4-2016

Dynamic green split optimization in intersection signal design for urban street network

Peng Jiao
Purdue University

Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By PENG JIAO

Entitled
DYNAMIC GREEN SPLIT OPTIMIZATION IN INTERSECTION SIGNAL DESIGN FOR URBAN STREET NETWORK

For the degree of Master of Science in Civil Engineering

Is approved by the final examining committee:

SAMUEL LABI
Chair

KUMARES C. SINHA

KONSTANTINA GKRITZA

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): SAMUEL LABI

Approved by: DULCY M. ABRAHAM 4/22/2016

Head of the Departmental Graduate Program Date

DYNAMIC GREEN SPLIT OPTIMIZATION IN INTERSECTION SIGNAL DESIGN FOR URBAN STREET NETWORK

A Thesis

Submitted to the Faculty

of

Purdue University

by

Peng Jiao

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Civil Engineering

May 2016

Purdue University

West Lafayette, Indiana

To my Parents.

ACKNOWLEDGEMENTS

First, I am deeply grateful to my major professor, Dr. Samuel Labi, for his insight, guidance, and involvement in my graduate program and for continually challenging me to explore and excel in my research endeavors and my studies. Without his guidance and untiring help, this thesis would not have been possible. I also thank my committee members, Dr. Kumares Sinha and Dr. Nadia Gkritza, who were supportive in diverse ways. In addition, I am grateful to my undergraduate study mentor at Illinois Institute of Technology, Dr. Zongzhi Li, who provided significant help with the model application in the Chicago TRANSIMS platform, data collection, and processing and model runs. Dr. Li provided numerous insights into learning the fundamentals of signal timing design focusing on an isolated intersection, multiple intersections along one or more parallel corridors, and then extensive intersections within an urban street network, which eventually led to the methodology documented in this thesis.

I am forever grateful to my parents for their unconditional love, for their support in all the decisions I made, and for their encouragement when I encountered challenges in my graduate studies. I also sincerely thank my other family members who were always there in my time of need and provided emotional support. I am thankful to the faculty and staff of the School of Civil Engineering at Purdue University for maintaining an open and conducive intellectual environment and to my classmates and friends for their valuable friendship and support during my time at Purdue.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

LIST OF ABBREVIATIONS

Abbreviation	Definition
CMAP	Chicago Metropolitan Agency for Planning
DS	Degree of saturation
HCM	Highway Capacity Manual
LWR	Lighthill-Whitham-Richards (shockwave theory)
MOSTOP	Multi-objective signal timing optimization problem
OPAC	Optimization policies for adaptive control
QRNAT	Queuing delays and queue rear no-delay time
Sa-PSO	Simulated annealing-particle swarm optimization
SCOOT	Split cycle and offset optimizing technique
TOD	Time-of-day
VHT	Vehicle hours of travel

ABSTRACT

Jiao, Peng, M.S.C.E., Purdue University, May 2016. Dynamic Green Split Optimization in Intersection Signal Design for Urban Street Network. Major Professor: Samuel Labi.

In the past few decades, auto travel demand in the United States has significantly increased, but roadway capacity unfortunately has not expanded as quickly, which has led to severe levels of highway traffic congestion in many areas. In theory, the problem of congestion addressed through demand management and roadway expansion. However, system expansion in urban areas is difficult due to the extremely high cost of land; therefore, maximizing the existing capacity therefore often is considered the most realistic option. In urban areas, most of the traffic congestion and delays typically occur at signalized intersections. This thesis aims to prove the hypothesis that it is possible to increase capacity by establishing traffic signal timing plans that are more effective than existing plans. A new methodology is introduced in this thesis for dynamic green split optimization as a part of intersection signal-timing design to achieve maximized reduction in overall delay at all the intersections within an urban street network. The measurement of effectiveness in this new method is reduction in the average delay per vehicle per signal cycle. This thesis used data from 143 signalized intersections and 334 street segments in the Chicago Loop area street network to demonstrate the proposed methodology.

The results suggest that it is possible to reduce delay by approximately 35% through the optimization of signal green splits for the four-hour AM and four-hour PM peak periods of a typical day.

CHAPTER 1 INTRODUCTION

1.1 Background

Transportation systems help facilitate freight shipments and economic activities in regions and cities in ways that reflect the distribution of these activities, and urban productivity is closely related to effective usage of transportation systems. Highway traffic congestion is an issue of great concern in large and dense urban areas. Traffic congestion causes a waste of approximately seven billion hours of extra time and three million gallons of additional fuel in urban areas of the United States as reported in the *Urban Mobility Report* [TTI, 2015]. In theory, congestion problems can be resolved through demand management and roadway expansion. However, urban system expansion is typically difficult due to the extremely high cost of purchasing land in urban areas (Sinha and Labi, 2007). To address this issue, it is hypothesized that the utilization of available system capacity can be maximized. One of the most commonly-used palliatives for traffic mitigation is the design of traffic signal timings that assign time slots in an efficient manner. Traffic signals, which were first installed in London in 1868, have played a critical role in urban traffic control since then and have contributed greatly to urban traffic mobility and safety.

In densely populated cities, traffic congestion continues to grow as travel demand increases. While projects that increase the capacity of transportation facilities generally resolve the problem of congestion, the reality is that the construction of additional lanes is not always feasible due to the high cost of land in urban areas. Therefore, maximizing the utilization of existing capacity in the most efficient manner is the preferred approach, such as the development of signal timings that minimize delay.

1.2 Problem Statement

The mitigation of traffic congestion issues, especially related to intersection delays in dense urban areas with a large number of intersections, needs a new methodology for signal timing optimization that will dynamically adjust the green splits of individual phases for individual intersections without changing the existing cycle length and signal coordination. Minimizing the average vehicular delay per cycle over several consecutive cycles also should be a priority for this new method.

1.3 Study Objectives and Scope

Objectives: The general objective of this thesis is to optimize intersection signal design for urban street network and aims to accomplish the following:

- develop a method to calculate vehicle delays at signalized intersection in consecutive cycles under different traffic conditions (undersaturated and oversaturated);

- formulate a green split optimization model that will achieve minimum vehicle delays per intersection per cycle averaged over consecutive cycles with vehicle delays computed using the above method;
- develop an iterative computational process for a large number of intersections in an urban street network; and
- implement the proposed optimization model using a case study.

Study Scope: The proposed methodology will interface with and integrate into a large-scale, high-fidelity simulation-based traffic model to update green split designs based on dynamically assigned traffic using the intersection over fixed time intervals during the AM and PM periods. The proposed method will minimize the intersection delays in terms of the delays per vehicle per cycle averaged over several consecutive cycles.

1.4 Chapter Organization

This thesis consists of five chapters. Chapter 1 discusses the traffic congestion problem in urban areas and a description of the study objectives. Chapter 2 documents the findings of the review of the literature addressing intersection signal-timing optimization. Chapter 3 elaborates on the proposed methodology, and Chapter 4 presents the methodology's application and the results of the numerical analysis. Chapter 5 summarizes the contributions of this thesis and future research directions.

CHAPTER 2 LITERATURE REVIEW

The initial step of this thesis was a review of the literature pertaining to the current methodologies for signal timing optimization at urban intersections.

2.1 Studies on Intersection Vehicle-Delay Modeling

Macroscopic traffic flow models are rooted in mathematical relationships between traffic flow, density, and speed and are helpful because they provide a theoretical basis for the planning and design of efficient ways to increase highway capacity [Robert, 1998; Garber and Hoel, 2001]. With regard to urban intersections, in the past few decades, the shockwave models developed to better characterize traffic flow on road segments under various conditions at intersections have helped engineers to develop appropriate measures of effectiveness to increase the efficiency of intersection capacity.

Wirasinghe [1978] applied the traffic shockwave theory of Lighthill and Whitham to model the moving incidents associated with vehicle overtaking, and established a graphical method to derive the delays for individual or all vehicles and their related costs. The study also developed a new formulation to measure the upstream total delay arising from an incident downstream and demonstrated that the new formulation produced the same results as deterministic queuing theory.

Michalopoulo *et al*. [1981] studied a real-time signal control policy for minimizing total intersection delay subject to queue length constraints. The authors concluded that the shockwaves that occurred upstream of the stop lines were caused by irregular service of traffic at the signal. Based on this conclusion, they developed a new model and proposed a real-time signal control policy based on the model that managed the queue lengths of two conflicting streams through a traffic light controlled in time and space. Using the current pre-timed control policy at an intersection with a high volume of traffic as a comparison target of the proposed policy, the authors established that the proposed policy was more efficient, particularly under conditions where demand exceeded the saturation level.

In order to describe the characteristics of queues in coordinated traffic signal systems and the traffic wave motion that spreads from link to link, Hisai and Sasaki [1993] studied shockwaves to formulate a new model. Their work produced a visualization of the shockwave phenomenon as it spreads under various streets, traffic, and signal conditions, including both the undersaturated and oversaturated cases. The optimization of signal control timing can be studied using the Hisai and Sasaki model.

Dion *et al*. [2004] compared the delays calculated by the INTEGRATION microscopic traffic simulation model and the delays produced by analytical delay models for a one-lane approach to a pre-timed signalized intersection under undersaturated to oversaturated conditions. The analytical model used for the comparison represented the steady-state stochastic delay, time-dependent stochastic delay models, deterministic queuing, and shockwave. To evaluate the consistency of the calculated vehicle delays from the two models, they conducted a comparison over a range of volume-to-capacity (v/c) ratios extending from 0.1 to 1.4. Over this range, the delay models from the 1981 Australian Capacity Guide [Akçelik, 1980], the 1995 Canadian Capacity Guide for Signalized Intersections [ITE, 1995], the 1997 Highway Capacity Manual (HCM) [TRB, 1997], and nearly consistent delay estimates were produced from the INTEGRATION microscopic traffic simulation model. In this manner, the conditions were validated. In addition, the study recommended evaluation of such consistency for more complex situations.

A study conducted by Liu *et al*. [2009] presented a creative approach for assessing intersection queue lengths with existing detectors; and by using this methodology; it was possible to assess time-dependent queue lengths on signal links congested with long queues, which was a key contribution of their study. Moreover, it was possible to differentiate traffic states at an intersection by applying the Lighthill-Whitham-Richards (LWR) shockwave theory with high-resolution traffic signal data in order to assess the queue length under congested conditions. The authors evaluated three models by comparing the estimated maximum queue lengths with the ground count data recorded by cameras and human observers and confirmed that the basic model produced precise outputs (the other two outputs were also acceptable).

Wu *et al*. [2010] studied a quantifiable measure of oversaturation by addressing its negative effects in both the temporal and spatial dimensions. The authors characterized the temporal negative effect by the occurrence of a residual queue, referring to the negative effect as a spillover from a downstream intersection to upstream. This study proposed two algorithms to diagnose oversaturated signalized intersection: 1) an algorithm for assessing residual queue length applying shockwave method and 2) an algorithm for detecting spillover by identifying long detector occupancy times in the green phase. The authors used the green time caused by a residual queue or a spillover to explain the oversaturation severity index and confirmed that the proposed algorithm effectively identified oversaturated conditions.

Ban *et al*. [2011] proposed a new shockwave methodology for assessing real-time queue length at signalized intersections and applied it to travel time from mobile traffic sensors. The authors used travel time as the model input, rather than detailed trajectories, to avoid the issue of privacy protection. Their methodology consisted of three major components. The first component consisted of processing raw sample vehicle delays to queuing delays, and the second component assessed the queuing delay patterns using sample queuing delays and queue rear no-delay time (QRNAT). The first component was to calculate maximum or minimum queue lengths and constructing real-time length curves. The main concept of their proposed method was to relate QRNAT to the non-smoothness of queuing delay patterns and queue length changes. Compared to regular methods for traffic modeling using mobile data, the proposed method represented a reverse-thinking process.

2.2 Studies on Static Signal Timing Optimization

Lu *et al*. [2010] developed an intersection traffic signal control model based on reinforcement learning and proposed an optimization method for signal timing of single intersection using a SARSA algorithm. Dong *et al*. [2010] developed a simulated annealing-particle swarm optimization (Sa-PSO) algorithm established from particle swarm optimization (PSO) and metropolis rule. Chin *et al*. [2011] studied a Q-learning approach that could handle a traffic signal timing plan more efficiently by optimizing traffic flows. In another study to derive initial solutions for the particle swarm optimization algorithm, Chen and Xu [2006] used the PSO algorithm to resolve the traffic signal timing optimization problem by installing a local fuzzy-logic controller (FLC) at each junction. To resolve the multi-objective signal timing optimization problem (MOSTOP), Sun *et al*. [2003] applied non-dominated sorting genetic algorithm.

A non-linear optimization model applicable for an individual intersection built up was tested by Li *et al*. [2009] using an ant colony algorithm based on mesh strategy coded in MATLAB. In another study, Mussa and Selekwa [2003] proposed a method of traffic flow optimization during the transition period using a time-of-day (TOD) timing plans approach applied in CORSIM. Li *et al*. [2010] developed a simulation system using VB code based on the traffic characteristics of China for an isolated signal intersection.

2.3 Studies on Adaptive Signal Timing Optimization

Generally, a traffic demand pattern depends on time and location and unpredictable factors, such as crashes, special events, or construction activities, that influence the outcome. Currently, numerous cities are using traffic signal control systems that can continuously optimize signal-timing plans in response to the detected traffic demand of each approach, known as adaptive signal control systems. The first functional deployments appeared in the early 1980s. Since then, propelled by the wide implementation of Intelligent Transportation System (ITS) devices, adaptive signal control systems have become increasingly popular, particularly for vehicle detection and classification. First-generation adaptive signal control systems select the proper signal timing plans in response to the detected traffic demand pattern from the library of pre-stored signal timing plans prefixed off-line based on historical data. A change in the traffic conditions, the time of day, and the day of the week triggers a change in the signal-timing plan. A limitation of this generation of systems is that by the time the system responds, the registered traffic conditions that triggered the response may have become obsolete. Second-generation adaptive signal control systems use an on-line library that implements signal timing plans based on real-time traffic data and predicted values. The signal-timing optimization process can be updated every five minutes. Generally, frequent changes in signal timing plans may lead to transition disturbance. Therefore, in practice, the frequency of changing signal plans cannot be less than ten minutes. The first two generations are also referred to as responsive/adaptive systems. The third generation allows the parameters of the signal plans to change continuously in response to real-time measurement of traffic variables, which allows for acyclic operations.

Some researchers and organizations have dedicated their efforts to the development of adaptive signal control systems. For example, the Roads and Traffic Authority of New South Wales, Australia developed the Sydney Coordinated Adaptive Traffic System (SCATS) in the early 1970s. The optimization algorithm in SCATS uses the concept of degree of saturation (DS), defined as the ratio of fully used green length to effective green length, to determine signal details where cycle length is adjusted in every cycle and splits and offsets are selected from prefixed patterns. Hunt *et al*. [1981] developed the Split Cycle and Offset Optimizing Technique (SCOOT), which optimizes the signal on-line by adjusting the three types of elements (splits, offsets, and cycle lengths) based on the predicted arrival profile and the updated flow information collected by the upstream detectors. The Optimization Policies for Adaptive Control (OPAC) developed by Gartner [1983] is a demand-responsive signal control system that uses the introduced rolling horizon approach to adjust signal parameters in response to the estimated queue length. Koshi [1972], Koshi [1989], and Asano *et al*. [2003] developed a signal control system using the concept of shifting the cumulative diagrams by investigating whether or not a small advance or delay of the next traffic light phase may decrease the aggregate delay to the users. Changes in the offset, split, and cycle were implemented. Lertworawanich [2010] developed a split optimization method for a single isolated intersection by constructing the space-time diagram that was capable of adjusting the split in response to different traffic demand patterns even where the queues extend beyond the detector location. Roshandeh *et al.* [2014] developed a method for optimizing intersection signal timing for an entire urban street network based on the shockwave theory by simultaneously minimizing vehicle and pedestrian delays in each signal cycle over a 24-hour period.

2.4 Limitation of the Existing Methods

Most of the existing models focus on isolated intersections or individual corridors. The lack of a rigorous methodology for addressing the network impacts of intersection traffic signal timing optimization makes it likely they will produce ineffective signal timing plans for efficient utilization of the existing capacity of intersections, corridors, or urban street networks.

CHAPTER 3 PROPOSED METHODOLOGY

This chapter discusses the proposed methodology for system-wide signal timing optimization considering vehicle delays at individual intersection approaches. It begins with the proposed new dynamic split optimization model for vehicle delay computation using the shockwave theory and then discusses the computational analysis process for model execution. It further describes interfacing and integrating the proposed model into the TRANSIMS toolbox for large-scale urban network applications.

3.1 Basic Concepts of Traffic Movements at Signalized Intersections

3.1.1 Merits of the Proposed Model

As seen in the shockwave model introduced by Roshandeh *et al*. [2014], both undersaturated and oversaturated traffic movements at signalized intersections are considered. This method was further refined in this thesis to characterize traffic movements more accurately from the following four aspects:

First, the shockwave model by Roshandeh *et al*. [2014] considers a fixed time point between the before-hump and after-hump transition speeds under the oversaturated traffic condition. The current model allows a flexible point for the transition speeds, depending on the vehicles entering the intersection approaches during the signal cycle and the interactions of the entering vehicles with the corresponding green intervals. Second, the model by Roshandeh *et al*. [2014] assumes that the oversaturated and undersaturated cases share the same maximum queue length. This

rarely the case in real world circumstances. This thesis assumes that the queue length accumulates as time goes on, as is the case with vehicular delays. Third, the shockwave model by Roshandeh *et al*. [2014] considers that the last vehicles entering the intersection during the green intervals dissipate precisely at the end of the yellow interval. In fact, this represents the worst case and that the last vehicle may clear before the end of the yellow interval. This thesis allows the clearance of the last entering vehicle to occur before or at the end of the yellow interval. Fourth, all the wave speeds calculated in the shockwave model by Roshandeh *et al*. [2014] are based on the critical lane volumes. Conversely, this thesis uses the total vehicle volumes entering from all intersection approaches for the computation. The improvements this thesis makes over the earlier model by Roshandeh *et al*. [2014] ensure that the proposed model provides a more accurate estimation of vehicle delays per cycle per intersection. The details of delay calculation are presented in the next section.

3.1.2 Vehicle Delay Calculation Using the Proposed Model

Figure 1 depicts how a vehicular queue forms and discharges due to a red signal when the traffic flow is undersaturated. When the traffic signal turns red, the vehicles stop and form a queue at wave speed v_1 until point A (illustrated by line 1). Then, the queueing back speed slows down because the traffic demand upstream is not as high as before. Therefore, the queue forms at slower wave speed v_2 (illustrated by line 2). When the signal turns green, the queue discharges at wave speed v_3 (illustrated by line 3). The queue discharges completely at point B (the intersection of the line 2 and line 3). After point B, the newly arrived vehicles join the discharge flow without any stopping. Moreover, the forward shockwave speed is v_4 (illustrated by line 4). After point C, the traffic flow of this approach comes back to the original status until the next red signal.

