion cost New York $16.9 billion last year. http://www.crainsnewyork.com/article/20170224/TRANSPORTATION/170229928/congestion-cost-new-york-16-9-billion-in-2016, 2017. [Online; retrieved 30-May-2017].

[6] X. Du. Beijing takes action to ease its thick traffic congestion. http://www.chinadaily.com.cn/business/motoring/2016-01/04/content_22920572.htm, 2016. [Online; retrieved 2-June-2017].

[7] TomTom Traffic Index. Traffic congestion statistics for Beijing based on TomTom's historical database for 2016. https://www.tomtom.com/en_gb/trafficindex/city/beijing, 2016. [Online; retrieved 5-June-2017].

[8] M. Rouse. Intelligent Transportation System (ITS). http://whatis.techtarget.com/definition/intelligent-transportation-system, 2017. [Online; retrieved 8-June-2017].

[9] Wikipedia. Dynamic programming. https://en.wikipedia.org/wiki/Dynamic_programming, 2017. [Online; retrieved 12-July-2017].

[10] Wikipedia. Maximum principle. https://en.wikipedia.org/wiki/Maximum_principle, 2017. [Online; retrieved 13-July-2017].

[11] Wikipedia. Monte Carlo method. https://en.wikipedia.org/wiki/Monte_Carlo_method, 2017. [Online; retrieved 20-July-2017].

[12] Nasir U. Ahmed. *Elements of finite dimensional systems and control theory*. John Wiley & Sons, Inc., 1988.

[13] Asim J Al-Khalili. "Urban traffic control - A general approach." *IEEE transactions on systems, man, and cybernetics*, (2):260-271, 1985.

[14] Richard Bellman. *Dynamic Programming*. Courier Corporation, 2013.

[15] Dimitri P. Bertsekas. *Dynamic programming and optimal control*, volume 2. Athena Scientific, Belmont, MA, 2007.

[16] Shukai Chen and Daniel Jian Sun. "An improved adaptive signal control method for isolated signalized intersection based on dynamic programming." *IEEE Intelligent Transportation Systems Magazine*, 8(4):4-14, 2016.

[17] Yujie Dai and Dongbin Zhao. "A traffic signal control algorithm for isolated intersections based on adaptive dynamic programming." In *Networking, Sensing and Control (ICNSC), 2010 International Conference on*, pages 255-260. IEEE, 2010.

[18] Arie Dubi. *Monte Carlo applications in systems engineering*. Wiley, 2000.

[19] Geoffrey Grimmett and David Stirzaker. *Probability and random processes*. Oxford university press, 2001.

[20] Rein Luus. "Iterative dynamic programming: from curiosity to a practical optimization procedure." *Control and Intelligent Systems*, 26(1):1, 1998.

[21] Pitu Mirchandani and Larry Head. "A real-time traffic signal control system: architecture, algorithms, and analysis." *Transportation Research Part C: Emerging Technologies*, 9(6):415-432, 2001.

[22] George L. Nemhauser. *Introduction to Dynamic Programming*. JohnWiley & Sons, New York, 1966.

[23] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC , 1987.

[24] Mustazibur Rahman, Nasir Uddin Ahmed, and Hussein T. Mouftah. "City traffic management model using Wireless Sensor Networks." In *Electrical and Computer Engineering (CCECE), 2014 IEEE 27th Canadian Conference on*, pages 1-6. IEEE, 2014.

[25] Suvrajeet Sen and K Larry Head. "Controlled optimization of phases at an intersection." *Transportation science*, 31(1):5-17, 1997.

[26] Steven Shelby. "Single-intersection evaluation of real-time adaptive traffic signal control algorithms." *Transportation Research Record: Journal of the Transportation Research Board*, (1867):183-192, 2004.

[27] Yu A. Shreider. *The Monte Carlo method: the method of statistical trials*, volume 87, Elsevier, 2014.

[28] James C. Spall and Daniel C. Chin. "A model-free approach to optimal signal light timing for system-wide traffic control." In *Decision and Control, 1994., Proceedings of the 33rd IEEE Conference on*, Volume 2, pages 1868-1875. IEEE, 1994.

[29] Shi'an Wang and N.U. Ahmed. "Dynamic model of urban traffic and optimum management of its flow and congestion." *Dynamic Systems and Applications*, 26:575-588, 2017.

[30] Shi'an Wang and N.U. Ahmed. "Optimum Management of the Network of City Bus Routes Based on a Stochastic Dynamic Model." *Journal of Industrial and Management Optimization* (2018). (To appear)

[31] Shi'an Wang and N.U. Ahmed. "Stochastic Dynamic Model of City Bus Routes and Their Optimum Management." *Control Science and Systems Engineering (ICCSSE), 2018 4th International Conference on.* IEEE, 2018. (To appear)

[32] Shi'an Wang and N.U. Ahmed. "Dynamic Model of Bank Queuing System and Its Optimal Management." *Control Science and Systems Engineering (ICCSSE), 2018 4th International Conference on.* IEEE, 2018. (To appear)

[33] Shi'an Wang, N.U. Ahmed and Tet Yeap. "Optimum Management of Urban Traffic Flow Based on a Stochastic Dynamic Model." *IEEE Transactions on Intelligent Transportation Systems.* (Submitted)

[34] X-H Yu and Wilfred W. Recker. "Stochastic adaptive control model for traffic signal systems." *Transportation Research Part C: Emerging Technologies*, 14(4):263-282, 2006.

