The Pennsylvania State University

The Graduate School

College of Engineering

OPTIMAL ADAPTIVE SIGNAL CONTROL FOR DIAMOND INTERCHANGES USING DYNAMIC PROGRAMMING

A Thesis in

Civil Engineering

by

Fang Fang

© 2004 Fang Fang

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2004

The thesis of Fang Fang was reviewed and approved* by the following:

Lily Elefteriadou
Associate Professor of Civil Engineering
Thesis Advisor
Chair of Committee

Konstadinos Goulias
Professor of Civil Engineering

Martin T. Pietrucha
Associate Professor of Civil Engineering

Natarajan Gautam
Associate Professor of Industry Engineering

Andrew Scanlon
Professor of Civil Engineering
Head of the Department of Civil and Environmental Engineering

* Signatures are on file in the Graduate School

ABSTRACT

The signalization of two closely spaced intersections in interchanges presents a major challenge in providing efficient traffic operations within the highway system. In current practice, PASSER III is the only existing signal optimization model for diamond interchanges. It optimizes the pre-timed signal plan based on off-line demand and cannot adapt itself to fluctuating demand situations. The two most popular adaptive signal control systems (i.e., OPAC and RHODES) have some limitations: OPAC cannot guarantee a globally optimum solution, both systems cannot be applied to optimize phase sequence, and the arrival patterns used for optimization horizons may not be reliable. Therefore, this research develops a methodology and a corresponding implementation algorithm using dynamic programming (DP) to provide optimal signal control of diamond interchanges in response to real-time traffic fluctuations. The problem is formulated as to find a phase sequencing decision with a phase duration that makes a pre-specified performance measure minimized over a finite horizon that rolls forward. The problem is solved by DP forward value iterations method. The optimization performance measure can be, for example, delay, queue length, number of stops, or any combination of these. A horizon of 10 seconds is divided into an integral number of intervals, each having 2.5 seconds. The optimal signal switches over each 2.5-second interval are found for each horizon. The optimization process proceeds one horizon after another and is based on the advanced vehicle information obtained from loop detectors set back a certain distance from the stop-line. A dynamic model of future vehicular detections, arrivals and departures is developed at the microscopic level in this study to estimate the traffic flows at the stop-line for each horizon.

The DP algorithm is coded in C++ language and dynamically linked to AIMSUN, a stochastic micro-simulation package, which is used for evaluation of the developed methodology. AIMSUN simulates a signalized diamond interchange instrumented with loop detectors that can provide vehicle counts and speeds to the DP algorithm. Based on this, the algorithm calculates the optimal phase sequence and the duration of each

horizon, and passes them back to AIMSUN, which subsequently controls the interchange in real time. To enable the algorithm to implement practical scenarios, a so-called majority rolling technique was also developed.

A sensitivity analysis using simulation results is conducted to study the characteristics of the DP algorithm. The results have shown that queue length and storage ratio defined performance measures are the best ones in minimizing system delays. A general rule of choosing the weight of an approach is that a larger weight applied for approaches having more demand. The study has also demonstrated the benefits of using dynamic weights without manually requiring the changing of weights. Dynamic weights can reduce system delay by 36 percent – 49 percent than fixed weights when the demand varies unpredictably every 15 minutes and is unbalanced. Moreover, the real-time DP algorithm has revealed the capability to accommodate various demand situations.

The real-time DP algorithm has also been compared to two off-line optimization packages: PASSER III and TRANSYT-7F. The optimized pre-timed signal plans from TRANSYT-7F and PASSER III are implemented in AIMSUN, and the results are compared to those from the DP algorithm. The simulation has exhibited that the real-time adaptive signal algorithm is superior to PASSER III and TRANSYT-7F in handling demand fluctuations for medium to high flow scenarios when the field demand is increased from the one used in off-line optimization. The performance of the three algorithms is almost identical if the simulation demand is the similar to off-demand situation and dose not vary much.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

First and foremost, I would like to express my sincere gratitude to my advisor and committee chair, Dr. Lily Elefteriadou, for her guidance, advice and encouragement throughout my study at the Pennsylvania State University in the past four years. Her many valuable comments and inputs at various stages of the research are the greatest help in the completion of this thesis.

I want to thank my committee members, Dr. Goulias, Dr. Pietrucha and Dr. Gautam for their constant support, help and insight into this research, as well as providing me with their expert guidance and suggestion in transportation engineering and optimization methods.

I would like to thank the NCHRP 3-60 project for the field diamond interchange data used in this research. I must also appreciate Mr. Jordi Cases for his technique assistance in using AIMSUN GETRAM Extension.

I am grateful to Dr. Elefteriadou, Dr. Goulias, Pennsylvania Transportation Institute, Mid-Atlantic University Transportation Center (MAUTC), Penn State Alumni Association Dissertation Award, Penn State College of Engineering for their financial support of this research.