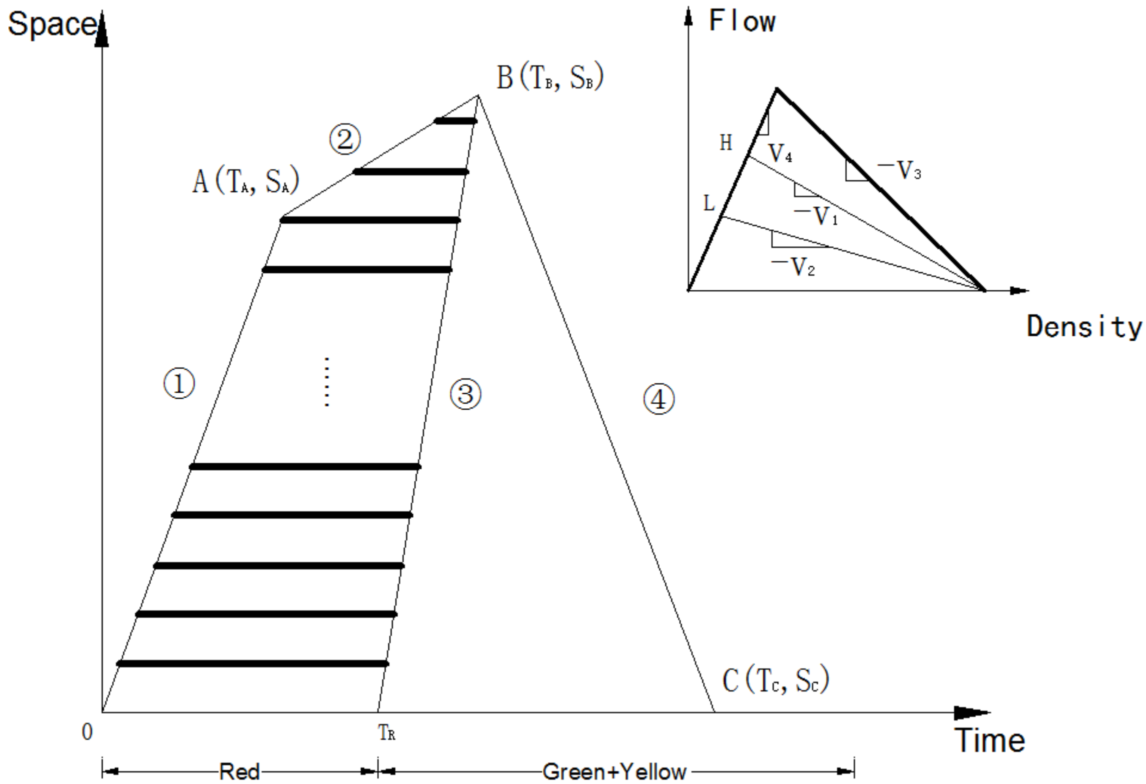

Figure 1 Time-Space Diagram of Vehicles Traversing Through Intersections under Undersaturated Traffic Conditions

The vehicles arrive at the intersection at different arrival rates, which is simplified by considering two constant rates. The relatively higher arrival rate is assumed to be oriented from the critical movement vehicles released by the upstream intersection. Also, the relatively lower arrival rate is caused by non-critical movement vehicles. In undersaturated traffic conditions, a triangular flow-density curve is assumed, as illustrated by the flow-density curve at the corner of Figure 1. The state of higher flow rate is denoted as point H, and the lower flow rate is denoted as point L in the figure.

The delay of each vehicle is measured as the waiting time when its velocity is zero. As illustrated by the vertical bold lines between line 3 and line 1, and line 2. From the left-hand side to the right-hand side, the length of each line is the delay of the corresponding vehicle. Obviously, the delay depends on the vehicle's location in the queue. According to the geometric relationship illustrated above, one can safely derive the following relationship:

$$n = \left\lfloor \frac{S_B}{Jam\ Spacing} \right\rfloor = \lfloor S_B * K_j \rfloor \approx S_B * K_j \qquad (3\text{-}1)$$

where, n is the number of vehicles waiting in this queue during one cycle; and K_j is the jam density.

Also, denote the location of the i_{th} vehicle as $L(i)$, then:

$$L(i) = L(i-1) + Jam\ spacing = L(i-1) + \frac{1}{K_j} \qquad (3\text{-}2)$$

$$L(1) = 0; \qquad (3\text{-}3)$$

The general expression for $L(i)$ according to (3-2) and (3-3) is:

$$L(i) = \frac{i-1}{K_j} \qquad (3\text{-}4a)$$

$$dL = \frac{di}{K_j} \qquad (3\text{-}4b)$$

In addition, Equation (3-4a) is the total differential expression of Equation (3-4b).

Equation of the three lines:

Line 1 $(0 \le S \le S_A)$: $T_1(S) = \frac{S}{v_1}$

Line 2$(S_A \le S \le S_B)$: $T_2(S) = \frac{S}{v_2} + T_A - \frac{S_B}{v_2}$

Line 3$(0 \le S \le S_B)$: $T_3(S) = \frac{S}{v_3} + T_R$

Therefore, for $L(i) \le S_A$:

$$delay(i) = T_3(i) - T_1(i) = \left(\frac{1}{v_3} - \frac{1}{v_1}\right) * L(i) + T_R \quad (3\text{-}5)$$

For $S_A \le L(i) \le S_B$:

$$delay(i) = T_3(i) - T_2(i) = \left(\frac{1}{v_3} - \frac{1}{v_2}\right) * L(i) + T_R - T_A + \frac{S_B}{v_2} \quad (3\text{-}6)$$

In sum:

$$delay[L(i)] = \begin{cases} \left(\frac{1}{v_3} - \frac{1}{v_1}\right) * L(i) + T_R \quad if\ L(i) \le S_A \\ \left(\frac{1}{v_3} - \frac{1}{v_2}\right) * L(i) + T_R - T_A + \frac{S_B}{v_2} \quad if\ S_A \le L(i) \le S_B \end{cases} \quad (3\text{-}7)$$

In short:

$$delay[L(i)] = \begin{cases} D_1(L) \quad if\ L(i) \le S_A \\ D_2(L) \quad if\ S_A \le L(i) \le S_B \end{cases} \quad (3\text{-}8)$$

The next step is to compute the total delay of this queue. In the following step, approximation is made by replacing the summation with the integration in order to simplify the computation.

$$\sum_{i=1}^{n} delay(i) \approx \int_1^n delay(i) * di \xRightarrow{plug\ in\ Eq(4-2)} \int_0^{S_B} K_j * delay(L) * dL$$

$$= K_j * \left[\int_0^{S_A} D_1(L) * dL + \int_{S_A}^{S_B} D_2(L) * dL\right] = K_j\Omega \quad (3\text{-}9)$$

where, Ω is the area of the quadrangle bounded by the line 1, 2, 3 and the time axle.

$$\Omega = \frac{(T_R+T_B)*S_B - S_A T_A - (T_A+T_B)(S_B-S_A)}{2} \quad (3\text{-}10)$$

Further,

$$Average\ delay = \frac{Total\ delay}{n} = \frac{K_j\Omega}{K_j * S_B} = \frac{\Omega}{S_B} = \frac{(T_R+T_B)*S_B - S_A T_A - (T_A+T_B)(S_B-S_A)}{2S_B}$$

(3-11)

$$Total\ delay = K_j\Omega = \frac{(T_R+T_B)*S_B-S_AT_A-(T_A+T_B)(S_B-S_A)}{2}K_j \tag{3-12}$$

For oversaturated traffic conditions, as illustrated below (Figure 2), the moment when the signal turns red, the queue forms in three stages, as illustrated by lines 5, 1, and 2. In stage 1 (illustrated by line 5), the queue is formed at wave speed v_5 by the vehicles that queued in the previous cycle. These vehicles have to stop again because the green time allocated to this approach is not adequate for all the vehicles to pass the intersection in the previous cycle. In stage 2 (illustrated by line 1), the queue forms at wave speed v_1 by the newly arrived vehicles at a high arrival rate, which is similar to line 1 in the undersaturated case. In stage 3 (illustrated by line 2), the queue forms at speed v_2 by the newly arrived vehicles in a lower volume status, which is similar to line 2 in the undersaturated case. When the signal turns green, the queue discharges at wave speed v_3 (illustrated by line 3). After point B, which is the intersection of line 2 and line 3, the newly arrived vehicles pass through without stopping (illustrated by line 4). The forward wave speed is v_4.

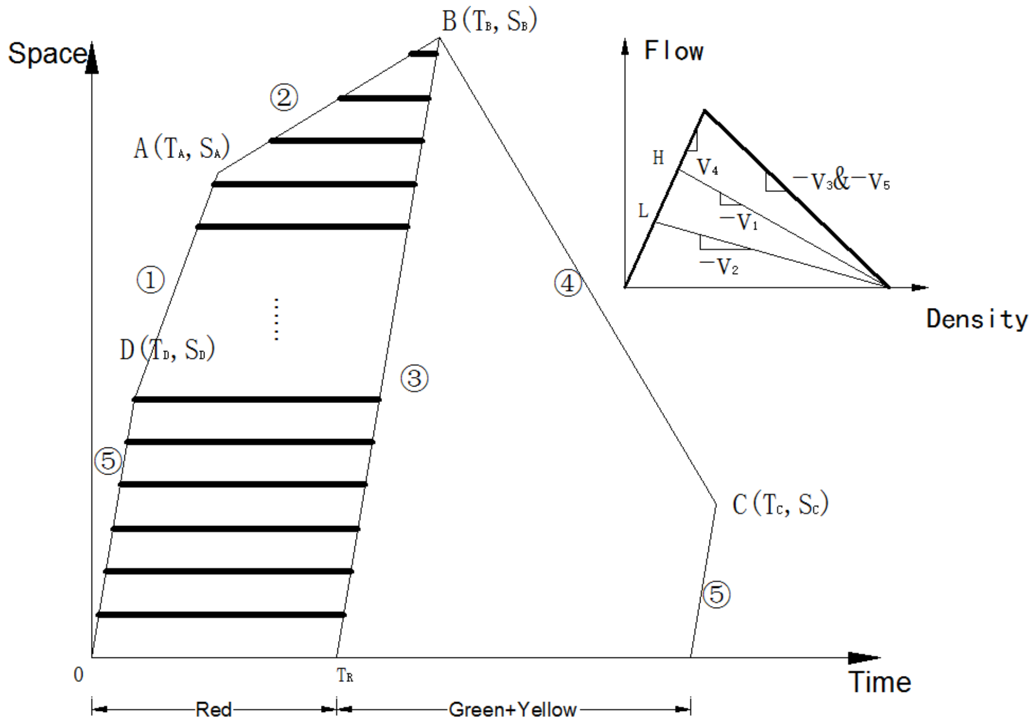

Figure 2 Time-Space Diagram of Vehicles Traversing through Intersections under Oversaturated Traffic Conditions

By the logic, the average vehicular delay is computed in the following steps:

$$Average\ delay = \frac{\Omega}{S_B}$$

$$= \frac{(T_R+T_B)*S_B-S_DT_D-(T_A+T_D)(S_A-S_D)-(T_A+T_B)(S_B-S_A)}{2S_B} \quad (3\text{-}13)$$

$$Total\ delay = \frac{(T_R+T_B)S_B-S_DT_D-(T_A+T_D)(S_A-S_D)-(T_A+T_B)(S_B-S_A)}{2}k_j$$

(3-14)

The formulae of average vehicular delays derived in the above differ from those of the previous studies, particularly those developed by Roshandeh *et al*. [2014].

The assumed triangular flow-density relationship is shown at the right-top corner of Figures 1 and 2, where H(k_h,q_h) represents the high volume status, L(k_l,q_l) represents the low volume status, and (k_{max}, q_{max}) represents the capacity status. Table 1 summarizes the wave speed calculations.

Table 1 Summary of Wave Speed Calculations

Wave Speed	Definition	Calculation using Field Measurements	
		Undersaturation	Oversaturation
v_1	$\left\|\frac{q_h}{k_j - k_h}\right\|$	$\frac{S_A}{T_A}$	$\frac{S_A - S_D}{T_A - T_D}$
v_2	$\left\|\frac{q_l}{k_j - k_l}\right\|$	$\frac{S_B - S_A}{T_B - T_A}$	$\frac{S_B - S_A}{T_B - T_A}$
v_3	$\left\|\frac{q_{max}}{k_j - k_{max}}\right\|$	$\frac{S_B}{T_B - T_R}$	$\frac{S_D}{T_D} = \frac{S_B}{T_B - T_R}$
v_4	$\left\|\frac{q_{max}}{k_{max}}\right\|$	$\frac{S_B}{T_C - T_B}$	$\frac{S_B - S_C}{T_C - T_B}$
v_5	$\left\|\frac{q_{max}}{k_j - k_{max}}\right\|$	N/A	$\frac{S_D}{T_D} = \frac{S_B}{T_B - T_R}$

3.2 Further Explanation of Traffic Movements at Signalized Intersections

The queuing back pattern discussion above occurs for every movement of each phase in a signal-timing plan if signals of two successive intersections are not well coordinate. A well-coordinated intersection releases vehicles in the traffic stream with a higher flow rate directly at the time they arrive at the intersection; and at the same time a queue forms during the red signal from vehicles in the traffic stream with a lower flow rate. Therefore, to some extent, a space-time diagram can be used to illustrate the worst case of a queueing back pattern caused by a signal. Consequently, the computed delay may be the maximum delay with the signal timing details and flow details given.

In terms of the data needed for the delay computation, vehicle running speed (v_4) can be collected by sensors or detectors. The capacity of each lane can be computed using information in the 2010 Highway Capacity Manual (HCM) [TRB, 2010] or other acceptable specifications. Field observations could be used to derive the jam density, which this thesis assumes as 150 veh/km/lane. The critical inputs of the model are the two flow rates. Generally, vehicle detectors collect only one value of the average flow rate, which indicates the general traffic flow demand using this link or the particular lane. Therefore, in order to achieve these two flow rates based on the given information, two factors (f_1 and f_2) are created to estimate q_h and q_l. The relationship established among these factors are given by (3-15) and (3-16).

$$q_h = f_1 * (q_{max} + q) \tag{3-15}$$

$$q_l = f_2 * q \tag{3-16}$$

Both f_1 and f_2 range from 0 to 1. Further regression of the data collected by field measurements will produce both factors. Formula (3-15) ensures that the higher flow rate is between the observed average flow rate and the capacity. Similarly, Formula (3-16) ensures that the lower flow rate is within zero and the observed average flow rate. However, applying these two formulae requires that the observed average flow rate is greater than zero and less than capacity. If the approach volume is zero during a certain period, these two factors also should be zero. When the observed average flow rate during the period exceeds capacity, the capacity used in this model should be updated accordingly.

Another critical factor is the time duration of both flow rates in a signal cycle. In other words, the proportion of time that the intersection experiences a higher flow rate and the proportion of time that is spent at a lower flow rate if green time is allocated to this approach all the time. Figure 3 illustrates the geometric relationship between this time proportion factor f and other factors.

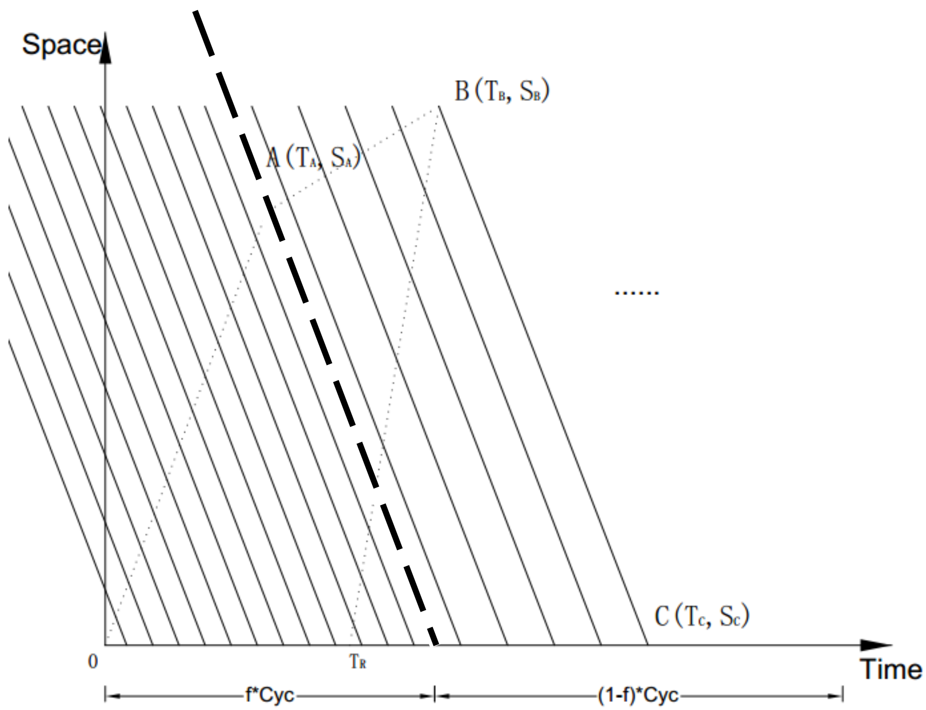

Figure 3 Fraction of Cycle Used by Two Arrival Volumes

In Figure 3, the dash line is the boundary separating the traffic stream between the higher flow rate and the lower flow rate conditions. Correspondingly, this line crosses transition point A. The dotted line is the end of the queue formed during red time discussed in the previous section. The series of solid parallel lines on the left side of the dash line indicates the trajectories of the vehicles in the traffic stream with a higher flow rate. Likewise, the parallel lines on the right side of the dash line are the trajectories of the vehicles in the traffic stream with a lower flow rate. Based on the conservation law of traffic volume, Equations (3-17) and (3-18) can be derived as follows:

$$q_h * f * Cyc + q_l * (1 - f) * Cyc = q * Cyc \quad (3\text{-}17)$$

$$f = \frac{q - q_l}{q_h - q_l} \quad (3\text{-}18)$$

Furthermore, according to the geometric relationship represented by Figure 3.3, the coordinate of point A can be computed as follows:

$$T_A = \frac{v_4 * f * Cyc}{v_1 + v_4} \quad (3\text{-}21)$$

$$S_A = v_1 T_A = \frac{v_1 v_4 * f * Cyc}{v_1 + v_4} \quad (3\text{-}22)$$

After the coordinates of point A has been found, the locations of point B and point C can be computed based on the corresponding shockwave speeds. If the time coordinate of point C is not greater than the cycle length, the traffic condition is undersaturated. Consequently, the vehicle delays for the undersaturated case can be computed.