[35] Xu Zhang and Thomas Riedel. "Urban traffic control: present and the future." *International Journal of Urban Sciences*, 21(sup1):87-100, 2017.

APPENDICES

Appendix A: Derivations of Formulas

Appendix B: Traffic Arrivals Source Code

Appendix C: Source Codes of the Simulation Programs

Appendix A

Derivations of Formulas

1. To derive the equality (2.6).

$$\begin{aligned}
\mathbf{E}\{p((t_k, t_{k+1}], \lambda_k)\} &= \sum_{n=0}^{\infty} n e^{-\lambda_k(t_{k+1}-t_k)} (\lambda_k(t_{k+1}-t_k))^n/n! \\
&= \sum_{n=0}^{\infty} e^{-\lambda_k(t_{k+1}-t_k)} (\lambda_k(t_{k+1}-t_k))^{n+1}/n! \\
&= e^{-\lambda_k(t_{k+1}-t_k)} \lambda_k(t_{k+1}-t_k) \sum_{n=0}^{\infty} (\lambda_k(t_{k+1}-t_k))^n/n! \\
&= e^{-\lambda_k(t_{k+1}-t_k)} \lambda_k(t_{k+1}-t_k) e^{\lambda_k(t_{k+1}-t_k)} \\
&= \lambda_k(t_{k+1}-t_k). \qquad \text{(A.1)}
\end{aligned}$$

2. To derive the equality (2.7).

$$\begin{aligned}
\mathbf{E}\{(p((t_k, t_{k+1}], \lambda_k))^2\} &= \sum_{n=0}^{\infty} n^2 e^{-\lambda_k(t_{k+1}-t_k)} (\lambda_k(t_{k+1}-t_k))^n/n! \\
&= e^{-\lambda_k(t_{k+1}-t_k)} \sum_{n=1}^{\infty} \frac{n}{(n-1)!} (\lambda_k(t_{k+1}-t_k))^n \\
&= e^{-\lambda_k(t_{k+1}-t_k)} \left\{ \sum_{n=2}^{\infty} \frac{(\lambda_k(t_{k+1}-t_k))^n}{(n-2)!} + \sum_{n=1}^{\infty} \frac{(\lambda_k(t_{k+1}-t_k))^n}{(n-1)!} \right\} \\
&= e^{-\lambda_k(t_{k+1}-t_k)} \left\{ (\lambda_k(t_{k+1}-t_k))^2 + \lambda_k(t_{k+1}-t_k) \right\} \sum_{n=0}^{\infty} \frac{(\lambda_k(t_{k+1}-t_k))^n}{n!} \\
&= e^{-\lambda_k(t_{k+1}-t_k)} \{ (\lambda_k(t_{k+1}-t_k))^2 + \lambda_k(t_{k+1}-t_k) \} e^{\lambda_k(t_{k+1}-t_k)} \\
&= (\lambda_k(t_{k+1}-t_k))^2 + \lambda_k(t_{k+1}-t_k). \qquad \text{(A.2)}
\end{aligned}$$

3. To derive the equality (2.8).
By applying equation A.1 and A.2 obtained above, we have

$$
\begin{aligned}
&\mathbf{E}\Big(p((t_k,t_{k+1}],\lambda_k)-\lambda_k(t_{k+1}-t_k)\Big)^2\\
&\quad=\mathbf{E}\Big((p((t_k,t_{k+1}],\lambda_k))^2-2\lambda_k(t_{k+1}-t_k)p((t_k,t_{k+1}],\lambda_k)+(\lambda_k(t_{k+1}-t_k))^2\Big)\\
&\quad=\mathbf{E}\{(p((t_k,t_{k+1}],\lambda_k))^2\}-2\lambda_k(t_{k+1}-t_k)\mathbf{E}\{p((t_k,t_{k+1}],\lambda_k)\}+(\lambda_k(t_{k+1}-t_k))^2\\
&\quad=\mathbf{E}\{(p((t_k,t_{k+1}],\lambda_k))^2\}-2(\lambda_k(t_{k+1}-t_k))^2+(\lambda_k(t_{k+1}-t_k))^2\\
&\quad=\mathbf{E}\{(p((t_k,t_{k+1}],\lambda_k))^2\}-(\lambda_k(t_{k+1}-t_k))^2\\
&\quad=\mathbf{E}\{(p((t_k,t_{k+1}],\lambda_k))^2\}-(\mathbf{E}\{p((t_k,t_{k+1}],\lambda_k)\})^2\\
&\quad=[(\lambda_k(t_{k+1}-t_k))^2+\lambda_k(t_{k+1}-t_k)]-(\lambda_k(t_{k+1}-t_k))^2\\
&\quad=\lambda_k(t_{k+1}-t_k).
\end{aligned}
\tag{A.3}
$$