Finally, I want to express my earnest thanks to my parents, my sister, Ling Fang and Michael Zekkos for their support and understanding.

CHAPTER 1
INTRODUCTION

1.1 Diamond Interchanges Operational Problems and Signal Plans

Signalized interchanges serve a critical function in highway systems. The most common interchange types are the diamond interchanges, which consist of two intersections, with one connection made for each freeway direction. These intersections are typically signalized when demand flows are high. The signalized intersections connecting to the arterial cross street are often the key operational element within the interchange system.

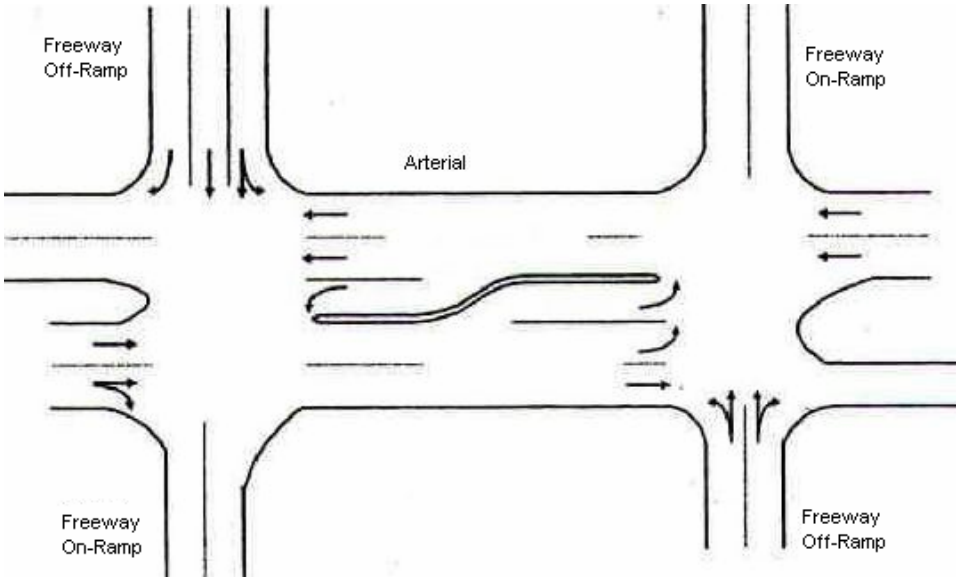


Figure 1.1 Geometric Layout of Diamond Interchange (Koonce et al, 1999)

The distance between the two intersections varies from less than 400 ft in densely developed urban areas to 800 ft or more in suburban areas (Messer et al, 1997). The close proximity of the two intersections creates a number of interactive effects that complicate the operation. This distance limits the storage available for queued vehicles. Thus if the signal timing is not properly set, the queue from the downstream intersection will spill back and block the upstream approaches. Another phenomenon that adversely affects

interchange operation is called demand starvation. It occurs when portions of the green at the downstream intersection are not used because conditions prevent vehicles at the upstream intersection from reaching the downstream stop-line (HCM, 2000). Additionally, diamond interchanges usually have predominant left-turn movements that adversely affect progression and platoon cohesion along the surface street. To mitigate these unique operational problems at diamond interchanges, it is essential to provide optimal signal control of these two closed-space signalized intersections for efficiently accommodating all movements involved.

The most common diamond interchange timing plans are three-phase or four-phase. Each of them has advantages and disadvantages in minimizing delay and queuing, but none of them is optimal for every possible diamond interchange geometry and traffic pattern. From the literature review undertaken, the only existing optimization model for diamond interchanges is PASSER III, developed by Texas Transportation Institute (TTI) (Messer, et al, 1977, Fambro, et al, 1991). PASSER III is a model designed specifically for diamond interchange signal optimization. The optimization principle of PASSER III is to search for a signal plan or a value of the parameter (such as cycle length, phase duration, offset, etc.) that results in the minimum delay from the restricted alternatives provided to the program. Every search focuses on one variable such as cycle length, phase duration, etc. Thus, the optimal solution provided by PASSER III may not be the globally optimum one. Moreover, PASSER III is an off-line, pre-timed signalized diamond interchange model and does not consider real-time demand fluctuations.