With regard to the oversaturated case, the computation of vehicle delays requires information on the residual queue length left from the previous cycle. Therefore, the space-time diagram should be drawn from the beginning of the cycle for which the residual queue length is zero. As illustrated by Figure 2, the queue length of the oversaturation case accumulates over time. The residual queue length from the last cycle can be found at point C of the previous cycle and represents point D of the current cycle. Naturally, the total vehicle delays in the current cycle can be computed by summing up the delays in the previous cycle and the area of the shaded parallelogram. According to the geometric relationship, the coordinate of point C for the first cycle in the oversaturated case can be computed using Equations (3-23) and (3-24).

$$T_{D2} = T_{C1} = \frac{S_{B1}}{v_{41}} + T_{B1} \quad (3\text{-}23)$$

$$S_{D2} = S_{C1} = \frac{(T_{C1} - Cyc)v_{31}v_{41}}{v_{31} + v_{41}} \quad (3\text{-}24)$$

Unlike the undersaturated case, the maximum queue length and the total vehicle delays depend on the number of cycles considered. In practice, considering two to five cycles is desirable. The average value of delays in multiple cycles can be utilized as the indicator of delay measurement.

3.3 Proposed Method for Signal Timing Optimization

The current model also refines the optimization of green splits according to the vehicle volumes expected to enter the intersections approaches in the near future signal cycles to achieve minimum delays per vehicle per cycle averaged over multiple consecutive cycles. In addition, the optimization is conducted using fewer constraints to be more consistent with real world situations.

The optimization formulation is as follows:

$$Minimize\ \sum_{i \in I} \sum_{m \in M_i} Delay_m \quad (3\text{-}25)$$

Where:

$$Delay_{Ui} = \frac{(T_R + T_B) * S_B - S_A T_A - (T_A + T_B)(S_B - S_A)}{2} k_j$$

$$Delay_{Oi} = \frac{(T_R + T_B)S_B - S_D T_D - (T_A + T_D)(S_A - S_D) - (T_A + T_B)(S_B - S_A)}{2} k_j$$

(3-26)

Subject to:

$$T_{Gi} \geq \frac{W_i}{v_{ped}} \quad (3\text{-}27)$$

$$T_{Gi} + T_{Ri} + T_{Yi} = Cyc \quad (3\text{-}28)$$

where, i and the phase ID and m is the movement traffic ID disallowed to move in phase i . M_i is the set containing the IDs of all movements disallowed within phase i. I is the set containing all the phase IDs of the signal timing plan of a given intersection. T_{Gi}, T_{Ri}, T_{Yi} is the green time, red time, and yellow time for phase i. Cyc is the cycle length of this intersection. W_i is the width of the intersection corresponding to phase i. v_{ped} is the walking speed of pedestrians.

In this signal timing optimization model, objective function (3-25) computes the sum of the vehicle delays of all the movements in all phases within a signal-timing plan at a given condition. Formula (3-26) indicates the quantity of total vehicle delays depending on whether it is possible to discharge the queue formed in the red signal during the non-red signal fully. Constraint (3-27) sets the lower bound of the green time equivalent to the minimum pedestrian crossing time. If the signal time plan contains one or two protected left-turn phases, constraint (3-27) can be relaxed for those protected left-turn phases. Constraint (3-18) indicates that the cycle length remains unchanged. Unlike using average vehicle delays on a critical lane for each phase in the objective function adopted by Roshandeh *et al.* [2014], the total vehicle delays aggregated for all vehicle movements in all phases within a signal cycle is used as the mobility performance measure.

3.4 Iterative Solution Process for the Proposed Model

The above optimization modeling of green splits is applied for the AM and PM peak periods of a typical day where each peak period is further split into fixed time intervals and, within each time interval, multiple signal cycles are involved. The optimal green splits are determined using the following iterative computation process:

- Determine the predicted vehicle volumes entering individual intersection approaches in each signal cycle for the given time interval.
- Calculate the total vehicle delays using the predicted vehicle volumes entering individual intersection approaches for the given signal timing design for the time interval.
- Adjust the green splits of the signal timing design to achieve the lowest delays.
- Apply the new green splits of signal timing designs for intersections within an urban street network to the subsequent signal cycle, which is expected to trigger traffic redistribution in the urban street network, leading to changes in traffic volumes entering into intersection approaches in the subsequent signal cycle.
- Repeat Steps 2-4 until the time sequence of the entire AM or PM peak period is complete.

Figure 4 depicts the iterative process for solving the optimization problem at a certain intersection, which is the essence of Steps 2 and 3. After optimizing all the intersections in the network, the updated signal-timing plan should be exported as the input for a traffic simulation in order to examine the traffic redistribution due to the change of signal timing plan. As a part of the output data reported by the simulation system, the simulated traffic data for the next time step will serve as the input of the next iteration. Figure 5 illustrates the alternated computational process from the given green splits in a signal-timing plan and the vehicle volume to optimize green splits with minimized delays and a new vehicle volume in response to the new green splits with signal cycles progressing to the end of the signal optimization period.

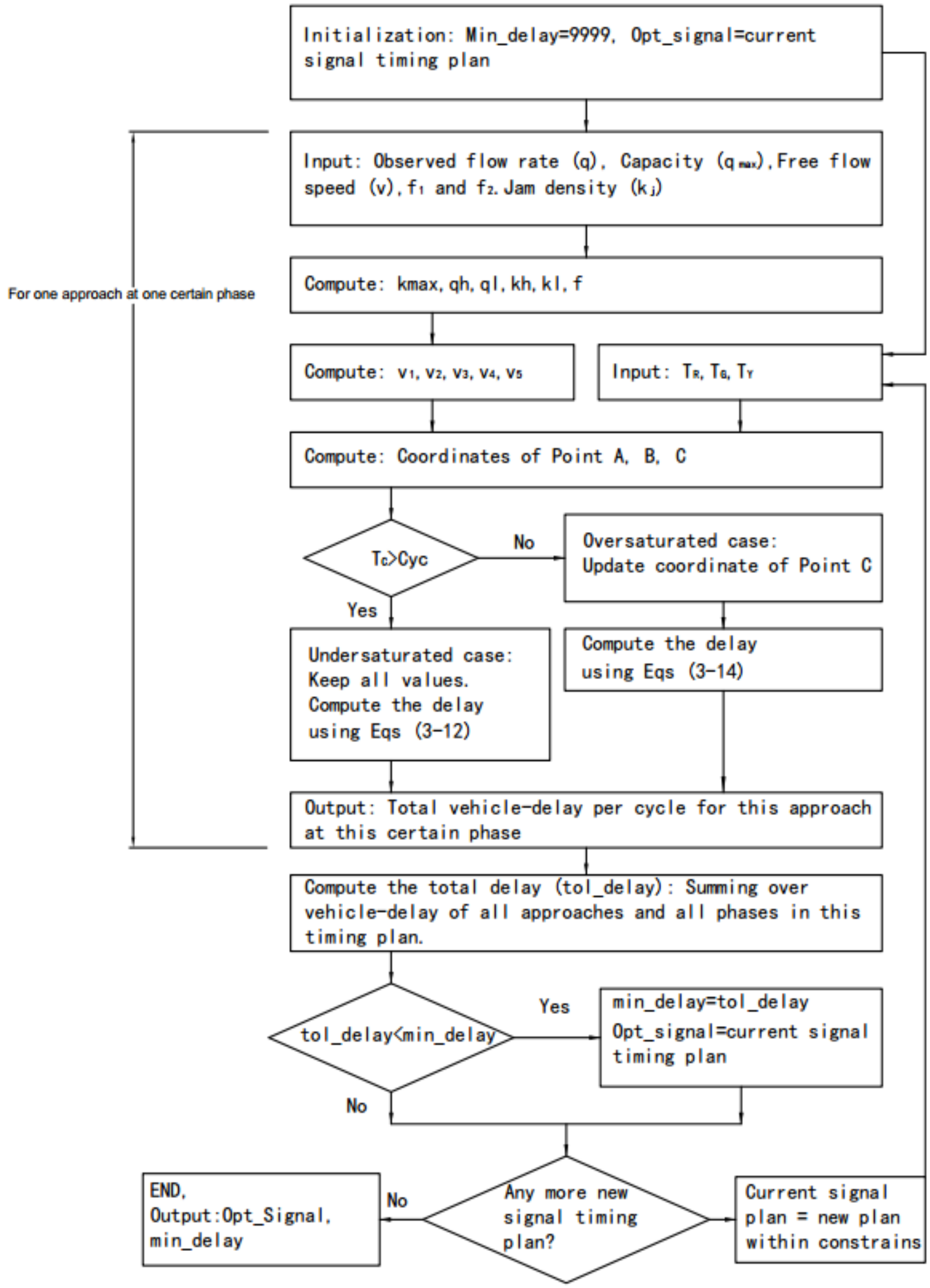

Figure 4 Iterative Solution Process for the Proposed Model

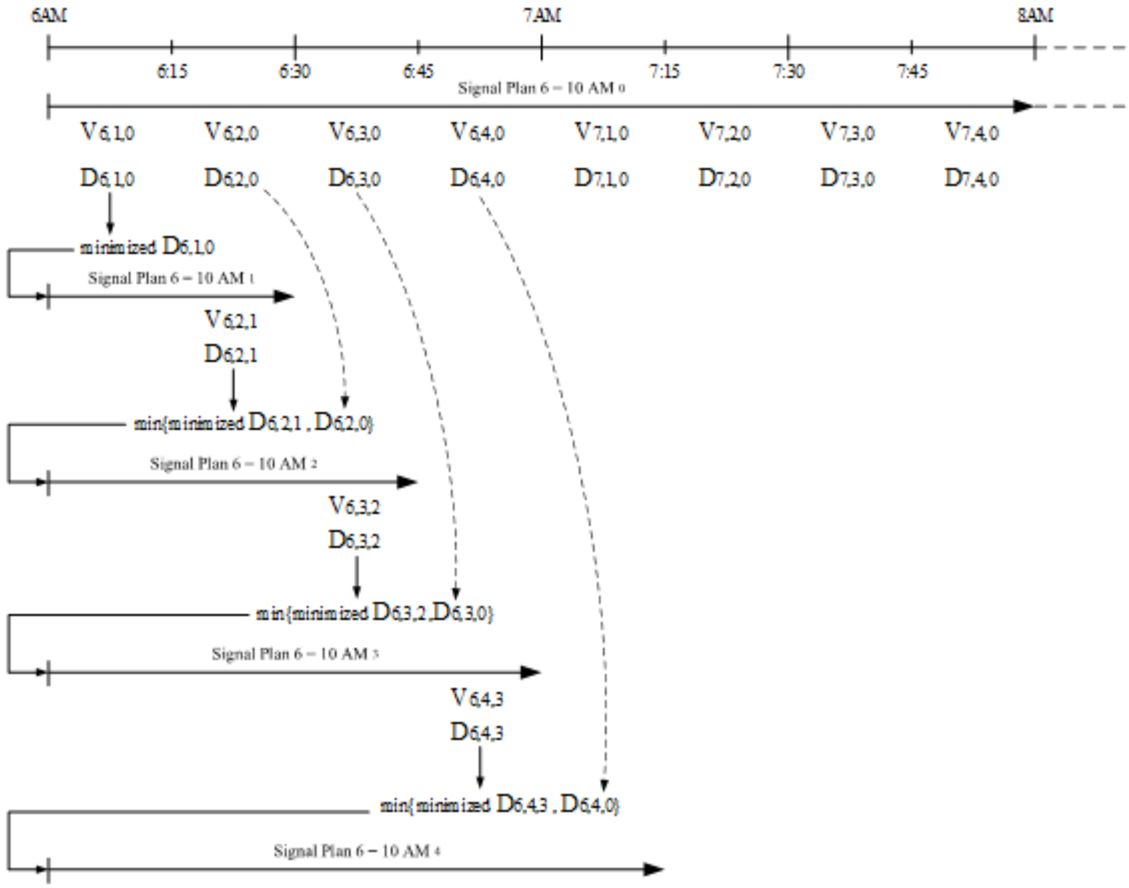

Figure 5 Computation Process of Green Split Optimization

3.5 Integrating the Proposed Model and Computing Process into TRANSIMS

TRansportation ANalysis and SIMulation System (TRANSIMS) is an integrated system of travel forecasting models designed to give transportation planners accurate and complete information on traffic impacts, congestion, and pollution [Li *et al*., 2012]. It is one of the very few analytical tools capable of conducting large-scale, high fidelity simulation-based traffic assignments using the regional daily origin-destination (O-D) travel demand and signal timing plans for intersections within an urban street network. It uses supercomputing facilities to obtain the predicted traffic volumes for individual intersection approaches with the traffic assignment results updated on a second-by-second basis. The platform virtually can handle a regional multimodal transportation network of any size that may contain a large number of signalized intersections. For this reason, it is used for the methodology application. The TRANSIMS model calibrated for Chicago by the Illinois Institute of Technology (IIT) in conjunction with the Argonne National Laboratory is the largest and most complex TRANSIMS-based model currently available in the United States as the next generation tool for transportation planning, traffic operations management, and evacuation planning/emergency management analysis. It was successfully calibrated and validated using fine-grained field traffic counts and is applied for a number of real world planning and operations scenarios. For this reason, the Chicago TRANSIM model was adopted in this thesis and augmented to demonstrate the proposed model.

First, the traffic signal timing plans for intersections in the study area were collected. Next, the iterative solution process as described in Section 3.3 was coded using Python programming language to obtain new signals timing plan. Without changing the existing cycle lengths and signal coordination, the green splits of all the signal phases of the existing signal timing plans for the AM peak, PM peak, and remaining periods of the day were adjusted. This iterative process was repeated until all the possible green splits were examined to finally achieve minimized vehicle delays per vehicle per cycle as the objection functions of Equations 3-13 and 3-14. The new signal-timing plan then was used as an input set of data in the TRANSIMS platform to iteratively estimate traffic volumes on each intersection approach by the time interval employed for vehicle volume aggregation. This iterative process stopped when the aggregated traffic volumes in the iterative computation process became stable. Finally, the differences in the vehicle travel times, delays per cycle, number of vehicles stopped in queues, and average speeds before and after green split optimization were used as measures to assess the effectiveness of the proposed model.

CHAPTER 4 METHODOLOGY APPLICATION

This chapter focuses on applying the proposed model along with the iterative computation process integrated into the TRANSIMS platform to obtain the optimal green splits for all phases of a signal timing design for a specific intersection without changing the cycle length of the original signal timing design and coordination for multiple intersections. The model output results before and after optimizing the green splits of intersection signal timing plans are used for model assessment.

4.1 The Study Area

With respect to the Chicago metropolitan area, the central business district (CBD) network contains a large number of signalized intersections, making it an ideal study area to apply the proposed model. Further, significant delays at intersections in the Chicago CBD area occur within its core area of the Chicago Loop bounded by Wacker Drive along the Chicago River, Roosevelt Road, and Lakeshore Drive (Figure 6). Therefore, the Chicago Loop was selected as the study area, which contains 143 major signalized intersections.

4.2 Green Split Optimization Time Period and Interval Considerations

For the intersections located in the Chicago Loop street network, the most severe delays occur during the AM and PM peak periods. As such, the model application focused on signal timing adjustments through green split optimization for the 143 intersections within the Loop area for the AM and PM periods. In considering the long duration of each peak period, a four-hour duration was considered to ensure the peak and the adjacent to peak time slots were all inclusive in the analysis. In order to capture the traffic dynamics, four 15-minute time intervals were considered for each hour. Within each 15-minute time interval, multiple signal cycles were involved.

Without altering the cycle length of a specific intersection and signal coordination of multiple intersections, the green splits of each intersection were adjusted according to the vehicle volumes traversing the intersection to ensure achieving the lowest extent of vehicle delays per vehicle per cycle averaged over consecutive cycles.

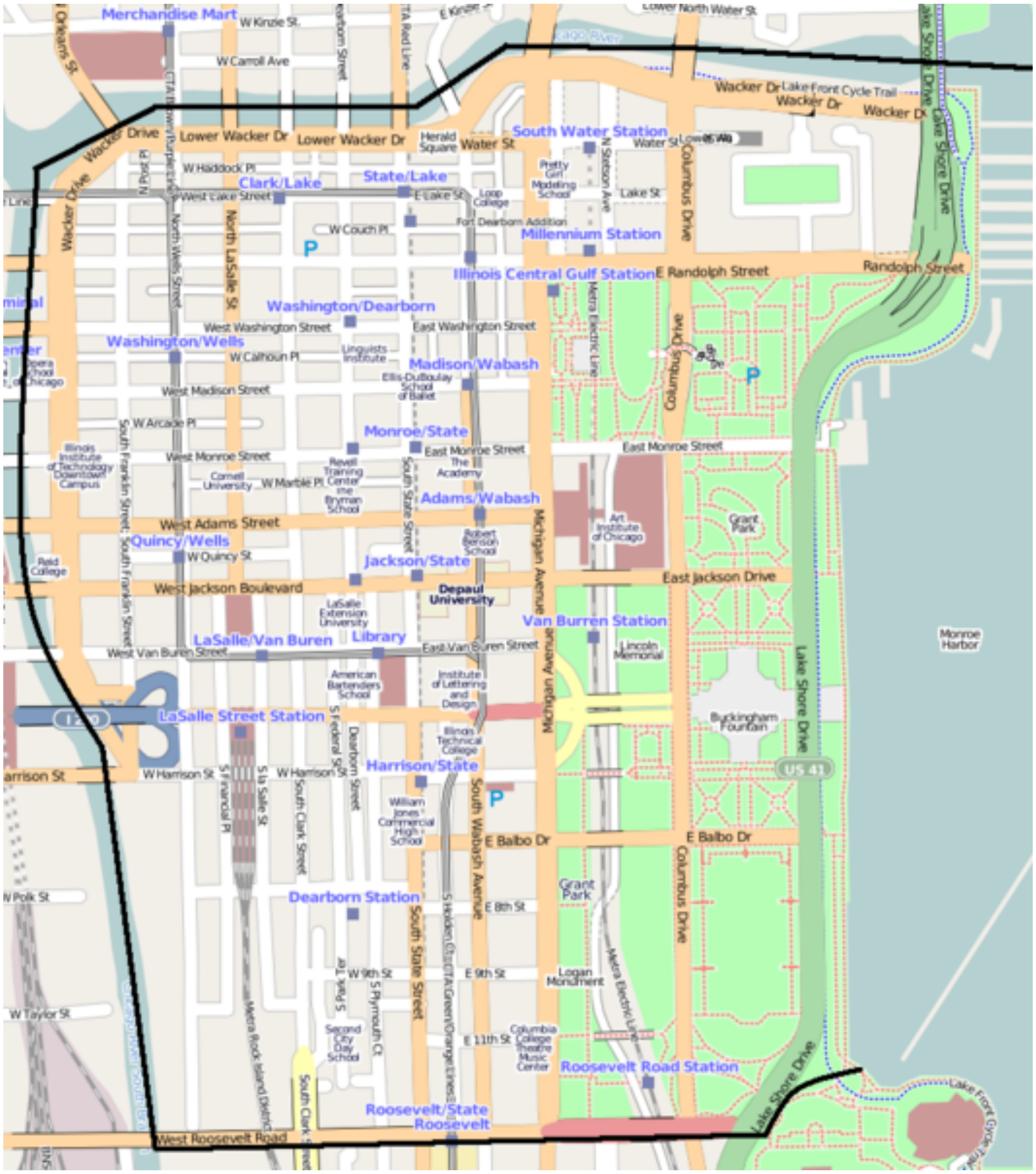

Figure 6 Study Area for Applying the Proposed Dynamic Split Optimization Model
Source: Chicago City Map Loop Area

4.3 Data Collection and Processing

Data details of travel demand, geometric designs, and traffic controls including signal-timing plans associated with the highway network in the Chicago metropolitan area were assembled for applying the proposed model for green split optimization integrated into the Chicago TRANSIMS platform. The primary data categories are discussed below.