Appendix B

Traffic Arrivals Source Code

```
%% First scenario: Poisson traffic arrival rates with two ordinary peaks.

% Sub-scenario (early peaks) of the first scenario

% Traffic streams related to R1
r1 = [poissrnd(1,36,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(4,12,1); poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(1,36,1)];

r2 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,24,1); poissrnd(2,12,1);
      poissrnd(3,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(3,12,1);
      poissrnd(2,24,1); poissrnd(5,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,24,1); poissrnd(1,12,1)];

r3 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(2,24,1); poissrnd(4,12,1);
      poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(8,12,1); poissrnd(7,12,1);
      poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(6,12,1);
      poissrnd(5,12,1); poissrnd(8,12,1); poissrnd(9,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1)];

r4 = [poissrnd(2,12,1); poissrnd(1,12,1); poissrnd(2,24,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(8,12,1);
      poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1)];
```

```
% Traffic streams related to R2
r5 = [poissrnd(1,36,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(5,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(1,36,1)];

r6 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,24,1); poissrnd(2,12,1);
      poissrnd(4,12,1); poissrnd(7,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,24,1);
      poissrnd(1,12,1)];

r7 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,12,1); poissrnd(2,12,1);
      poissrnd(3,12,1); poissrnd(6,12,1); poissrnd(10,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(5,12,1); poissrnd(8,12,1); poissrnd(10,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,12,1)];

r8 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(10,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(8,12,1);
      poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(8,12,1); poissrnd(10,12,1);
      poissrnd(9,12,1); poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1)];

plot(1:288,r1,'-.r^');
hold on;
plot(1:288,r2,'-.k^');
hold on;
plot(1:288,r3,'-ro');
hold on;
plot(1:288,r4,'-ko');
hold on;
plot(1:288,r5,':b*');
hold on;
plot(1:288,r6,':g*');
hold on;
plot(1:288,r7,'-b+');
hold on;
plot(1:288,r8,'-g+');
hold on;
xlabel('Stage(Time)');
```

```
ylabel('Poisson arrival rate');


%% Second scenario: Poisson traffic arrivals with short morning and long afternoon peak

% Traffic streams related to R1
r1 = [poissrnd(1,36,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(4,12,1); poissrnd(5,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(1,36,1)];

r2 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,24,1); poissrnd(2,12,1);
      poissrnd(3,12,1); poissrnd(6,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(2,12,1); poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(5,12,1); poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,24,1);
      poissrnd(1,12,1)];

r3 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(2,24,1); poissrnd(4,12,1);
      poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(7,12,1); poissrnd(6,12,1);
      poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(9,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1)];

r4 = [poissrnd(2,12,1); poissrnd(1,12,1); poissrnd(2,24,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(7,12,1);
      poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(7,12,1); poissrnd(6,12,1);
      poissrnd(5,12,1); poissrnd(7,12,1); poissrnd(8,12,1); poissrnd(9,12,1);
      poissrnd(9,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1)];

% Traffic streams related to R2
r5 = [poissrnd(1,36,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(6,12,1); poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(1,36,1)];

r6 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,24,1); poissrnd(2,12,1);
      poissrnd(4,12,1); poissrnd(7,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(3,12,1); poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(5,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,24,1);
```

```
      poissrnd(1,12,1)];

r7 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,12,1); poissrnd(2,12,1);
      poissrnd(3,12,1); poissrnd(6,12,1); poissrnd(10,12,1); poissrnd(7,12,1);
      poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(7,12,1);
      poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(8,12,1); poissrnd(10,12,1);
      poissrnd(9,12,1); poissrnd(8,12,1); poissrnd(6,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,12,1)];

r8 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(10,12,1); poissrnd(7,12,1);
      poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(7,12,1);
      poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(10,12,1);
      poissrnd(9,12,1); poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1)];

plot(1:288,r1,'-.r^');
hold on;
plot(1:288,r2,'-.k^');
hold on;
plot(1:288,r3,'-ro');
hold on;
plot(1:288,r4,'-ko');
hold on;
plot(1:288,r5,':b*');
hold on;
plot(1:288,r6,':g*');
hold on;
plot(1:288,r7,'-b+');
hold on;
plot(1:288,r8,'-g+');
hold on;
xlabel('Stage(Time)');
ylabel('Poisson arrival rate');


%% Third scenario: Poisson traffic arrival rates with one peak.

% Sub-scenario (morning peak) of the third scenario

% Traffic streams related to R1
r1 = [poissrnd(1,36,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(4,12,1);
      poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(2,12,1);
```

```
      poissrnd(2,12,1); poissrnd(1,12,1); poissrnd(1,36,1)];

r2 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(2,24,1); poissrnd(4,12,1);
      poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(5,12,1);
      poissrnd(5,24,1); poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,12,1)];

r3 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(5,12,1);
      poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(1,12,1)];

r4 = [poissrnd(2,12,1); poissrnd(1,12,1); poissrnd(2,24,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(5,12,1); poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(1,12,1)];

% Traffic streams related to R2
r5 = [poissrnd(1,36,1); poissrnd(2,12,1); poissrnd(3,12,1); poissrnd(4,12,1);
      poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(1,36,1)];

r6 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(2,24,1); poissrnd(4,12,1);
      poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(2,24,1);
      poissrnd(1,12,1)];

r7 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(3,12,1); poissrnd(4,12,1);
      poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(10,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(1,12,1)];

r8 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(3,12,1); poissrnd(5,12,1);
      poissrnd(7,12,1); poissrnd(8,12,1); poissrnd(10,12,1); poissrnd(9,12,1);
```

```
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(7,12,1); poissrnd(6,12,1);
      poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1); poissrnd(1,12,1)];

plot(1:288,r1,'-.r^');
hold on;
plot(1:288,r2,'-.k^');
hold on;
plot(1:288,r3,'-ro');
hold on;
plot(1:288,r4,'-ko');
hold on;
plot(1:288,r5,':b*');
hold on;
plot(1:288,r6,':g*');
hold on;
plot(1:288,r7,'-b+');
hold on;
plot(1:288,r8,'-g+');
hold on;
xlabel('Stage(Time)');
ylabel('Poisson arrival rate');
```