4.3.1 Travel Demand

Travel demand data were obtained from the Chicago Metropolitan Agency for Planning (CMAP), which contained information on 28.5 million trips for a typical day in the current year classified by trip purpose and hour of the day that were generated from 1,961 traffic analysis zones (TAZs) in the entire Chicago metropolitan area. The Chicago model uses two types of traffic demand inputs for regional traffic assignments:

1) Inter-zonal, intra-zonal, and external trips and diurnal distributions by hour of the day for a 24-hour period, which were separately established for ten different trip purposes, which mainly included home-based work (HBW), home-based other (HBO), and non-home-based (NHB) auto and transit trips, airport trips, and external trips.

2) Departure time of each trip during the 24-hour period.

4.3.2 Intersection Signal Timing Plans

The intersection traffic signal timing dial during each day may be split into multiple dials to accommodate AM peak, PM peak, and all other time period conditions.

- Monday - Friday: 6AM-10AM
- Monday - Friday: 3PM-7PM
- All other periods

Therefore, as a part of intersection traffic signal timing updating, new Python scripts were added to accommodate the option of three dials per day as follows:

Time_Period_Breaks 0:00, 6:00, 10:00, 15:00, 19:00

As shown in Table 2, five time slots were created for a given 24-hour period within the three timing dials.

Table 2 Intersection Signal Timing Dial Conversions

TRANSIMS Dial	TRANSIMS Start Time	Real Time Dial
1	0:00	Dial 1
2	6:00	Dial 2
3	10:00	Dial 1
4	15:00	Dial 3
5	19:00	Dial 1

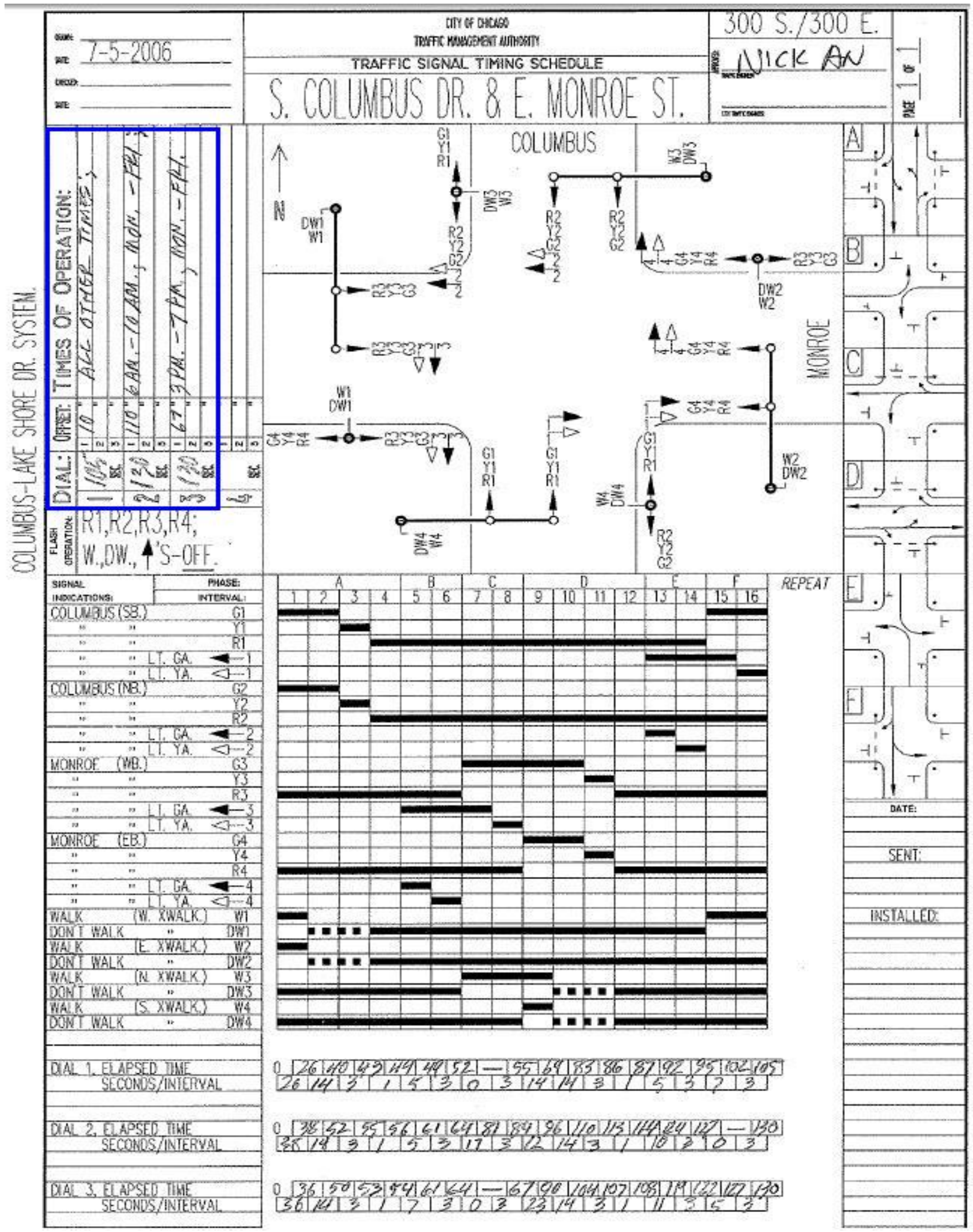

Figure 7 Sample Traffic Signal Timing Sheet for a city of Chicago-maintained Intersection. Source: Chicago Department of Transportation

4.3.3 Travel Time, Speed, Traffic Volume, and Intersection-Related Vehicle Delays

The TRANSIMS model produced the average travel time, travel speed, and traffic volume by the hour of the day for each highway segment or intersection approach before and after optimization of green splits in the signal timing design for each intersection and constituted the data to be analyzed. As the Chicago Loop area was selected as the study area for the model application, the trips with O-D paths falling within the Loop area were relevant. Hence, the average travel time, speed, and traffic volume, as well as the vehicle delays at intersections in the Loop area before and after optimizing the green splits were computed and then used to assess the effectiveness of signal optimization.

With respect to the calculation of the reduction in vehicle delays per vehicle per signal cycle in green split optimization, it was assumed that the queued vehicles in a specific signal cycle could be potentially dissipated within two consecutive signal cycles. As such, the reduction in vehicle delays after green split optimization was computed as the average over the reductions in vehicle delays in two consecutive signal cycles.

4.4 Preliminary Data Analysis before Model Application

Prior to executing the proposed model within the Chicago TRANSISM platform, the total traffic demand using the Chicago Loop street network in the AM peak period from 6:00AM to 10:00AM was evaluated. As shown in Figure 8, a steady increasing trend in traffic demand aggregated in 15-minute time intervals was observed from 6:00AM to 9:00AM and began to drop from 9:00AM to 10:00AM. This seems to suggest that the AM peak period is from 8:00AM to 10:00AM. For the four-hour time duration, the minimum, maximum, and average number of vehicles using the Loop street network were approximately 20,000, 55,000, and 39,500 vehicles per 15-minute time period, respectively.

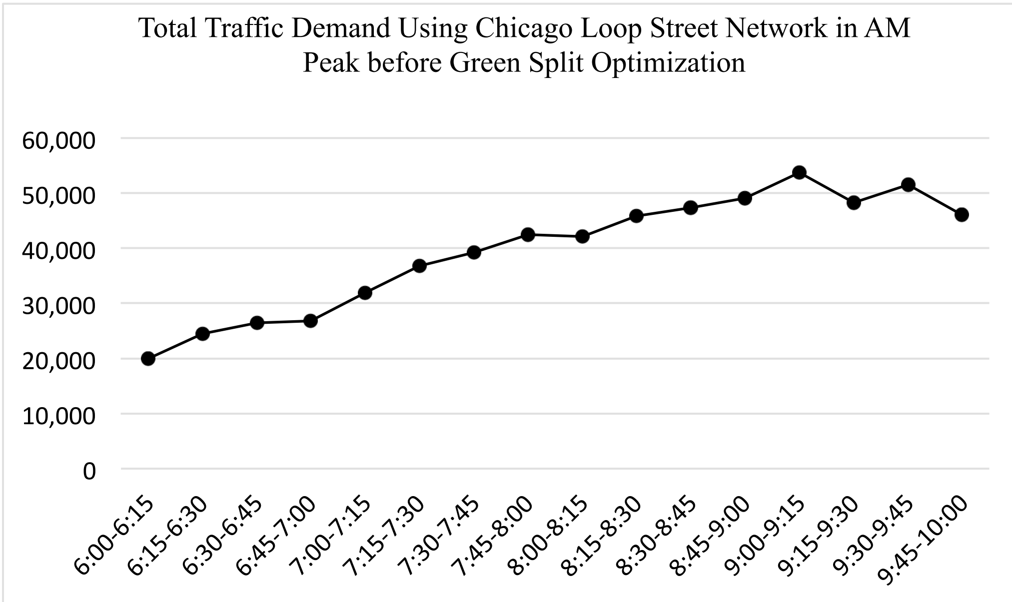

Figure 8 Total Traffic Demand Using Chicago Loop Street Network in AM Peak before Green Split Optimization

The traffic demand for the Chicago Loop street network in the afternoon peak is presented in Figure 9. The traffic demand slightly fluctuates around 50,000 vehicles from 3 PM to 6 PM. After 6 PM, the traffic demand suddenly dropped to 39,000 at 5:30 PM and remained steady until the end of the afternoon peak. Similarly, it is reasonable to believe the afternoon peak period spanned three hours, ranging from 3:00 to 6:00 PM. Generally, the maximum quarterly volume was 56,615 vehicles and was experienced in the fourth quarter of 4 PM. The minimum quarterly volume was 37,520 vehicles in the third quarter of 6 PM, and the average quarterly volume was 49,869 vehicles.

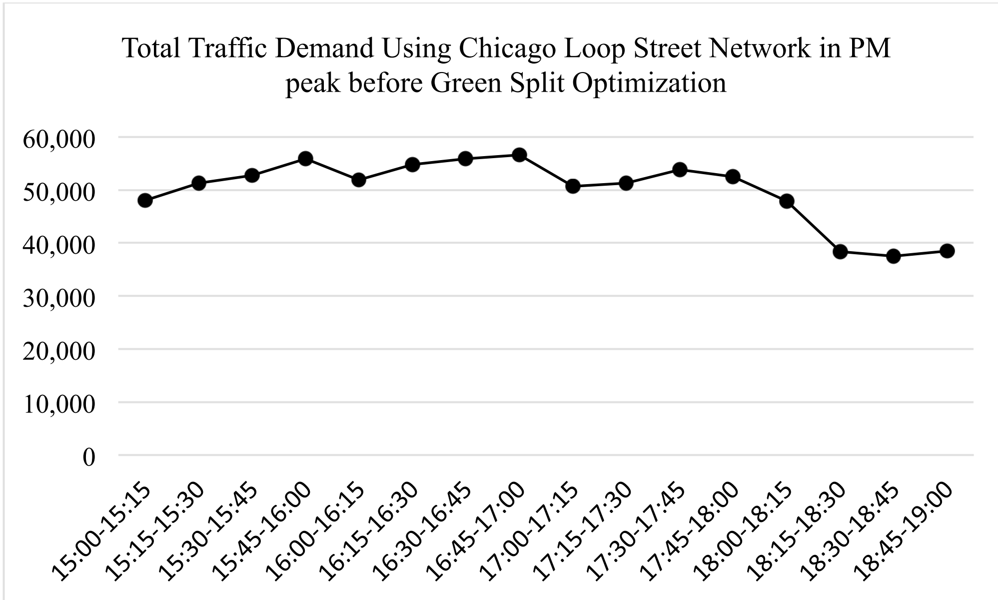

Figure 9 Total Traffic Demand Using Chicago Loop Street Network in PM Peak before Green Split Optimization

Compared with the flow pattern during the AM peak, the flow pattern in the PM peak exhibited greater fluctuation, particularly from 3:00 PM to 6:00 PM. The reason for this complex flow pattern is that some employers have adopted the so-called staggered rush-hour policy. Therefore, the increasing trends that appear at 3:00 PM, 4:00 PM, and 5:00 PM were caused by the corresponding end-of-office hours. The slightly increasing trend in the last 30 minutes may be attributable to travelers driving for recreational purposes after work. Table 3 summarizes the total vehicle hours of travel (VHT) in the Chicago Loop street network before the green split optimization. The total VHT over the eight-hour peak period was 2,017.09 vehicle hours.

Table 3 Total VHT in Peak Hours before Signal Timing Optimization

AM Peak	6:00-7:00	7:00-8:00	8:00-9:00	9:00-10:00
VHT (Veh-hr)	40.62	157.16	128.95	239.04
PM Peak	15:00-16:00	16:00-17.00	17:00-18:00	18:00-19:00
VHT (Veh-hr)	322.37	567.20	567.20	281.95

4.5 Model Application Results

4.5.1 Reductions in Peak Period Vehicle Delays

Tables 4 and 5 summarize the reductions in vehicle delays for 15-minute time intervals of one-hour duration in the AM peak and PM peak periods. The tables indicate that the delay reductions increased gradually from the beginning of the AM peak period and became stable at approximately 39% by the end of the AM peak period. The trend of delay reductions in the PM peak period increased in the beginning and reaches the zenith in the second quarter of 17:00 PM. After reaching the maximum, the delay reductions dropped until the end of the PM peak period. Generally, the delay reductions did not vary significantly over time. Therefore, the proposed model appears to be effective in delay reductions in the time domain.

Table 4 Reductions in AM Peak Vehicle Delays

Time Interval	Before Optimization		After Optimization		
	Average Delay (sec/cyc)	Volume	Average Delay (sec/cyc)	Volume	Reduction in Percentage
6:00-6:15	7.72	19845	5.58	21840	23.6%
6:15-6:30	10.43	24360	5.95	25515	28.5%
6:30-6:45	13.07	26355	7.10	30345	32.6%
6:45-7:00	14.36	26355	7.77	28770	33.8%
6:00-7:00	11.40	24228.8	6.60	26617.5	29.6%
7:00-7:15	15.16	31707	7.83	33462	35.4%
7:15-7:30	14.79	36580	6.97	38350	36.8%
7:30-7:45	14.13	39120	7.31	42120	34.3%
7:45-8:00	15.67	42185	8.51	44902	34.9%
7:00-8:00	14.86	37398	7.65	39708.5	35.4%
8:00-8:15	14.58	41850	7.97	49500	31.0%
8:15-8:30	15.17	45724	8.54	50876	35.3%
8:30-8:45	14.60	46953	7.79	51465	33.9%
8:45-9:00	16.63	48900	8.40	53400	38.7%
8:00-9:00	15.25	45856.7	8.18	51310.3	34.7%
9:00-9:15	15.47	48400	7.98	48400	35.4%
9:15-9:30	16.28	48184	8.70	51680	37.1%
9:30-9:45	**17.71**	**51528**	**9.11**	**44840**	**39.2%**
9:45-10:00	17.62	45900	8.74	49800	39.0%
9:00-10:00	16.77	48503	8.63	48680	37.6%

Table 5 Reductions in PM Peak Vehicle Delays

Time Interval	Before Optimization		After Optimization		Reduction in Percentage
	Average Delay (sec/cyc)	Volume	Average Delay (sec/cyc)	Volume	
15:00-15:15	12.54	47880	8.83	48020	29.6%
15:15-15:30	16.58	51150	9.05	52950	31.1%
15:30-15:45	15.75	52624	8.55	55384	33.1%
15:45-16:00	14.88	55680	8.52	54462	32.6%
15:00-16:00	14.94	51833.5	8.74	52704	31.6%
16:00-16:15	14.71	51504	8.36	54984	33.6%
16:15-16:30	15.35	54646	9.22	55180	31.6%
16:30-16:45	16.17	55536	8.82	56960	35.9%
16:45-17:00	17.51	56108	8.99	56446	38.9%
16:00-17:00	15.94	54448.5	8.85	55892.5	35.0%
17:00-17:15	16.69	50400	7.99	53928	39.9%
17:15-17:30	**18.56**	**51012**	**9.30**	**53508**	**42.0%**
17:30-17:45	16.79	53694	8.52	51810	37.5%
17:45-18:00	16.83	52390	9.19	49755	35.4%
17:00-18:00	17.22	51874	8.75	52250.3	38.7%
18:00-18:15	15.61	47724	7.92	46084	36.5%
18:15-18:30	14.98	38038	7.83	44044	34.2%
18:30-18:45	13.84	37386	7.77	37654	31.1%
18:45-19:00	14.70	38412	8.06	38544	32.4%
18:00-19:00	14.78	40390	7.89	41581.5	33.5%

Spatial Distribution of Reductions in Vehicle Delays. Figures 10 through17 present the spatial distribution of average delay reductions within each hour in the AM and PM peak periods. As shown in this the series of visualization plots, the vehicle delay reductions appear to be stable for most of the intersections within the Chicago Loop street network. Therefore, the proposed model appears to be effective in triggering reductions in vehicle delays across the various intersections.

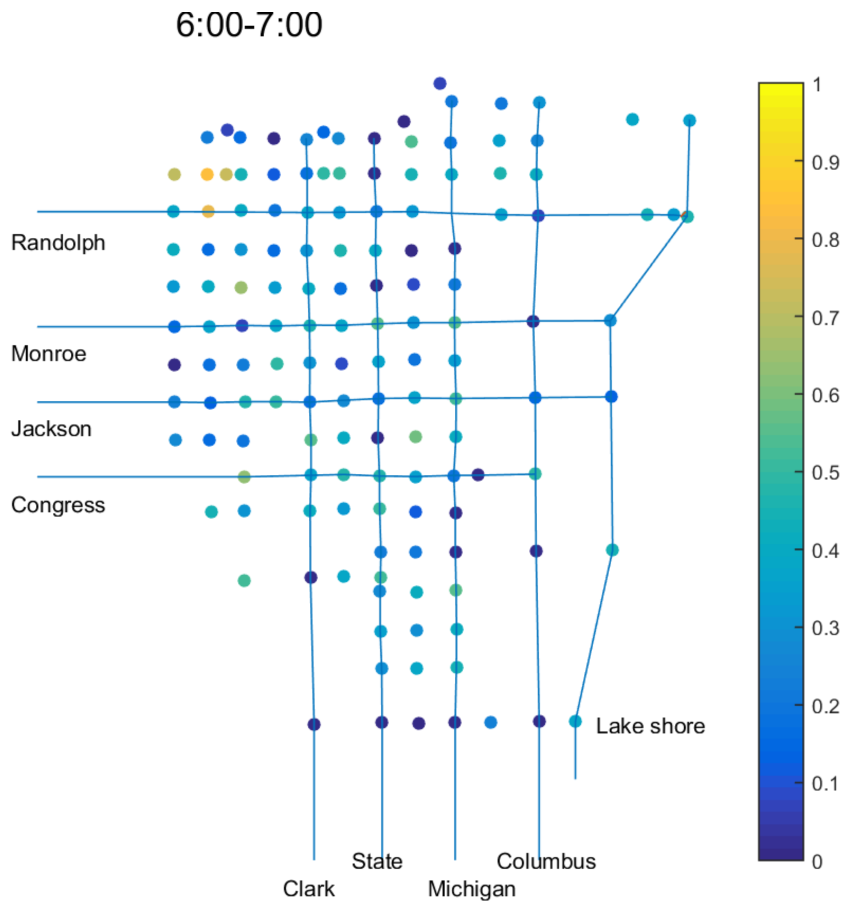

Figure 10 Spatial Distribution of Reductions in Vehicle Delays (6:00AM-7:00AM)

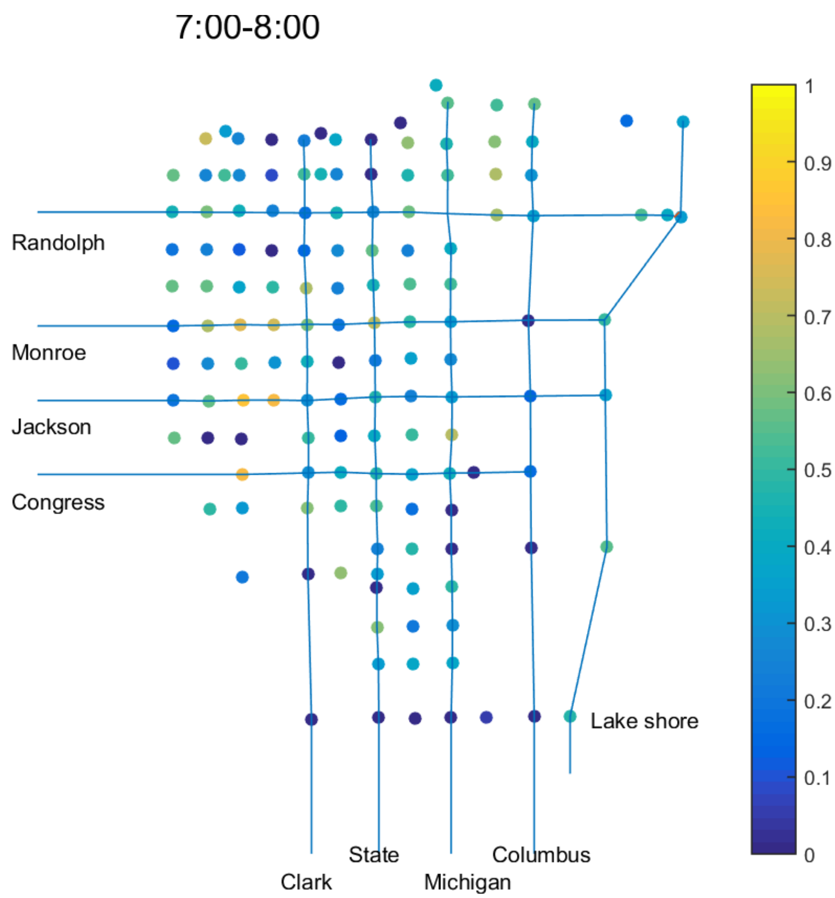

Figure 11 Spatial Distribution of Reductions in Vehicle Delays (7:00AM-8:00AM)

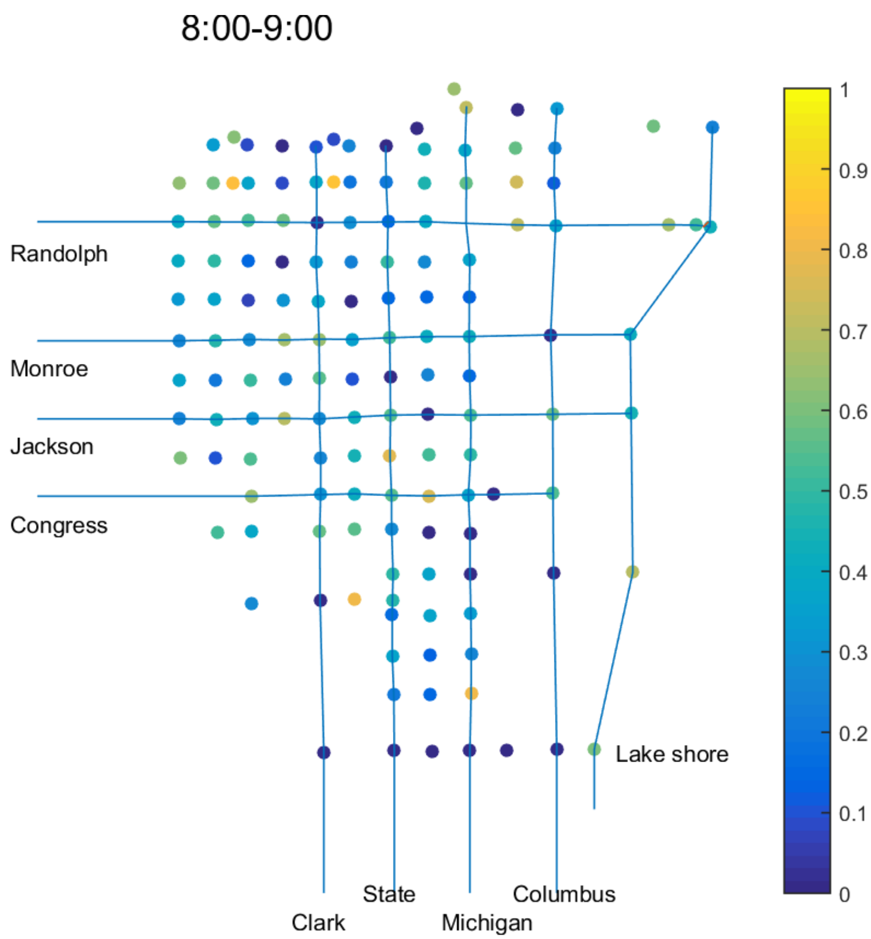

Figure 12 Spatial Distribution of Reductions in Vehicle Delays (8:00AM-9:00AM)

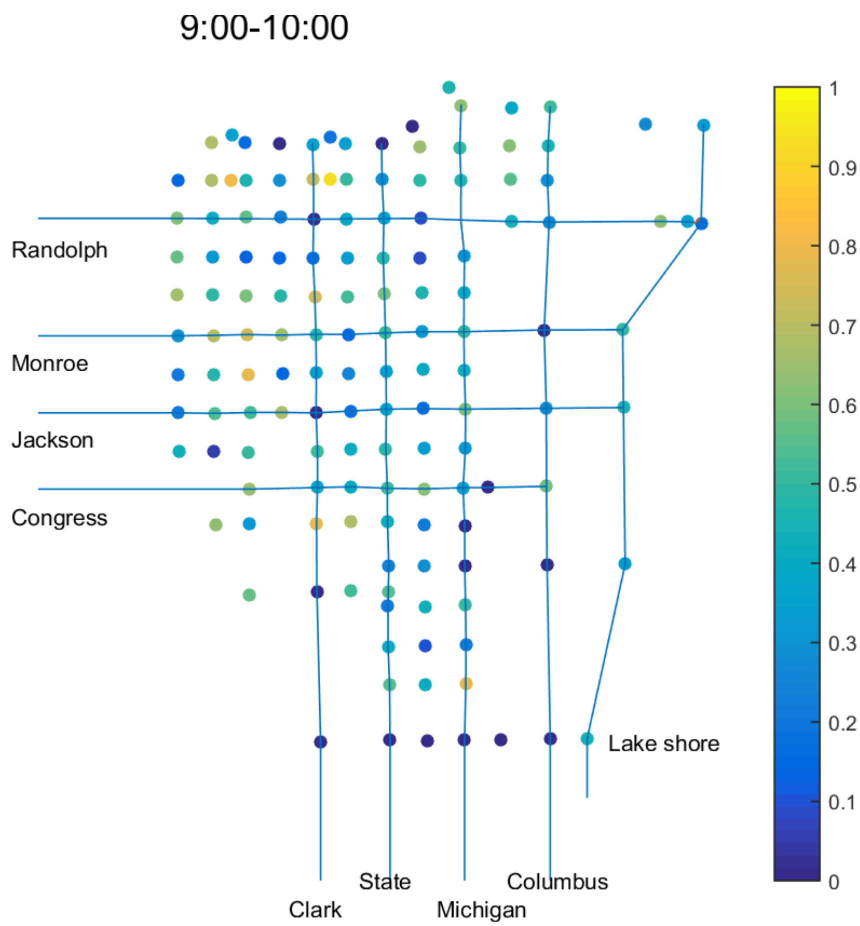

Figure 13 Spatial Distribution of Reductions in Vehicle Delays (9:00AM-10:00AM)

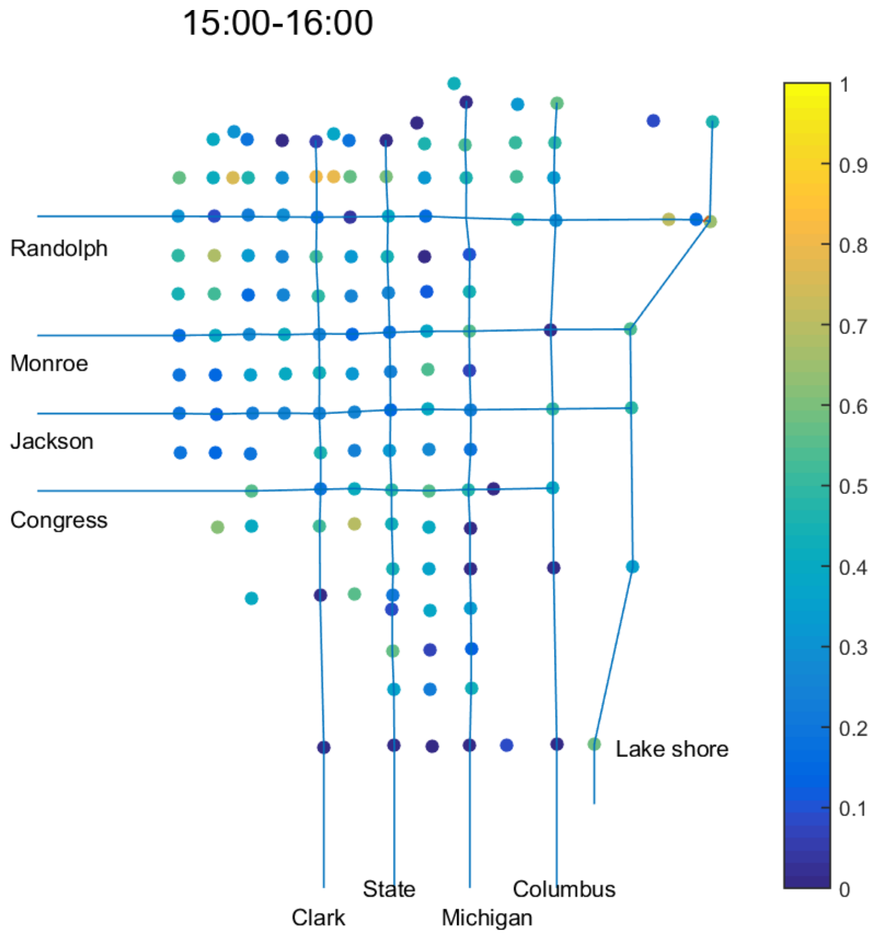

Figure 14 Spatial Distribution of Reductions in Vehicle Delays (15:00PM-16:00PM)

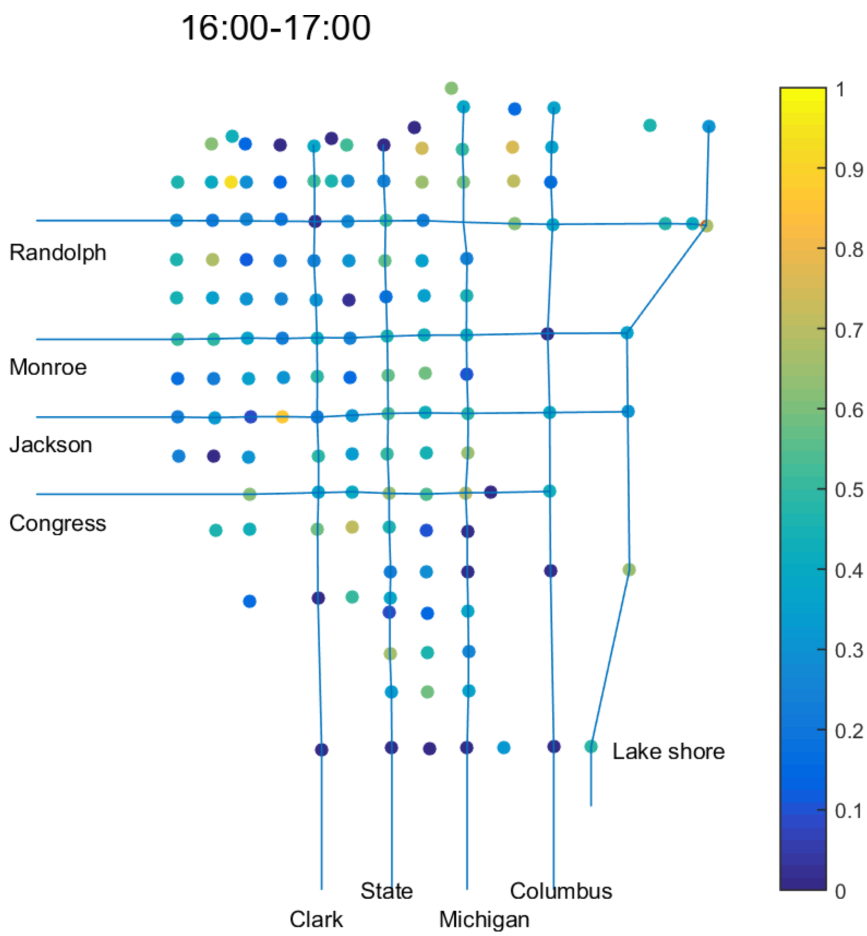

Figure 15 Spatial Distribution of Reductions in Vehicle Delays (16:00PM-17:00PM)

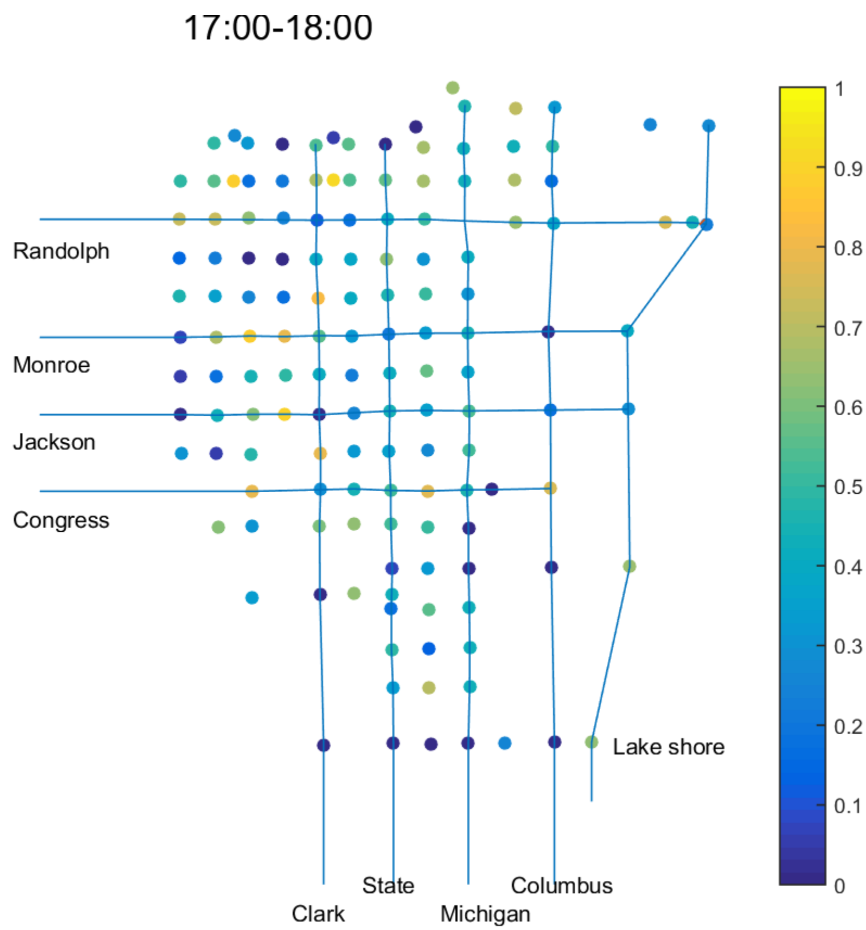

Figure 16 Spatial Distribution of Reductions in Vehicle Delays (17:00PM-18:00PM)

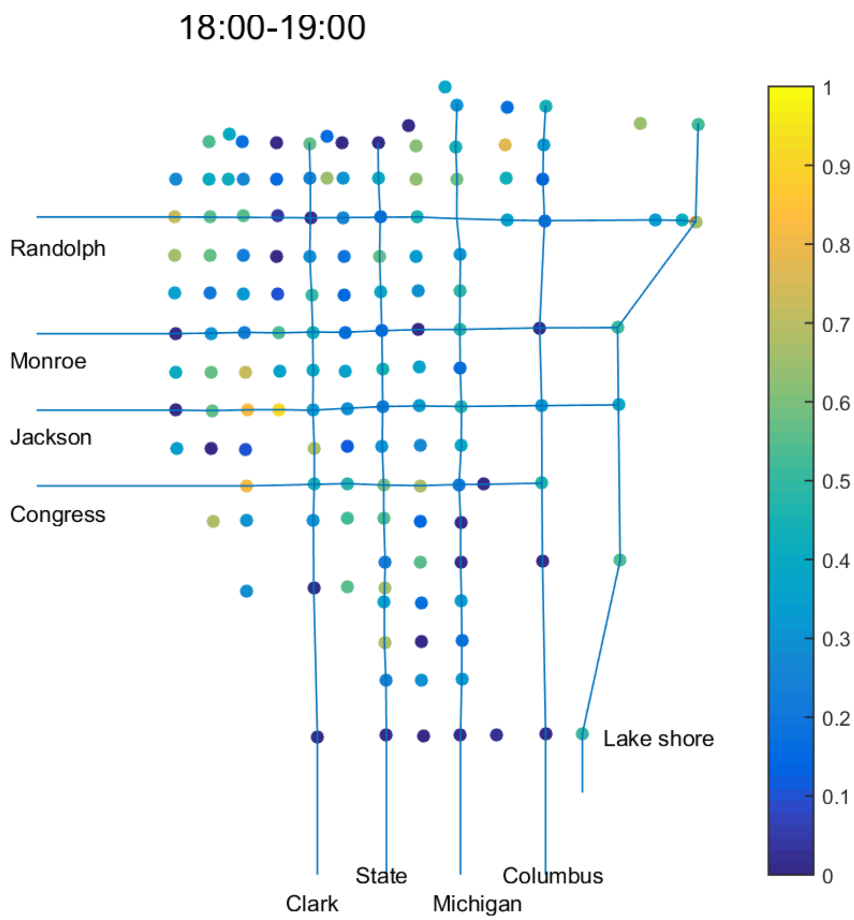

Figure 17 Spatial Distribution of Reductions in Vehicle Delays (18:00PM-19:00PM)

4.6 Discussions

In comparing the spatial distributions of delay reductions over time, it was shown that the intersections located at the outskirts of the Chicago Loop area had relatively lower delay reductions, as shown in Figures 18 and 19. The likely reason is that most of the vehicles in the boundary area prefer driving on Lakeshore Drive or Wacker Drive, which are urban expressways with greater capacities and fewer signalized intersections. Consequently, lower vehicle volumes on the boundary area streets resulted in lower levels of vehicle delays before green split optimization. As a result, the potential for delay reductions was relatively limited.