Appendix C

Source Codes of the Simulation Programs

```
%%300 Monte Carlo Simulations
%%Sub-scenario (early peaks) of the first scenario
clear; clc;
%% Initialization of the system
N = 287; %Number of stage
J = 5; %Length of each time interval
C1 = 1; C2 = 2;
C3 = 3; C4 = 4;
C5 = 2; C6 = 3;
C7 = 5; C8 = 6; %Capacity of each road segment
l1 = 1; l2 = 2;
l5 = 2; l6 = 3; %Fixed CBLs for cross streams
theta1 = 0.2; theta2 = 0.2;
theta3 = 0.1; theta4 = 0.1;
theta5 = 0.2; theta6 = 0.2;
theta7 = 0.1; theta8 = 0.1; %Reduced arrival ratio
beta1 = 1; beta2 = 1;
beta3 = 2; beta4 = 2;
beta5 = 1; beta6 = 1;
beta7 = 2; beta8 = 2; %Service capacity of each road segment
W1 = 50; W2 = 100;
W3 = 300; W4 = 350;
W5 = 100; W6 = 150;
W7 = 360; W8 = 400; %Weight given to the congestion
V1 = 10; V2 = 15;
V3 = 30; V4 = 40;
V5 = 15; V6 = 20;
V7 = 50; V8 = 60; %Weight given to the waiting time
a = 30240; b = 10080;
```

```
c = 2520; d = 504;
e = 168; f = 42;
g = 7; %Number of combinations for different states
a_k = [zeros(48,1); ones(216,1); zeros(24,1)]; %Weight for time
x = zeros(8,288); %Create the empty matrix for the states
l3_sum = zeros(288,301);
l4_sum = zeros(288,301);
l7_sum = zeros(288,301);
l8_sum = zeros(288,301);
gamma3_sum = zeros(288,301);
gamma4_sum = zeros(288,301);
l3_average = zeros(288,1);
l4_average = zeros(288,1);
l7_average = zeros(288,1);
l8_average = zeros(288,1);
gamma1_average = zeros(288,1);
gamma2_average = zeros(288,1);
gamma3_average = zeros(288,1);
gamma4_average = zeros(288,1); %Create empty matrix
A = zeros(1587600,288);
B = zeros(1587600,288);
C = zeros(1587600,288);
D = zeros(1587600,288);
U = zeros(1587600,288);
V = zeros(1587600,288); %Store optimal control values
x(:,1) = [0; 1; 1; 2; 0; 1; 1; 2]; %Initial state
k = N-1;
s = 1;


for t = 1:300 %Run Monte Carlo Simulation
% Traffic streams related to R1
r1 = [poissrnd(1,36,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(4,12,1); poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(1,36,1)];

r2 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,24,1); poissrnd(2,12,1);
      poissrnd(3,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(3,12,1);
      poissrnd(2,24,1); poissrnd(5,12,1); poissrnd(3,12,1); poissrnd(2,12,1);
      poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,24,1); poissrnd(1,12,1)];

r3 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(2,24,1); poissrnd(4,12,1);
```

```
      poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(8,12,1); poissrnd(7,12,1);
      poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(6,12,1);
      poissrnd(5,12,1); poissrnd(8,12,1); poissrnd(9,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(2,12,1)];

r4 = [poissrnd(2,12,1); poissrnd(1,12,1); poissrnd(2,24,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(7,12,1); poissrnd(9,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(8,12,1);
      poissrnd(6,12,1); poissrnd(6,12,1); poissrnd(8,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1)];

% Traffic streams related to R2
r5 = [poissrnd(1,36,1); poissrnd(1,24,1); poissrnd(2,12,1); poissrnd(3,12,1);
      poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(4,12,1); poissrnd(3,12,1);
      poissrnd(2,12,1); poissrnd(5,12,1); poissrnd(3,12,1); poissrnd(3,12,1);
      poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(1,36,1)];

r6 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,24,1); poissrnd(2,12,1);
      poissrnd(4,12,1); poissrnd(7,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(5,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(4,12,1); poissrnd(6,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(3,12,1); poissrnd(2,24,1);
      poissrnd(1,12,1)];

r7 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,12,1); poissrnd(2,12,1);
      poissrnd(3,12,1); poissrnd(6,12,1); poissrnd(10,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(8,12,1);
      poissrnd(7,12,1); poissrnd(5,12,1); poissrnd(8,12,1); poissrnd(10,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(4,12,1);
      poissrnd(3,12,1); poissrnd(2,12,1); poissrnd(2,12,1)];

r8 = [poissrnd(2,12,1); poissrnd(1,24,1); poissrnd(1,12,1); poissrnd(3,12,1);
      poissrnd(5,12,1); poissrnd(6,12,1); poissrnd(10,12,1); poissrnd(9,12,1);
      poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(6,12,1); poissrnd(8,12,1);
      poissrnd(6,12,1); poissrnd(5,12,1); poissrnd(8,12,1); poissrnd(10,12,1);
      poissrnd(9,12,1); poissrnd(8,12,1); poissrnd(7,12,1); poissrnd(5,12,1);
      poissrnd(4,12,1); poissrnd(3,12,1); poissrnd(2,12,1)];

%Computation for the last stage
for x_11 = 0:1
  for x_12 = 0:2
    for x_13 = 0:3
```

```
      for x_14 = 0:4
        for x_21 = 0:2
          for x_22 = 0:3
            for x_23 = 0:5
              for x_24 = 0:6
                     i = 1;
                for l3 = 2:3
                  for l4 = 3:4
                    for l7 = 4:5
                      for l8 = 5:6
                        if x_11 >= l1
                           I1 = 1;
                          else I1 = 0;
                        end
                        if x_12 >= l2
                           I2 = 1;
                          else I2 = 0;
                        end
                        if x_13 >= l3
                           I3 = 1;
                          else I3 = 0;
                        end
                        if x_14 >= l4
                           I4 = 1;
                          else I4 = 0;
                        end

                        if x_21 >= l5
                           I5 = 1;
                          else I5 = 0;
                        end
                        if x_22 >= l6
                           I6 = 1;
                          else I6 = 0;
                        end
                        if x_23 >= l7
                           I7 = 1;
                          else I7 = 0;
                        end
                        if x_24 >= l8
                           I8 = 1;
                          else I8 = 0;
                        end

for gamma3 = 0.15:0.05:0.20
```

```
   for gamma4 = 0.35:0.05:0.50
         gama1 = 0.3 - gamma3;
         gama2 = 0.7 - gamma4;
     %Weighted congestion on R1
     congestion_R1 = W1.*(x_11 - l1).*I1 + W2.*(x_12 - l2).*I2
                  + W3.*(x_13 - l3).*I3 + W4.*(x_14 - l4).*I4;
     %Weighted congestion on R2
     congestion_R2 = W5.*(x_21 - l5).*I5 + W6.*(x_22 - l6).*I6
                  + W7.*(x_23 - l7).*I7 + W8.*(x_24 - l8).*I8;
     %Weighted waiting time from R1
     waitingtime_R1 = V1.*l1 + V2.*l2 + V3.*l3 + V4.*l4;
     %Weighted waiting time from R2
     waitingtime_R2 = V5.*l5 + V6.*l6 + V7.*l7 + V8.*l8;
     %Weighted throughput from R1
     throughput_R1 = (min(beta1,x_11) + min(beta2,x_12)).*gama1.*J
                  + (min(beta3,x_13) + min(beta4,x_14)).*gama2.*J;
     %Weighted throughput from R2
     throughput_R2 = (min(beta5,x_21) + min(beta6,x_22)).*gamma3.*J
                  + (min(beta7,x_23) + min(beta8,x_24)).*gamma4.*J;
     %Total weighted congestion
     congestion(i,1) = a_k(k+2,1).*(congestion_R1 + congestion_R2);
     %Total weighted waiting time
     waitingtime(i,1) = a_k(k+2,1).*(waitingtime_R1 + waitingtime_R2);
     %Total throughput in the intersection
     throughput(i,1) = throughput_R1 + throughput_R2;
     %Total cost function in the intersection
     cost(i,1) = congestion(i,1) + waitingtime(i,1) - throughput(i,1);
     i = i + 1;
   end
end
                     end
                   end
                 end
               end

%% Find the optimal controls

number = max(find(cost == min(cost))); %Find the minimum cost for each state
[l3_optimal,l4_optimal,l7_optimal,l8_optimal,gamma3_optimal,gamma4_optimal] = optimalco
A(s,288) = l3_optimal;
B(s,288) = l4_optimal;
C(s,288) = l7_optimal;
D(s,288) = l8_optimal;
U(s,288) = gamma3_optimal;
V(s,288) = gamma4_optimal;
```

```
I_min_cost(s,1) = min(cost);
s = s + 1;
              end
            end
          end
        end
      end
    end
  end
end


%Backward sweep for the optimal controls
while k >= 0
m = 1;
for x_11 = 0:1
  for x_12 = 0:2
    for x_13 = 0:3
      for x_14 = 0:4
        for x_21 = 0:2
          for x_22 = 0:3
            for x_23 = 0:5
              for x_24 = 0:6
                     j = 1;
                for l3 = 2:3
                  for l4 = 3:4
                    for l7 = 4:5
                      for l8 = 5:6
                        if x_11 >= l1
                              I1 = 1;
                           else I1 = 0;