For all corridors, including North-South (Lakeshore, Columbus, Michigan, State, and Clark) and East-West (Randolph, Monroe, Jackson, and Congress) corridors within the Chicago Loop area, the delay reductions were stable over different time intervals and peak periods. The only expressway in this area, Lakeshore Drive, had the most stable delay reductions, which was unexpected because the signalized intersections on Lakeshore Drive maintain large spacing. Traffic disruptions between two successive intersections were virtually quite low and the traffic volumes on Lakeshore Drive were quite stable over time. The low traffic disruptions, coupled with the stable traffic conditions led to stable delay reductions after green split optimization.

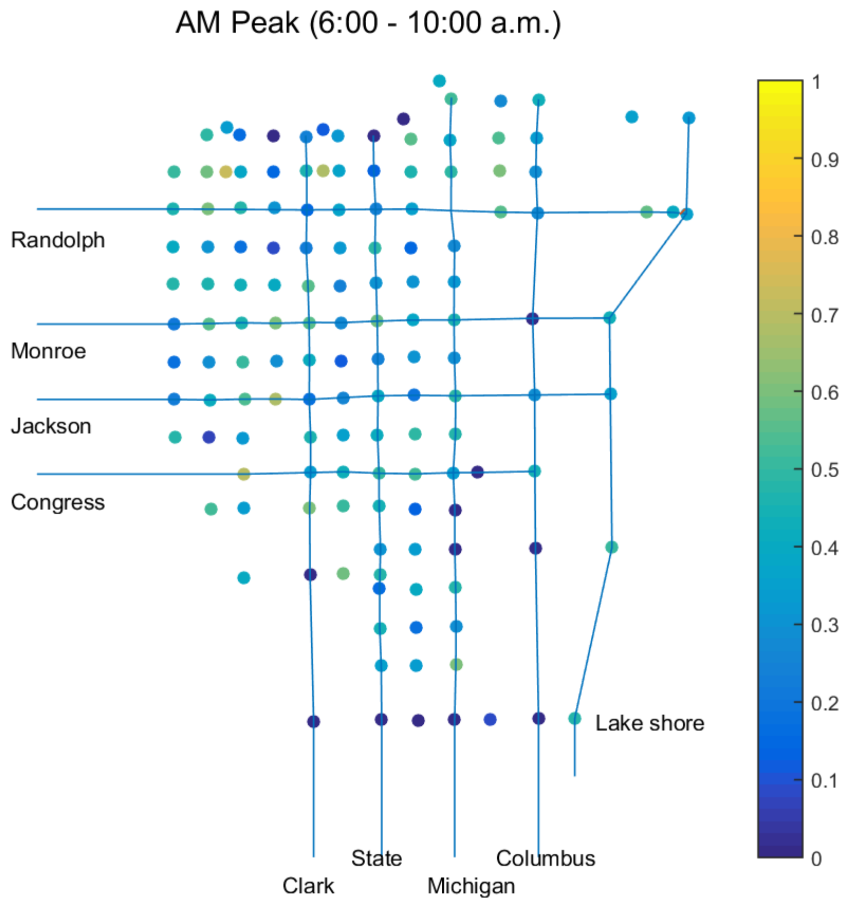

Figure 18 Spatial Distribution of Reductions in AM Peak Vehicle Delays

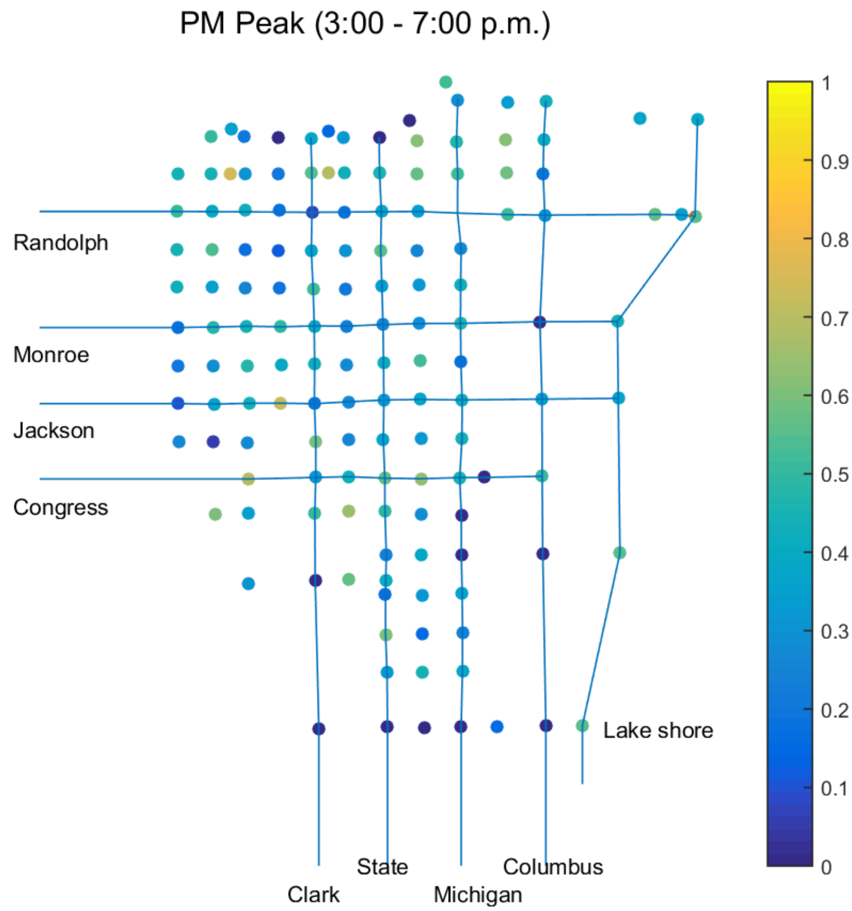

Figure 19 Spatial Distribution of Reductions in PM Peak Vehicle Delays

CHAPTER 5 SUMMARY AND CONCLUSION

5.1 Thesis Summary

This thesis first conducted a review of the existing literature on the modeling of traffic movements at signalized intersections and the optimization of intersection signal-timing plans designed to achieve the lowest extent of vehicle delays at intersections. Based on the limitations of the existing models dealing with traffic movements and signal timing optimization identified, this thesis proposed a dynamic green split optimization model. The proposed model iteratively adjusts the green splits in a signal timing design to reach the lowest level of vehicle delays per vehicle per cycle, averaged over consecutive cycles without changing the cycle length and multi-intersection signal coordination. A refined delay calculation method for vehicle delay computation and an iterative model execution process also were introduced. In addition, the iterative computation process between the pair of original green splits and the entering vehicle volumes and the new pair of green splits and updated vehicle volumes were demonstrated. The dynamic split optimization model and the iterative solution process were integrated into the TRANSIMS platform to facilitate the model's execution to derive the optimal green splits for given traffic demand conditions.

The Chicago TRANSIMS model was utilized in this thesis. The Chicago Loop street network which consists of one hundred and forty three intersections was selected as the study area for the application of the model. The traffic dynamics were captured for each given hour, by segmenting the hourly time duration into four 15-minute time intervals. Multiple rounds of green split optimization aimed to minimize the vehicle delays per vehicle per cycle averaged over consecutive cycles were performed for each 15-minute time interval in accordance with the number of signal cycles involved. The computational experiments revealed that the average vehicle delays per vehicle per cycle after green split optimization were reduced by approximately 34.5 percent.

5.2 Conclusion

Based on the above outcomes, this thesis concluded that the vehicular delays at most intersections with existing signal timing plans still have the potential for improvement. However, the extent of the delay reductions using the proposed model depends on the traffic demand at a specific intersection or the number of entering vehicles. If the demand is rather low, the delays are likely to be low, meaning that the potential for further delay reductions is low. In this respect, the proposed model may not be suitable to handle the low demand traffic conditions.

On the other hand, if the traffic demand is relatively high and all the traffic flow is oversaturated, the limited time resources will not be sufficient for reallocation to the intersection approaches demanding additional green time to reduce vehicle delays. If the intersections are independent of each other and the traffic flow is uninterrupted, the proposed model may provide a much better result in the reduction of delays because the situation was consistent with the two assumptions of ignoring the coordination of signals and maintaining relatively stable arrival rates.

5.3 Future Research Directions

The proposed model in this thesis considered only the worst case, in which a queue forms behind a stop bar caused by a red signal, and then designed a signal timing plan to minimize the overall delay. In this worst case, the vehicles traveling at a higher flow rate were assumed to arrive at the intersection by the time the red signal starts. However, in reality, that may not happen, Particularly when the signals of successive intersections are well coordinated. Therefore, purely considering the delay in the worst case as the normal delay is slightly conservative. Future research could include combining the proposed methodology with signal coordination.

The proposed model does not take into consideration special events and bus transit systems. In the model application, the predicted volumes were determined using historical data and prediction techniques, including time series and Bayesian inference. However, these techniques have some limitations with regard to addressing uncertainties such as special events. Bus transit systems play an important role in the urban transportation network, but a bus in the traffic stream may cause additional automobile delays on links, and consequently, affect the arrival rate in the intersection and lead to unintended queuing back patterns. The effect of these two factors also could be studied in future research.

In the proposed model, specific assumed flow rates were used instead of the conventional unique flow rate. Future research could focus on estimation techniques based on the conventional flow rates and other field measurements such as the left-turn, right-turn, and through-movement volumes in the upstream intersection.

The proposed model oversimplifies the shared lane capacity. For example, the model regards one shared lane (one lane shared by right-turn movement and through movement) as two separate lanes (one right-turn lane and one through-movement lane). In other words, the capacity for each movement is overestimated, which may result in underestimated delay for these two approaches in particular.

The proposed model only contains two phase-movement relationships: allowed-to-move and not-allowed-to-move. Therefore, every phase-movement relationship is categorized as one of these two types: Protected and permitted movements belong to the allowed-to-move type. In terms of the permitted left-turn vehicles, they are allowed to move in the green time but, in reality, will need to yield to the through-movement vehicles and pedestrians. However, the proposed model allows these vehicles to move right at the start of the green time. Therefore, the delays for these permitted left-turn vehicles are underestimated. With regard to the right-turn vehicles, they are permitted to make turns yielding to the perpendicular movement vehicles in their red signal. In addition, they are protected to make turns on the green. The computed delay for the right-turn vehicles may be overestimated in some locations. However, in downtown Chicago, the high pedestrian volume forces the right-turn vehicles to wait for their protected phase (often posted as “No Turn on Red” signs in their permitted phase). Therefore, it is expected that the right-turn delay will not be affected unduly in the Chicago model.

The developed model does not consider the turning bay length limitations and link length constraints. If the left-turn queue length exceeds the length of the turning bay, the newly-arriving left-turn vehicles will continue to wait at the end of the left-turn queue and will use the through movement lane; in this case, the capacity of the through movement lanes certainly will be affected. Therefore, future research should allocate space as well as time for all movements.

REFERENCES

REFERENCES

Airault, Vincent ; Stéphane Espié ; Claude Lattaud and Jean-Michel Auberlet [2004]. "Interaction between pedestrians and their environment when road-crossing: a behavioural approach." Proceedings of the 24th Urban Data Management Symposium. In E. Fendel, & M. Rumor, Italy. N.p.: n.p. N. pag. Print.

Akcelik, Rahmi [1981]. Traffic Signals: Capacity and Timing Analysis. Rep. Melbourne, Australia: Australian Road Research Board. Print. Ser. 123.

Asano, Miho, et al. [2003]. "Traffic signal control algorithm based on queuing model using ITS sensing technologies." Proceedings of the 10th World Congress and Exhibition on Intelligent Transport Systems and Services, CD-ROM.

Antonini, Gianluca; Michel Bierlaire; and Mats Weber [2006]. "Discrete choice models of pedestrian walking behavior." Transportation Research Part B: Methodological 40.8: 667-87. Print.

Ban, Xuegang; Peng Hao; and Zhanbo Sun [2011]. "Real time queue length estimation for signalized intersections using travel times from mobile sensors." Transportation Research Part C: Emerging Technologies 19.6: 1133-156. Print.

Banks, Robert [2013] Towing Icebergs, Falling Dominoes, and Other Adventures in Applied Mathematics. Princeton, NJ: Princeton University Press.

Chen, Juan and Lihong Xu [2006]. "Road-junction traffic signal timing optimization by an adaptive particle swarm algorithm." Control, Automation, Robotics and Vision,2006. ICARCV'06. 9th International Conference on. IEEE.

Chin, Yit Kwong, et al. [2011]. "Exploring Q-learning optimization in traffic signal timing plan management." Computational Intelligence, Communication Systems and Networks (CICSyN), 2011 Third International Conference on. IEEE.

Curio, Cristobal; Johann Edelbrunner; Thomas Kalinke; Christos Tzomakas; and Werner Von Seelen [2000]. "Walking pedestrian recognition." IEEE Transactions On Intelligent Transportation 1.3: 155-63. Print.

Desyllas, Jake; Elspeth Duxbury; John Ward; and Andrew Smith [2003]. Pedestrian Demand Modelling of Large Cities: An Applied Example from London. Rep. London, UK: Centre for Advanced Spatial Analysis University College London. Print.

Dijkstra, Jan, and Harry Timmermans [2002]. "Towards a Multi-agent Model for Visualizing Simulated User Behavior to Support the Assessment of Design Performance." Automation in Construction 11.2: 135-45. Print.

Dion, Francois; Hesham Rakha; and Youn-Soo Kang [2004]. "Comparison of delay estimates at undersaturated and oversaturated pre-timed signalized intersections." Transportation Research Part B: Methodological 38.2: 99-122. Print.

Dong, Chaojun; Shiqing Huang; and Xiankun Liu [2010]. "Urban area traffic signal timing optimization based on Sa-PSO." Proc. of 2010 International Conference on Artificial Intelligence and Computational Intelligence. Vol. 3. N.p.: IEEE. 80-84. Print.

Gartner, Nathan [1983]. OPAC: A demand-responsive strategy for traffic signal control. No. 906.

Garber, Nicholas and Lester Hoel [2002]. Traffic and Highway Engineering. Pacific Grove, CA: Brooks/Cole Publishing. Print.

Garber, Nicholas, and Lester Hoel [2014]. Traffic and Highway Engineering, 5th Edition. Boston, MA: Cengage Learning.

Hisai, Mamoru and Satoshi Sasaki [1993]. Shock Wave Propagation Analysis in Coordinated Signal Systems by Kinematic Wave Theory. Rep. 2nd ed. Vol. 5. N.p.: Technology Reports of the Yamaguchi University. Print.

Hoogendoorn, Serge [2004]. "Pedestrian flow modeling by adaptive control." Proc. Of TRB 2004 Annual Meeting, Washington DC. N.p.: n.p. N. pag. Print.

Hoogendoorn, Serge and Piet Bovy [2004]. "Pedestrian route-choice and activity scheduling theory and models." Transportation Research Part B: Methodological 38.2: 169-90. Print.

Hughes, Ronald G., and Nagui M. Rouphail [2002]. Evaluation and Application of Pedestrian Modeling Capabilities Using Computer Simulation. Rep. N.p.: WesternMichigan University. Print.

Hunt, P. B., et al. [1981]. SCOOT- A traffic responsive method of coordinating signals. No. LR 1014 Monograph.

King, Michael; Jon Carnegie; and Reid Ewing [2003]. "Pedestrian safety through a raised median and redesigned intersection." Transportation Research Record 1828.1: 56-66. Print.

Kitazawa, Kay and Michal Batty [2004]. "Pedestrian behaviour modelling: an Application to retail movements using a genetic algorithm." Proc. of 7th International Conference on Design and Decision Support Systems in Architecture and Urban Planning. N.p.: n.p. N. pag. Print.

Koshi, Masaki [1972]. "On-line feedback control of offsets for area control of traffic." Traffic Flow and Transportation.

Koshi, Masaki [1989]. "Cycle time optimization in traffic signal coordination." Transportation Research Part A: General 23.1: 29-34.

Kukla, Robert, Jon Kerridge, Alex Willis, and Julian Hine [2001]. "PEDFLOW: Development of an autonomous agent model of pedestrian flow." Transportation Research Record 1774.1: 11-17. Print.

Lave, Charles [1969]. “A behavioral approach to modal split forecasting.” Transportation Research 3, 463-480.

Lertworawanich, Ponlathep [2010]. "A traffic signal split optimization using time-space diagrams." European Transport Conference, 2010.

Li, Meiling; Jian Rong; Zhenhua Mou; and Jin Ran [2010]. "A simulation optimization system of signalized intersections based on knowledge." Proceedings of the 10th International Conference of Chinese Transportation Professionals. N.p.: n.p.. N. pag. Print

Li, Zongzhi; David Zavattero; Kermit Wies; Young-Jun Son and Herbert Levinson [2012]. “Development and Application of the TRANSIMS Toolbox for Transportation Operations Management in and around Chicago Central Area.” Phase I Final Report. Washington, DC: Federal Highway Administration, U.S. Department of Transportation, Print.