                        end
                        if x_12 >= l2
                              I2 = 1;
                           else I2 = 0;
                        end
                        if x_13 >= l3
                              I3 = 1;
                           else I3 = 0;
                        end
                        if x_14 >= l4
                              I4 = 1;
                           else I4 = 0;
                        end
                        if x_21 >= l5
```

```
                              I5 = 1;
                           else I5 = 0;
                        end
                        if x_22 >= l6
                              I6 = 1;
                           else I6 = 0;
                        end
                        if x_23 >= l7
                              I7 = 1;
                           else I7 = 0;
                        end
                        if x_24 >= l8
                              I8 = 1;
                           else I8 = 0;
                        end
for gamma3 = 0.15:0.05:0.20
   for gamma4 = 0.35:0.05:0.50
         gama1 = 0.3 - gamma3;
         gama2 = 0.7 - gamma4;
     congestion_R1 = W1.*(x_11 - l1).*I1 + W2.*(x_12 - l2).*I2
                   + W3.*(x_13 - l3).*I3 + W4.*(x_14 - l4).*I4;
     congestion_R2 = W5.*(x_21 - l5).*I5 + W6.*(x_22 - l6).*I6
                   + W7.*(x_23 - l7).*I7 + W8.*(x_24 - l8).*I8;
     waitingtime_R1 = V1.*l1 + V2.*l2 + V3.*l3 + V4.*l4;
     waitingtime_R2 = V5.*l5 + V6.*l6 + V7.*l7 + V8.*l8;
     throughput_R1 = (min(beta1,x_11) + min(beta2,x_12)).*gama1.*J
                   + (min(beta3,x_13) + min(beta4,x_14)).*gama2.*J;
     throughput_R2 = (min(beta5,x_21) + min(beta6,x_22)).*gamma3.*J
                   + (min(beta7,x_23) + min(beta8,x_24)).*gamma4.*J;
     congestion(j,1) = a_k(k+1,1).*(congestion_R1 + congestion_R2);
     waitingtime(j,1) = a_k(k+1,1).*(waitingtime_R1 + waitingtime_R2);
     throughput(j,1) = throughput_R1 + throughput_R2;
     cost(j,1) = congestion(j,1) + waitingtime(j,1) - throughput(j,1);
     j = j + 1;
   end
end
                        end
                     end
                  end
               end

%% Find the optimal controls

number = max(find(cost == min(cost))); %Find the minimum cost for each state
[l3_optimal,l4_optimal,l7_optimal,l8_optimal,gamma3_optimal,gamma4_optimal] = optimalco
```

```
gama1_optimal = 0.3 - gamma3_optimal;
gama2_optimal = 0.7 - gamma4_optimal;%Follow the assumed time distribution
A(m,k+1) = l3_optimal;
B(m,k+1) = l4_optimal;
C(m,k+1) = l7_optimal;
D(m,k+1) = l8_optimal;
U(m,k+1) = gamma3_optimal;
V(m,k+1) = gamma4_optimal;%Store optimal control values

%% Find the next state

[x_11_next, number1] = Nextstate(x_11, r1(k+1,1), gama1_optimal, l1, C1);
[x_12_next, number2] = Nextstate(x_12, r2(k+1,1), gama1_optimal, l2, C2);
[x_13_next, number3] = Nextstate(x_13, r3(k+1,1), gama2_optimal, l3, C3);
[x_14_next, number4] = Nextstate(x_14, r4(k+1,1), gama2_optimal, l4, C4);
[x_21_next, number5] = Nextstate(x_21, r5(k+1,1), gama3_optimal, l5, C5);
[x_22_next, number6] = Nextstate(x_22, r6(k+1,1), gama3_optimal, l6, C6);
[x_23_next, number7] = Nextstate(x_23, r7(k+1,1), gama4_optimal, l7, C7);
[x_24_next, number8] = Nextstate(x_24, r8(k+1,1), gama4_optimal, l8, C8);

%% Find the optimal cost for each state at a specific stage

running_cost(m,1) = min(cost);
position = a.*number1 + b.*number2 + c.*number3 + d.*number4
         + e.*number5 + f.*number6 + g.*number7 + number8 + 1;%Locate the state
I_min_cost(m,1) = running_cost(m,1) + I_min_cost(position,1);%Calculate the cost
m = m + 1;%Go to next state combination
              end
            end
          end
        end
      end
    end
  end
end
k = k - 1;%Go to the previous stage
end

%% Forward sweep for the optimal trajectory
%% with the given initial condition
for n = 1:288
  x(1,n) = State(x(1,n), C1);
  x(2,n) = State(x(2,n), C2);
  x(3,n) = State(x(3,n), C3);
  x(4,n) = State(x(4,n), C4);
```

```
  x(5,n) = State(x(5,n), C5);
  x(6,n) = State(x(6,n), C6);
  x(7,n) = State(x(7,n), C7);
  x(8,n) = State(x(8,n), C8);
%Locate the state
u_position = a.*x(1,n) + b.*x(2,n) + c.*x(3,n) + d.*x(4,n)
           + e.*x(5,n) + f.*x(6,n) + g.*x(7,n) + x(8,n) + 1;

l3_trajectory(n,1) = A(u_position,n);
l4_trajectory(n,1) = B(u_position,n);
l7_trajectory(n,1) = C(u_position,n);
l8_trajectory(n,1) = D(u_position,n);
gamma3_trajectory(n,1) = U(u_position,n);
gamma4_trajectory(n,1) = V(u_position,n);
gamma3_value = U(u_position,n);
gamma4_value = V(u_position,n);
gama1_value = 0.3 - gamma3_value;
gama2_value = 0.7 - gamma4_value;%Find the optimal controls
%Find the next optimal state
x(1,n+1) = Trajectory(x(1,n), l1, r1(n,1), gama1_value);
x(2,n+1) = Trajectory(x(2,n), l2, r2(n,1), gama1_value);
x(3,n+1) = Trajectory(x(3,n), l3_trajectory(n,1), r3(n,1), gama2_value);
x(4,n+1) = Trajectory(x(4,n), l4_trajectory(n,1), r4(n,1), gama2_value);
x(5,n+1) = Trajectory(x(5,n), l5, r5(n,1), gama3_value);
x(6,n+1) = Trajectory(x(6,n), l6, r6(n,1), gama3_value);
x(7,n+1) = Trajectory(x(7,n), l7_trajectory(n,1), r7(n,1), gama4_value);
x(8,n+1) = Trajectory(x(8,n), l8_trajectory(n,1), r8(n,1), gama4_value);
end
%Store the Monte Carlo simulation results
l3_sum(:,t+1) = l3_sum(:,t) + l3_trajectory;
l4_sum(:,t+1) = l4_sum(:,t) + l4_trajectory;
l7_sum(:,t+1) = l7_sum(:,t) + l7_trajectory;
l8_sum(:,t+1) = l8_sum(:,t) + l8_trajectory;
gamma3_sum(:,t+1) = gamma3_sum(:,t) + gamma3_trajectory;
gamma4_sum(:,t+1) = gamma4_sum(:,t) + gamma4_trajectory;

end
%Take the average as the final results
l3_average = (1/300).*l3_sum(:,301);
l4_average = (1/300).*l4_sum(:,301);
l7_average = (1/300).*l7_sum(:,301);
l8_average = (1/300).*l8_sum(:,301);
gamma3_average = (1/300).*gamma3_sum(:,301);
gamma4_average = (1/300).*gamma4_sum(:,301);
gamma1_average = 0.3.*ones(288,1) - gamma3_average;
```

```
gamma2_average = 0.7.*ones(288,1) - gamma4_average;
%Plot the optimal trajectory of gamma4(for example)
plot(1:288,gamma4_average);
xlabel('Stage(Time)');
ylabel('gamma4');
%% M-Files that include different functions
%% M-File: Function to find the optimal controls

function [l3, l4, l7, l8, gamma3, gamma4] = optimalcontrol(X)
%Find the optimal controls for each state

                if X > 0 && X <= 4
                    l3_optimal = 2;
                    l4_optimal = 3;
                    l7_optimal = 4;
                    l8_optimal = 5;
                    gamma3_optimal = 0.15;
                    gamma4_optimal = 0.05.*X + 0.3;
                  elseif X <= 8
                    l3_optimal = 2;
                    l4_optimal = 3;
                    l7_optimal = 4;
                    l8_optimal = 5;
                    gamma3_optimal = 0.20;
                    gamma4_optimal = 0.05.*(X - 4) + 0.3;
                  elseif X <= 16
                    l3_optimal = 2;
                    l4_optimal = 3;
                    l7_optimal = 4;
                    l8_optimal = 6;
                    if mod(X,8) > 0 && mod(X,8) < 5
                        gamma3_optimal = 0.15;
                        gamma4_optimal = 0.05.*(X - 8) + 0.3;
                    else
                        gamma3_optimal = 0.20;
                        gamma4_optimal = 0.05.*(X - 12) + 0.3;
                    end
                  elseif X <= 24
                    l3_optimal = 2;
                    l4_optimal = 3;
                    l7_optimal = 5;
                    l8_optimal = 5;
                    if mod(X,16) > 0 && mod(X,16) < 5
                        gamma3_optimal = 0.15;
                        gamma4_optimal = 0.05.*(X - 16) + 0.3;
```

```
    else
        gamma3_optimal = 0.20;
        gamma4_optimal = 0.05.*(X - 20) + 0.3;
    end
elseif X <= 32
    l3_optimal = 2;
    l4_optimal = 3;
    l7_optimal = 5;
    l8_optimal = 6;
    if mod(X,24) > 0 && mod(X,24) < 5
        gamma3_optimal = 0.15;
        gamma4_optimal = 0.05.*(X - 24) + 0.3;
    else
        gamma3_optimal = 0.20;
        gamma4_optimal = 0.05.*(X - 28) + 0.3;
    end
elseif X <= 64
    p = mod(X,32);
    l3_optimal = 2;
    l4_optimal = 4;
    if p > 0 && p < 5
       l7_optimal = 4;
       l8_optimal = 5;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 32) + 0.3;
    elseif p > 4 && p < 9
       l7_optimal = 4;
       l8_optimal = 5;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 36) + 0.3;
    elseif p > 8 && p < 13
       l7_optimal = 4;
       l8_optimal = 6;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 40) + 0.3;
    elseif p > 12 && p < 17