Liu, Henry; ., Xinkai Wu; Wenteng Ma; and Heng Hu [2009]. "Real-time queue length estimation for congested signalized intersections." Transportation Research Part C: Emerging Technologies 17.4: 412-27. Print.

Lum, Kelvin and Harun Halim [2006]. "A before-and-after study on green signal countdown device installation." Transportation Research Part F: Traffic Psychology and Behaviour 9.1: 29-41. Print.

Michalopoulos, Panos; Gregory Stephanopoulos; and George Stephanopoulos [1981]. "An application of shock wave theory to traffic signal control." Transportation Research Part B: Methodological 15.1: 35-51. Print.

Montella, Alfonso [2005]. "Safety reviews of existing roads: quantitative safety assessment methodology." Transportation Research Record 1922.1: 62-72. Print.

Mussa, Renatus and Majura Selekwa [2003]. "Proposed methodology of optimizing transitioning between time-of-day timing plans." Journal of Transportation Engineering 129.4: 392. Print.

Okazaki, Shigeyuki and Satoshi Matsushita [1993]. "A study of simulation model for pedestrian movement with evacuation and queuing." Engineering for Crowd Safety 1: 271-80. Print.

Osaragi, Toshihiro [2004]. "Modeling of pedestrian behavior and its applications to spatial evaluation." Proceedings of the 3rd International Joint Conference on Autonomous Agents and multi-agent Systems. N.p.: n.p. N. pag. Print.

Pant, Prahlad; Cheng Yizong; and Naharaju Kashayi [2005]. “Field Testing and Implementation of Dilemma Zone Protection and Signal Coordination at Closely-Spaced High-Speed Intersections.” Rep. Cincinnati: Univ. of Cincinnati. Print. FHWA/OH-2005/006.

Qin, Xiao and John Ivan [2001]. "Estimating pedestrian exposure prediction model in rural areas." Transportation Research Record 1773.1: 89-96. Print.

Roshandeh, Arash; Herbert Levinson; Zongzhi Li; Harshingar Patel; and Bei Zhou [2014]. “A new methodology for intersection signal timing optimization to simultaneously minimize vehicle and pedestrian delays.” ASCE Journal of Transportation Engineering, 140(5), 04014009.

Schrank, David; Bill Eisele; Tim Lomax; and Jim Bak [2015]. “Urban Mobility Scorecard.” Technical Report August, Texas A&M Transportation Institute and INRIX, Inc.

Sun, Dazhi; Rahim Benekohal; and S. Travis Waller [2003]. "Multiobjective traffic signal timing optimization using non-dominated sorting genetic algorithm." Proceedings of Intelligent Vehicles Symposium, 2003.. IEEE.

Teknomo, Kardi [2006]. "Application of microscopic pedestrian simulation model." Transportation Research Part F: Traffic Psychology and Behaviour 9.1: 15-27. Print.

Teknomo, Kardi [2006]. "Application of microscopic pedestrian simulation model." Transportation Research Part F: Traffic Psychology and Behaviour 9.1: 15-27. Print.

TranSystems and TransInfo LLC. [2008]. Pedestrian Activity in Chicago's Downtown. Rep. Chicago: Prepared for Chicago Dept. of Transportation. Print.

TRB[2008]. Highway Capacity Manual. Transportation Research Board, Washington DC. Print.

TRB [2000]. Highway Capacity Manual. Transportation Research Board, Washington, DC. Print.

Wirasinghe, S. Chandana [1978]. "Determination of traffic delays from shock-wave analysis." Transportation Research 12.5: 343-348.

Wu, Xinkai; Henry Liu; and Douglas Gettman [2010]. "Identification of oversaturated intersections using high-resolution traffic signal data." Transportation Research Part C: Emerging Technologies 18.4: 626-38. Print.

Xiao-feng, Chen and Shi Zhong-ke [2009]. "A hybrid optimization method and application in traffic signal timings optimization." Proc. of Computational Intelligence and Software Engineering. N.p.: n.p. 1-4. Print.

Zegeer, Charles [1978]. "Green-extension systems at high speed intersections." Institute of Transportation Engineers 11: 19-24. Print.

APPENDICES

Appendix A Typical Intersection Signal Timing Plans

Subarea_signal: the file recording for each intersection within the network, the effective time of each signal timing plan.

NODE	START	TIMING	TYPE	RINGS	OFFSET	COORDINATOR	NOTES
14521	0:00	27006	T	S	0	27006	4 Phase Timed
14521	6:00	27007	T	S	0	27007	4 Phase Timed
14521	10:00	27008	T	S	0	27008	4 Phase Timed
14521	15:00	27009	T	S	0	27009	4 Phase Timed
14521	19:00	27010	T	S	0	27010	4 Phase Timed
14532	0:00	27056	T	S	0	27056	4 Phase Timed
14532	6:00	27057	T	S	0	27057	4 Phase Timed
14532	10:00	27058	T	S	0	27058	4 Phase Timed
14532	15:00	27059	T	S	0	27059	4 Phase Timed
14532	19:00	27060	T	S	0	27060	4 Phase Timed
14533	0:00	27061	T	S	0	27061	4 Phase Timed
14533	6:00	27062	T	S	0	27062	4 Phase Timed
14533	10:00	27063	T	S	0	27063	4 Phase Timed
14533	15:00	27064	T	S	0	27064	4 Phase Timed
14533	19:00	27065	T	S	0	27065	4 Phase Timed
14535	0:00	27071	T	S	0	27071	4 Phase Timed

Subarea_timing: the file recording the phase configuration of each timing plan

TIMING	PHASE	NEXT_PHASE	MIN_GREEN	MAX_GREEN	EXT_GREEN	YELLOW	RED_CLEAR	RING	BARRIER	NOTES
27006	1	2	8	0	0	0	0	1	0	NODE 14521
27006	2	3	22	0	0	3	1	0	0	NODE 14521
27006	3	4	15	0	0	0	0	0	0	NODE 14521
27006	4	1	22	0	0	3	1	0	0	NODE 14521
27007	1	2	8	0	0	0	0	1	0	NODE 14521
27007	2	3	22	0	0	3	1	0	0	NODE 14521
27007	3	4	15	0	0	0	0	0	0	NODE 14521
27007	4	1	22	0	0	3	1	0	0	NODE 14521
27008	1	2	8	0	0	0	0	1	0	NODE 14521

Subarea_phasing: the file recording the movement-phase relationship for each timing

NODE	TIMING	PHASE	IN LINK	OUT LINK	PROTECTION	DETECTORS	NOTES
14521	27006	1	15302	15301	P	0	Protected Left
14521	27006	2	15302	15301	U	0	Unprotected Left
14521	27006	2	15302	15084	P	0	Protected Thru
14521	27006	2	15302	15074	P	0	Protected Right
14521	27006	4	15302	15074	S	0	Right on Red
14521	27006	3	15301	15084	P	0	Protected Left
14521	27006	4	15301	15084	U	0	Unprotected Left
14521	27006	4	15301	15074	P	0	Protected Thru
14521	27006	4	15301	15302	P	0	Protected Right
14521	27006	2	15301	15302	S	0	Right on Red
14521	27006	1	15084	15074	P	0	Protected Left
14521	27006	2	15084	15074	U	0	Unprotected Left
14521	27006	2	15084	15302	P	0	Protected Thru
14521	27006	2	15084	15301	P	0	Protected Right
14521	27006	4	15084	15301	S	0	Right on Red
14521	27006	3	15074	15302	P	0	Protected Left
14521	27006	4	15074	15302	U	0	Unprotected Left
14521	27006	4	15074	15301	P	0	Protected Thru
14521	27006	4	15074	15084	P	0	Protected Right
14521	27006	2	15074	15084	S	0	Right on Red
14521	27007	1	15302	15301	P	0	Protected Left