       l7_optimal = 4;
       l8_optimal = 6;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 44) + 0.3;
    elseif p > 16 && p < 21
       l7_optimal = 5;
       l8_optimal = 5;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 48) + 0.3;
```

```
    elseif p > 20 && p < 25
       l7_optimal = 5;
       l8_optimal = 5;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 52) + 0.3;
    elseif p > 24 && p < 29
       l7_optimal = 5;
       l8_optimal = 6;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 56) + 0.3;
    else
       l7_optimal = 5;
       l8_optimal = 6;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 60) + 0.3;
    end
elseif X <= 96
    p = mod(X,64);
    l3_optimal = 3;
    l4_optimal = 3;
    if p > 0 && p < 5
       l7_optimal = 4;
       l8_optimal = 5;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 64) + 0.3;
    elseif p > 4 && p < 9
       l7_optimal = 4;
       l8_optimal = 5;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 68) + 0.3;
    elseif p > 8 && p < 13
       l7_optimal = 4;
       l8_optimal = 6;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 72) + 0.3;
    elseif p > 12 && p < 17
       l7_optimal = 4;
       l8_optimal = 6;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 76) + 0.3;
    elseif p > 16 && p < 21
       l7_optimal = 5;
       l8_optimal = 5;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 80) + 0.3;
```

```
    elseif p > 20 && p < 25
       l7_optimal = 5;
       l8_optimal = 5;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 84) + 0.3;
    elseif p > 24 && p < 29
       l7_optimal = 5;
       l8_optimal = 6;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 88) + 0.3;
    elseif p > 28 && p < 33
       l7_optimal = 5;
       l8_optimal = 6;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 92) + 0.3;
    end
elseif X <= 128
    p = mod(X,96);
    l3_optimal = 3;
    l4_optimal = 4;
    if p > 0 && p < 5
       l7_optimal = 4;
       l8_optimal = 5;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 96) + 0.3;
    elseif p > 4 && p < 9
       l7_optimal = 4;
       l8_optimal = 5;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 100) + 0.3;
    elseif p > 8 && p < 13
       l7_optimal = 4;
       l8_optimal = 6;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 104) + 0.3;
    elseif p > 12 && p < 17
       l7_optimal = 4;
       l8_optimal = 6;
       gamma3_optimal = 0.20;
       gamma4_optimal = 0.05.*(X - 108) + 0.3;
    elseif p > 16 && p < 21
       l7_optimal = 5;
       l8_optimal = 5;
       gamma3_optimal = 0.15;
       gamma4_optimal = 0.05.*(X - 112) + 0.3;
```

```
                    elseif p > 20 && p < 25
                       l7_optimal = 5;
                       l8_optimal = 5;
                       gamma3_optimal = 0.20;
                       gamma4_optimal = 0.05.*(X - 116) + 0.3;
                    elseif p > 24 && p < 29
                       l7_optimal = 5;
                       l8_optimal = 6;
                       gamma3_optimal = 0.20;
                       gamma4_optimal = 0.05.*(X - 120) + 0.3;
                    elseif p > 28 && p < 33
                       l7_optimal = 5;
                       l8_optimal = 6;
                       gamma3_optimal = 0.20;
                       gamma4_optimal = 0.05.*(X - 124) + 0.3;
                    end
                 end

end

%% M-File: Function to find the next state

function [y, z] = Nextstate(x, r, gamma, l, C)
      if x < 0
         x = 0;
         y = x + J.*r - min(beta1,x).*gamma.*J;
      elseif x < l
         y = x + J.*r - min(beta1,x).*gamma.*J;
      elseif x <= C
         y = x + J.*theta1.*r - min(beta1,x).*gamma.*J;
      else
         x = C;
         y = x + J.*theta1.*r - min(beta1,x).*gamma.*J;
      end

      if y < 0
         z = 0;
      elseif y <= C
         z = round(y);
      else
         z = C;
      end
end

%% M-File: Function to find the optimal trajectory
```

```
function [y] = State(x, C)
      if x < 0
         y = 0;
      elseif x > C
         y = C;
      end
end

function [y] = Trajectory(x, l, r, gamma)
      if x < l
         y = x + J.*r - min(beta1,x).*gamma.*J;
      else
         y = x + J.*theta1.*r - min(beta1,x).*gamma.*J;
      end
         y = round(y);
end
```