Appendix B Python Code for the Application

```
class Node:
    def __init__(self, id):
        self.id = id
        self.timing_map = {}
        ss_file = open('initial/subarea_signal', 'rb')
        timing2hour = {}
        time_bound = []
        for row in ss_file:
            r = row.split('\t')  # split signal in rows
            if r[0] == self.id:  #
r=[Node,Start,Timing,Type,Rings,Offset,Coordinator,Notes]
                time_bound.append(int(r[1][:-3]))  # r[1]=0:00 or 19:00
                timing2hour[r[2]] = int(r[1][:-3])  # timing ID ->hour
        start_end = {}  # Timing plan start time-> timing effective hour list
        tmp = []
        for i in time_bound:  # Achieve start_end, timing start hour-> timing effective
hour list
            starttime = i
            add = True
            tmp.append(i)
            small = [i]
            while add:
```

```
                i += 1
                if i in time_bound or i in tmp or i == 24:
                    add = False
                    start_end[starttime] = small
                else:
                    tmp.append(i)
                    small.append(i)
        for i in timing2hour.keys():  # i is timing
            self.timing_map[i] = Timing(i, min(start_end[timing2hour[i]]) * 60,
                                        60 * max(start_end[timing2hour[i]]) + 60)  # timing ->
Timing object
        sp_file = open('initial/subarea_phasing', 'rb')
        for row in sp_file:
            r = row.split('\t')  # r=[Node,Timing, Phase, In_link,Out_link, protected,..]
            if r[0] == self.id and (r[3], r[4]) not in
self.timing_map[r[1]].phase_map[r[2]].link_pair:
                self.timing_map[r[1]].phase_map[r[2]].link_pair.append((r[3], r[4]))
                # To record all (In_link, Out_link) of node.timing.phase to
node.timing.phase.link_pair


   class Timing:
       def __init__(self, id, start_min, end_min):  # hours is a list contains the time this
timing plan works
```

```
        self.id = id
        self.start_time = start_min
        self.end_time = end_min
        self.phase_map = {}
        self.cycle_time = 0
        self.offset = 0
        st_file = open('initial/subarea_timing', 'rb')
        for row in st_file:
            r = row[:-1].split('\t')
            if r[0] == self.id:  # r=['TIMING', 'PHASE', 'NEXT_PHASE', 'MIN_GREEN',
                #  'MAX_GREEN', 'EXT_GREEN', 'YELLOW', 'RED_CLEAR', 'RING',
'BARRIER', 'NOTES']
                self.phase_map[r[1]] = Phase(r[1], r[2], int(r[3]), int(r[4]), int(r[5]), int(r[6]),
int(r[7]),
                                      int(r[8]), int(r[9]))
                self.cycle_time += int(r[3]) + int(r[6]) + int(r[7])

    def delay(self, time_slot):
        total = 0  # total delay
        nv = 0  # number of vehicles
        for phase in self.phase_map.values():  # (self, qmax, qvc,v,lane ,TR,cyc)
            TR = self.cycle_time - phase.green - phase.yellow
            for ll in phase.vol[time_slot]:  # ([qmax,qvc,v,ln])
```

```
            if ll != []:
                l = ll
                result = delay_function(l[0] * 100, l[1] * 100, l[2], l[3], TR,
self.cycle_time)
                total += result['delay']
                nv += result['number of veh']
        if total == 0:
            return 0.0
        else:
            return total / nv/5.0



class Phase:
    def __init__(self, id, next, green, maxgreen, extgreen, yellow, red_clear, ring,
barrier):
        self.id = id
        self.next = next
        self.green = green
        self.maxgreen = maxgreen
        self.extgreen = extgreen
        self.yellow = yellow
        self.red_clear = red_clear
        self.ring = ring
```

```
        self.barrier = barrier
        self.delay = 99999.0
        self.link_pair = []
        self.vol = {}
        self.check1 = False
        self.status = 'None'
        self.check2 = False
        self.check3 = False


def delay_function(qmax, qvc, v, lane, TR, cyc):  # Compute the delay
    kj = 150  # veh/km/lane
    v = v*3.6 # km/h
    kmax = float(qmax / v)  # veh/km/lane
    qh = 0.5 * (qvc + qmax) # veh/h/lane
    ql = qvc / 3.0  # veh/h/lane
    kh = float(qh / v)  # veh/km/lane
    kl = float(ql / v)  # veh/km/lane
    f = float((qvc - ql) / (qh - ql))
    v1 = qh / (kj - kh)/3.6  # in m/s
    v2 = ql / (kj - kl)/3.6  # in m/s
    v3 = qmax / (kj - kmax)/3.6  # in m/s
    v4 = v/3.6  # in m/s
```

```
        ta = float(v4 * f * cyc / (v1 + v4))  # in sec
        sa = float(ta * v1)  # in meter
        tb = float((sa + v3 * TR - v2 * ta) / (v3 - v2))  # in sec
        sb = v3 * (tb - TR)  # in meter
        tc = float(sb / v4) + tb  # in sec
        area = 0.5 * kj * ((TR + tb) * sb - sa * ta - (ta + tb) * (sb - sa))/1000  # in veh*sec
        if tc <= cyc:
            Delay = 2 * area * lane
            return {'delay': Delay, 'number of veh': (sb * kj / 1000 * lane * 2),
                    'status': 'Undersaturation'}  # Total delay within 10 cycles
        else:
            sc = float((tc - cyc) * v3 * v4 / (v3 + v4))
            Delay = (2 * area + 1 * sc * TR) * lane
            return {'delay': Delay, 'number of veh': (sb * kj / 1000 * lane * 2 + sc * kj / 1000
* 1 * lane),
                    'status': 'Oversaturation'}


    class Network:
        def __init__(self, nodes):
            # type: (object) -> object
            print 'Initializing network..'
            self.nodes = nodes
```

```
        self.node_map = {}
        for n in nodes:
            self.node_map[n] = Node(n)  # node->Node object
        self.link_map = {}
        turn = {}  # (NODE,IN_LINK,OUT_LINK,START_minutes)->VOLUME
        link = {}  # linkID-
>[ANODE,BNODE,LENGTH,LANES_AB,LEFT_AB,RIGHT_AB,LANES_BA,LEFT_
BA,RIGHT_BA]
        lane = {}  # number of lanes of (node,In link ,Out link)
        cap = {}  # (lINK,BNODE)->capcity of link going to B
        spd = {}  # (lINK,BNODE)->free flow speed of link going to B
        tv_file = open('initial/subarea_turn', 'rb')
        lk_file = open('initial/link', 'rb')
        lane_file = open('initial/subarea_connectivity', 'rb')
        link_file = open('initial/subarea_link', 'rb')
        for row in link_file:
            r = row[:-2].split('\t')
            if r[10] == '0' or r[15] == 'CAP_AB':
                cap[(r[0], r[3])] = 0
            else:
                cap[(r[0], r[3])] = int(r[15]) / float(r[10])
            if r[16] == '0' or r[21] == 'CAP_BA':
                cap[(r[0], r[2])] = 0
```

```
            else:
                cap[(r[0], r[2])] = int(r[21]) / float(r[16])
            if not r[14] == 'FSPD_AB':
                spd[(r[0], r[3])] = float(r[14])
                spd[(r[0], r[2])] = float(r[20])
        for row in lane_file:
            r = row[:-2].split('\t')
            if r[0] not in self.nodes: continue
            if (r[0], r[1], r[2]) in lane.keys():
                lane[(r[0], r[1], r[2])] += 1
            else:
                lane[(r[0], r[1], r[2])] = 1
        for row in lk_file:
            r = row[:-2].split('\t')
            link[r[0]] = [r[1:]]
        for row in tv_file:
            r = row[:-2].split(',')
            if r[0] in self.nodes: turn[(r[0], r[1], r[2], int(r[3]) / 60)] = int(r[-1]) * 4  #
equivalent hourly rate
        for node in self.node_map.values():  # For all node objects
            for time in node.timing_map.values():  # For all timing objects
                for phase in time.phase_map.values():  # For all phase objects
                    for h in range(360, 600, 15) + range(900, 1140, 15):
```

```
            temp = []  # Contain the input to calculate the delay
            for l in phase.link_pair:
                if (node.id, l[0], l[1], h) not in turn.keys(): continue
                qmax = cap[(l[0], node.id)]
                v = spd[(l[0], node.id)]
                if v == 0:
                    v = 11.4
                ln = lane[(node.id, l[0], l[1])]
                ii=turn[(node.id, l[0], l[1], h)]
                qvc = float(ii) / ln
                if qvc>qmax:
                    qmax=qvc+1.25
                temp.append([qmax, qvc, v, ln])
            phase.vol[h] = temp
            # self.__delay()
            # self.__update()


# self.connection_table=[]
def CreatNewTimingID(self):
    Original_TimingID = []
    ss_file = open('initial/subarea_signal', 'rb')
    for row in ss_file:
        r = row.split('\t')  # split signal in cells
```

```
    Original_TimingID.append(r[2])
NewTimingID = 10000
time_step1 = [0, 360]  # 00:00 to 6:00 in minutes
time_step2 = range(370, 540, 15)  # 6:00 to 9:00 every 10 minutes (in minutes)
time_step3 = [540, 960]  # 09:00 to 16:00 in minutes
time_step4 = range(970, 1140, 15)  # 16:00 to 19:00 every 10 minutes (in
minutes)
time_step5 = [1140, 1440]  # 19:00 to 24:00 in minutes
t = time_step1 + time_step2 + time_step3 + time_step4 + time_step5
time_step = []
for i in range(len(t) - 1):
    time_step.append([t[i], t[i + 1]])
st_file = open('initial/subarea_timing', 'rb')
sp_file = open('initial/subarea_phasing', 'rb')
new_ss_file = 'subarea_signal_new'
my_ss_writer = open(new_ss_file, "wb")
new_st_file = 'subarea_timing_new'
my_st_writer = open(new_st_file, "wb")
new_sp_file = 'subarea_phase_new'
my_sp_writer = open(new_sp_file, "wb")
# update subarea_signal
ss_file = open('initial/subarea_signal', 'rb')
for row in ss_file:
```

```
        r = row.split('\t')  # split signal in cells
        if r[0] in self.nodes:
            for ts in time_step:  # Crack the original interval into pieces and assign new
Timing ID
                while NewTimingID in Original_TimingID: NewTimingID += 1
                Original_TimingID.append(NewTimingID)
                if ts[0] >= 0 and ts[1] <= 360:
                    for s in self.node_map[r[0]].timing_map.values():
                        if s.start_time == 0: break
                    self.node_map[r[0]].timing_map[NewTimingID] = s
                    self.node_map[r[0]].timing_map[NewTimingID].start_time = ts[0]
                    self.node_map[r[0]].timing_map[NewTimingID].end_time = ts[1]
                    #  Update subarea_signal
                    cell = range(8)
                    cell[0] = str(r[0])
                    stt = self.node_map[r[0]].timing_map[NewTimingID].start_time
                    if len(str(stt % 60)) == 1:
                        cell[1] = ':'.join([str(stt / 60), '0' + str(stt % 60)])
                    else:
                        cell[1] = ':'.join([str(stt / 60), str(stt % 60)])
                    cell[2] = str(NewTimingID)
                    cell[3] = r[3]
                    cell[4] = r[4]
```

```
cell[5] = r[5]
cell[6] = cell[2]
cell[7] = r[7]
newrow = '\t'.join(cell)
my_ss_writer.write(newrow)
#  Update subarea_timing
st_file = open('initial/subarea_timing', 'rb')
find = False
for row1 in st_file:
    r1 = row1.split('\t')
    if r1[0] == str(s.id):
        r1[0] = str(NewTimingID)
        my_st_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[0] != str(s.id): break
# Update subarea_phasing
sp_file = open('initial/subarea_phasing', 'rb')
find = False
for row1 in sp_file:
    r1 = row1.split('\t')
    if r1[1] == str(s.id):
        r1[1] = str(NewTimingID)
```

```
            my_sp_writer.write('\t'.join(r1))
            find = True
        r1 = row1.split('\t')
        if find and r1[1] != str(s.id): break

elif ts[0] >= 360 and ts[1] <= 600:
    for s in self.node_map[r[0]].timing_map.values():
        if s.start_time == 360: break
    self.node_map[r[0]].timing_map[NewTimingID] = s
    self.node_map[r[0]].timing_map[NewTimingID].start_time = ts[0]
    self.node_map[r[0]].timing_map[NewTimingID].end_time = ts[1]

    cell = range(8)
    cell[0] = str(r[0])
    stt = self.node_map[r[0]].timing_map[NewTimingID].start_time
    if len(str(stt % 60)) == 1:
        cell[1] = ':'.join([str(stt / 60), '0' + str(stt % 60)])
    else:
        cell[1] = ':'.join([str(stt / 60), str(stt % 60)])

    cell[2] = str(NewTimingID)
    cell[3] = r[3]
    cell[4] = r[4]
```

```
cell[5] = r[5]
cell[6] = cell[2]
cell[7] = r[7]

newrow = '\t'.join(cell)
my_ss_writer.write(newrow)
#  Update subarea_timing
st_file = open('initial/subarea_timing', 'rb')
find = False
for row1 in st_file:
    r1 = row1.split('\t')
    if r1[0] == str(s.id):
        r1[0] = str(NewTimingID)
        my_st_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[0] != str(s.id): break
# Update subarea_phasing
find = False
sp_file = open('initial/subarea_phasing', 'rb')
for row1 in sp_file:
    r1 = row1.split('\t')
    if r1[1] == str(s.id):
```

```
            r1[1] = str(NewTimingID)
            my_sp_writer.write('\t'.join(r1))
            find = True
        r1 = row1.split('\t')
        if find and r1[1] != str(s.id): break

elif ts[0] == 540 and ts[1] == 960:
    for s in self.node_map[r[0]].timing_map.values():  # Break the interval
into [540,600]
        if s.start_time == 360: break
    self.node_map[r[0]].timing_map[NewTimingID] = s
    self.node_map[r[0]].timing_map[NewTimingID].start_time = 540
    self.node_map[r[0]].timing_map[NewTimingID].end_time = 600
    cell = range(8)
    cell[0] = str(r[0])
    stt = self.node_map[r[0]].timing_map[NewTimingID].start_time
    if len(str(stt % 60)) == 1:
        cell[1] = ':'.join([str(stt / 60), '0' + str(stt % 60)])
    else:
        cell[1] = ':'.join([str(stt / 60), str(stt % 60)])

    cell[2] = str(NewTimingID)
    cell[3] = r[3]
```

```
cell[4] = r[4]
cell[5] = r[5]
cell[6] = cell[2]
cell[7] = r[7]

newrow = '\t'.join(cell)
my_ss_writer.write(newrow)
#  Update subarea_timing
st_file = open('initial/subarea_timing', 'rb')
find = False
for row1 in st_file:
   r1 = row1.split('\t')
   if r1[0] == str(s.id):
      r1[0] = str(NewTimingID)
      my_st_writer.write('\t'.join(r1))
      find = True
   r1 = row1.split('\t')
   if find and r1[0] != str(s.id): break
# Update subarea_phasing
find = False
sp_file = open('initial/subarea_phasing', 'rb')
for row1 in sp_file:
   r1 = row1.split('\t')
```

```
    if r1[1] == str(s.id):
        r1[1] = str(NewTimingID)
        my_sp_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[1] != str(s.id): break


while NewTimingID in Original_TimingID: NewTimingID += 1
Original_TimingID.append(NewTimingID)
for s in self.node_map[r[0]].timing_map.values():  # Break the interval
into [600,900]
    if s.start_time == 600: break
self.node_map[r[0]].timing_map[NewTimingID] = s
self.node_map[r[0]].timing_map[NewTimingID].start_time = 600
self.node_map[r[0]].timing_map[NewTimingID].end_time = 900
cell = range(8)
cell[0] = str(r[0])
stt = self.node_map[r[0]].timing_map[NewTimingID].start_time
if len(str(stt % 60)) == 1:
    cell[1] = ':'.join([str(stt / 60), '0' + str(stt % 60)])
else:
    cell[1] = ':'.join([str(stt / 60), str(stt % 60)])
```

```
cell[2] = str(NewTimingID)
cell[3] = r[3]
cell[4] = r[4]
cell[5] = r[5]
cell[6] = cell[2]
cell[7] = r[7]

newrow = '\t'.join(cell)
my_ss_writer.write(newrow)
#  Update subarea_timing
find = False
st_file = open('initial/subarea_timing', 'rb')
for row1 in st_file:
    r1 = row1.split('\t')
    if r1[0] == str(s.id):
        r1[0] = str(NewTimingID)
        my_st_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[0] != str(s.id): break
# Update subarea_phasing
sp_file = open('initial/subarea_phasing', 'rb')
find = False
```

```
for row1 in sp_file:
    r1 = row1.split('\t')
    if r1[1] == str(s.id):
        r1[1] = str(NewTimingID)
        my_sp_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[1] != str(s.id): break

while NewTimingID in Original_TimingID: NewTimingID += 1
Original_TimingID.append(NewTimingID)
for s in self.node_map[r[0]].timing_map.values():  # Break the interval
into [900,960]
    if s.start_time == 900: break
self.node_map[r[0]].timing_map[NewTimingID] = s
self.node_map[r[0]].timing_map[NewTimingID].start_time = 900
self.node_map[r[0]].timing_map[NewTimingID].end_time = 960
cell = range(8)
cell[0] = str(r[0])
stt = self.node_map[r[0]].timing_map[NewTimingID].start_time
if len(str(stt % 60)) == 1:
    cell[1] = ':'.join([str(stt / 60), '0' + str(stt % 60)])
else:
```

```
    cell[1] = ':'.join([str(stt / 60), str(stt % 60)])

cell[2] = str(NewTimingID)
cell[3] = r[3]
cell[4] = r[4]
cell[5] = r[5]
cell[6] = cell[2]
cell[7] = r[7]

newrow = '\t'.join(cell)
my_ss_writer.write(newrow)
#  Update subarea_timing
st_file = open('initial/subarea_timing', 'rb')
find = False
for row1 in st_file:
    r1 = row1.split('\t')
    if r1[0] == str(s.id):
        r1[0] = str(NewTimingID)
        my_st_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[0] != str(s.id): break
# Update subarea_phasing
```

```
    sp_file = open('initial/subarea_phasing', 'rb')
    find = False
    for row1 in sp_file:
        r1 = row1.split('\t')
        if r1[1] == str(s.id):
            r1[1] = str(NewTimingID)
            my_sp_writer.write('\t'.join(r1))
            find = True
        r1 = row1.split('\t')
        if find and r1[1] != str(s.id): break

elif ts[0] >= 960 and ts[1] <= 1140:
    for s in self.node_map[r[0]].timing_map.values():
        if s.start_time == 900: break
    self.node_map[r[0]].timing_map[NewTimingID] = s
    self.node_map[r[0]].timing_map[NewTimingID].start_time = ts[0]
    self.node_map[r[0]].timing_map[NewTimingID].end_time = ts[1]
    cell = range(8)
    cell[0] = str(r[0])
    stt = self.node_map[r[0]].timing_map[NewTimingID].start_time
    if len(str(stt % 60)) == 1:
        cell[1] = ':'.join([str(stt / 60), '0' + str(stt % 60)])
    else:
```

```
    cell[1] = ':'.join([str(stt / 60), str(stt % 60)])

cell[2] = str(NewTimingID)
cell[3] = r[3]
cell[4] = r[4]
cell[5] = r[5]
cell[6] = cell[2]
cell[7] = r[7]

newrow = '\t'.join(cell)
my_ss_writer.write(newrow)
#  Update subarea_timing
find = False
st_file = open('initial/subarea_timing', 'rb')
for row1 in st_file:
    r1 = row1.split('\t')
    if r1[0] == str(s.id):
        r1[0] = str(NewTimingID)
        my_st_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[0] != str(s.id): break
# Update subarea_phasing
```

```
            sp_file = open('initial/subarea_phasing', 'rb')
            find = False
            for row1 in sp_file:
                r1 = row1.split('\t')
                if r1[1] == str(s.id):
                    r1[1] = str(NewTimingID)
                    my_sp_writer.write('\t'.join(r1))
                    find = True
                r1 = row1.split('\t')
                if find and r1[1] != str(s.id): break

        elif ts[0] >= 1140:
            for s in self.node_map[r[0]].timing_map.values():
                if s.start_time == 1140: break
            self.node_map[r[0]].timing_map[NewTimingID] = s
            self.node_map[r[0]].timing_map[NewTimingID].start_time = ts[0]
            self.node_map[r[0]].timing_map[NewTimingID].end_time = ts[1]
            cell = range(8)
            cell[0] = str(r[0])
            stt = self.node_map[r[0]].timing_map[NewTimingID].start_time
            if len(str(stt % 60)) == 1:
                cell[1] = ':'.join([str(stt / 60), '0' + str(stt % 60)])
            else:
```

```
    cell[1] = ':'.join([str(stt / 60), str(stt % 60)])

cell[2] = str(NewTimingID)
cell[3] = r[3]
cell[4] = r[4]
cell[5] = r[5]
cell[6] = cell[2]
cell[7] = r[7]

newrow = '\t'.join(cell)
my_ss_writer.write(newrow)
#  Update subarea_timing
st_file = open('initial/subarea_timing', 'rb')
find = False
for row1 in st_file:
    r1 = row1.split('\t')
    if r1[0] == str(s.id):
        r1[0] = str(NewTimingID)
        my_st_writer.write('\t'.join(r1))
        find = True
    r1 = row1.split('\t')
    if find and r1[0] != str(s.id): break
# Update subarea_phasing
```

```
            find = False
            sp_file = open('initial/subarea_phasing', 'rb')
            for row1 in sp_file:
                r1 = row1.split('\t')
                if r1[1] == str(s.id):
                    r1[1] = str(NewTimingID)
                    my_sp_writer.write('\t'.join(r1))
                    find = True
                r1 = row1.split('\t')
                if find and r1[1] != str(s.id): break


    else:
        my_ss_writer.write(row)  # Nothing changes for other nodes in
subarea_signal
        #  Update subarea_timing
        st_file = open('initial/subarea_timing', 'rb')
        find = False
        for row1 in st_file:
            r1 = row1.split('\t')
            if r1[0] == r[2]:
                my_st_writer.write(row1)
                find = True
            if find and r1[0] != r[2]: break
```

```
            # Update subarea_phasing
            sp_file = open('initial/subarea_phasing', 'rb')
            find = False
            for row1 in sp_file:
                r1 = row1.split('\t')
                if r1[1] == r[2]:
                    my_sp_writer.write(row1)
                    find = True
                if find and r1[1] != r[2]: break

    def creatNewCoordinator(self):
        new_sc_file = 'subarea_coordinator_new'
        my_sc_writer = open(new_sc_file, "wb")
        new_ss_file = open('subarea_signal_new', 'rb')
        first_row = True
        for row in new_ss_file:
            r = row.split('\t')
            if first_row:
                my_sc_writer.write('\t'.join(['ID', 'NOTES']))
                first_row = False
            else:
                my_sc_writer.write('\t'.join([r[2], 'IntControl Generated']))
```

```
def __update(self):  # Update the cycle length
    for node in self.node_map.values():
        for time in node.timing_map.values():
            new_cycle_time = 0
            for phase in time.phase_map.values():
                new_cycle_time += phase.green + phase.yellow + phase.red_clear
            time.cycle_time = new_cycle_time


def optimize_delay(self, time_slot):
    write1 = []
    write=[]
    mynodes=self.node_map.values()
    mynodes.sort()
    for node in mynodes:
        for time in node.timing_map.values():
            if time.start_time <= time_slot and time.end_time >= time_slot + 15:
                old_ave_delay = time.delay(time_slot)
                min_ave_delay = old_ave_delay
                optimal = time.phase_map.values()
                total_green = 0
                if min_ave_delay != 0:
                    for phase in time.phase_map.values():
                        total_green += phase.green
```

```
if len(time.phase_map.keys()) == 2:
    for g1 in range(8, total_green - 7):
        g2 = total_green - g1
        time.phase_map['1'].green = g1
        time.phase_map['2'].green = g2
        newdelay = time.delay(time_slot)
        print 'newdelay = ' + str(newdelay)
        print 'min_ave_delay = ' + str(min_ave_delay)
        if newdelay < min_ave_delay:
            min_ave_delay = newdelay
            optimal = time.phase_map.values()
if len(time.phase_map.keys()) == 3:
    for g1 in range(5, total_green - 9):
        for g2 in range(5, total_green - 9):
            for g3 in range(5, total_green - 9):
                if g1 + g2 + g3 == total_green:
                    time.phase_map['1'].green = g1
                    time.phase_map['2'].green = g2
                    time.phase_map['3'].green = g3
                    if time.delay(time_slot) <= min_ave_delay:
                        min_ave_delay = time.delay(time_slot)
                        optimal = time.phase_map.values()
if len(time.phase_map.keys()) == 4:
```

```
            for g1 in range(5, total_green - 20):
                for g2 in range(8, total_green - 17):
                    for g3 in range(5, total_green - 20):
                        for g4 in range(8, total_green - 17):
                            if g1 + g2 + g3 + g4 == total_green:
                                time.phase_map['1'].green = g1
                                time.phase_map['2'].green = g2
                                time.phase_map['3'].green = g3
                                time.phase_map['4'].green = g4
                                if time.delay(time_slot) < min_ave_delay:
                                    min_ave_delay = time.delay(time_slot)
                                    optimal = time.phase_map.values()
        # record
        ttss = ':'.join([str(time_slot / 60), str(time_slot % 60)]) + '-' + ':'.join(
            [str((time_slot + 15) / 60), str((time_slot + 15) % 60)])
        write1.append('\t'.join([str(node.id), ttss, str(old_ave_delay),
str(min_ave_delay)]))
        write1.append(chr(10))

        for phase in optimal:
            cell = range(11)
            cell[0] = time.id
            cell[1] = str(phase.id)
```

```
            cell[2] = str(phase.next)
            cell[3] = str(phase.green)
            cell[4] = str(phase.maxgreen)
            cell[5] = str(phase.extgreen)
            cell[6] = str(phase.yellow)
            cell[7] = str(phase.red_clear)
            cell[8] = str(phase.ring)
            cell[9] = str(phase.barrier)
            cell[10] = 'NODE ' + str(node.id)
            newrow = '\t'.join(cell)
            write.append(newrow)
        break
# report
if len(str(time_slot % 60)) == 1:
    cel = '-'.join([str(time_slot / 60), '0' + str(time_slot % 60)])
else:
    cel = '-'.join([str(time_slot / 60), str(time_slot % 60)])
filename = 'subarea_timing_new_' + cel
my_writer = open(filename, "wb")
st_file = open('initial/subarea_timing', 'rb')
for row in st_file:
    my_writer.write(row)
    break
```

```
        for newrows in write:
            my_writer.write(newrows)
            my_writer.write(chr(10))

        filename1 = 'Delay_record' + cel
        my_writer1 = open(filename1, "wb")
        my_writer1.write('\t'.join(['Node', 'Time_Slot', 'Before', 'After']))
        my_writer1.write(chr(10))
        for newrows in write1:
            my_writer1.write(newrows)

    def __timedelay(self, node, timing):
        totalgreen = 0
        totalvol = 0
        for phase in self.node_map[node].timing_map[timing].phase_map.values():
            totalgreen += phase.green
            totalvol += phase.vol
        Max_green = totalgreen * 2
        print Max_green
        Best = [0, 999999]
        for g in range(1, Max_green):
            r = g * 1.4
            ct = g + r
```

```
            delay = self.__delay_function(g, 3, ct, totalvol)['delay']
            if delay < Best[1]:
                Best[0] = g
                Best[1] = delay
            print g, delay, ct, totalvol
        for phase in self.node_map[node].timing_map[timing].phase_map.values():
            phase.green = float(phase.green /
self.node_map[node].timing_map[timing].cycle_time) * float(Best[0])

    def __delayRatio(self, node, timing):
        totalgreen = 0
        totalvol = 0
        totaldelay = 0
        for phase in self.node_map[node].timing_map[timing].phase_map.values():
            totalgreen += phase.green
            totalvol += float(phase.vol)
            totaldelay += phase.delay
        for phase in self.node_map[node].timing_map[timing].phase_map.values():
            phase.green = int(float(phase.vol / totalvol) * totalgreen)

    def optimize_delayRatio(self):
        for node in self.node_map.values():
            for time in node.timing_map.values():
```

```
            self.__delayRatio(node.id, time.id)
        self.__update()
        self.__delay()

    def report(self, node, time, phase):
        return [self.node_map[node].timing_map[time].phase_map[phase].delay,
            self.node_map[node].timing_map[time].phase_map[phase].check1,
            self.node_map[node].timing_map[time].phase_map[phase].check2,
            self.node_map[node].timing_map[time].phase_map[phase].check3,
            self.node_map[node].timing_map[time].phase_map[phase].status]



import time
#  import random
start = time.time()
import random

nodefile = open('initial/node', "rb")
nodes = []
for row in nodefile:
    nodes.append(row[:-2])
print str(len(nodes)) + ' nodes in the network to be optimized'
mynetwork = Network(nodes)
```

```
elapsed = (time.time() - start) / 60.0
print 'initial cost %s' % elapsed

for h in range(360, 600, 15) + range(900, 1140, 15):
# if True:
    mynetwork.optimize_delay(h)
    print h
    elapsed = (time.time() - start) / 60.0
    print 'time cost %s' % elapsed
#     h = 1125
#     mynetwork.optimize_delay(h)
#     print h
#     print 'time cost %s' % elapsed
# print 'ready'

# mynetwork.CreatNewTimingID()
# mynetwork.creatNewCoordinator()
# print 'new Timing has been assigned'
#
#
# mynetwork.optimize4()
# elapsed = time.time() - start
# print 'optimize4 cost %s' % elapsed
```

```
# mynetwork.optimize4()

# mynetwork.optimize4()

# elapsed = time.time() - start

# print 'total cost %s' % elapsed

# mynetwork.new_timing_file()
```