1.2 Optimal Adaptive Signal Control

Signal control in practice usually includes pre-timed, actuated and adaptive control. The pre-timed control operates on predetermined, fixed intervals and phase timings. The number and sequence of phases and the cycle length are also fixed. Thus when the traffic conditions change substantially, the timing plan will become less effective. More advanced than pre-timed, actuated control uses the gap-seek logic to respond to traffic fluctuations. The controller extends the green beyond a minimum time

until a gap in the traffic flow on the approaches currently with green has been detected or until the (pre-specified) maximum green time has been reached. Actuated signal control has certain limitations including the tendency to extend green inefficiently under low traffic flow conditions, and great sensitivity to incorrectly set maximum green times (Bell, 1990). Also, its effectiveness deteriorates rapidly as traffic demand increases. Actuated control works well only within certain ranges of demands. Both pre-timed and actuated control use pre-defined timing plans. The green time variation range (including minimum green, maximum green time, etc.) in actuated control is pre-defined. In contrast to these, adaptive control generates and implements the signal plan dynamically based upon real time traffic conditions, which are measured through a traffic detection system. The algorithm differs between the various adaptive control systems. Compared to pre-timed and actuated control, adaptive signal control has the potential to increase the operational efficiency of existing roadways, particularly for high demands or for varying traffic conditions.

Adaptive control can be defined as any signal control strategy that can adjust signal operations in response to fluctuating traffic demand in real time according to certain criteria (Lin and Vijayayumar, 1988). While the methods used to achieve such signal operations vary, adaptive control makes signal timing decisions based on detected or identified current and short-term or long-term future flow. The adaptive signal control concept was initiated by Miller (1963b). He described an algorithm for adjusting signal timings in small time intervals of 1~2 seconds. A decision to be made is whether to extend the current green duration or terminate it immediately. The algorithm calculates the difference in vehicle-seconds of delay between the gain made during an extension and the loss in the cross-street resulting from that extension.

Since Miller's pioneering work, considerable research has been done in the United Kingdom, Australia, and the United States to develop adaptive systems. Systems such as SCOOT (Hunt, et al, 1982), SCATS (Luk, 1984; Charles, 2001), OPAC (Gartner, 1983, Gartner and Pooran, 2001) and RHODES (Sen and Head, 1997, Mirchandani and Head,

2001) are among the best known. Some of them were tested and implemented in many cities worldwide. The adaptive control systems demonstrate the effectiveness of integrating Advanced Traffic Management Systems (ATMS) in Intelligent Transportation Systems (ITS) to reduce delay, reduce traffic accidents and improve the efficiency at intersections and networks. These systems include algorithms that provide signal timing plans in response to real-time traffic conditions. They require extensive surveillance, usually in the form of pavement loop detectors, and a communications infrastructure that allows for two-way communication with the central and/or local controllers.

1.3 Dynamic Programming (DP)

Among the other adaptive control systems listed above, OPAC and RHODES are two promising American systems that are based on dynamic programming and operate using the rolling horizon concept. They have been developed primarily for individual intersections and are being extended to networks. Both systems have their limitations. OPAC does not use a dynamic programming solution procedure; instead, a restricted optimal sequential constrained search (OSCO) is applied. Thus, the OPAC optimization methodology can be classified as trial-and-error enumeration that cannot guarantee a globally optimum solution. RHODES requires a fixed sequence of phases and a much longer projection time. Either one of these formulations cannot be applied to optimize phase sequence under phase constraints that are involved in the control for diamond interchanges. Nevertheless, both systems have demonstrated the potential of using DP to optimize the signal plan for a certain future horizon.

Earlier signal optimization algorithms considered a short future interval (i.e., 2 seconds), which cannot ensure an overall optimum solution and frequent computing of each interval is not appropriate for on-line implementation. More recent ones (i.e., OPAC and RHODES) take into account a longer future horizon (i.e., greater than 20 seconds). However, longer future time involves evaluating a large number of alternative signal timing sequences. One possible approach to solve this problem is to enumerate all possible phase sequencing paths. The number of possible alternatives is huge, and having

to calculate the performance measures for each path is not an appealing task and not appropriate for on-line implementation. Compared to the exhaustive enumeration methodology, DP is able to find a global optimal solution with much less effort and time, and thus can be implemented on-line. DP is a proven useful mathematical technique for selecting a sequence of interrelated decisions. It provides a systematic and efficient procedure for determining the optimal combination of decisions (Hillier and Lieberman, 2001).

In view of the above, DP should be a promising signal optimization technique for diamond interchange adaptive signal control. In this study DP is proposed to solve the phase sequencing and duration optimization problem at signalized diamond interchanges.

1.4 Objectives of the Thesis

This study aims at developing a methodology and the corresponding implementation algorithm to provide optimal traffic adaptive signal control of diamond interchanges using DP. To achieve this, two major objectives have been identified:

To develop a methodology for optimal solution of signal plan at diamond interchanges and the corresponding implementation algorithm.

The methodology will be based on DP because it provides great computational savings over exhaustive enumeration to find the best combination of phase sequencing decisions from a large number of alternatives. The proposed DP methodology in this study will be defined and formulated in a way that especially suits diamond interchange timing plans and will also lead to an efficient solution to the optimal signal plan. The framework and procedure to implement the proposed scheme will also be developed as part of this dissertation.

A vehicle arrival-discharge projection model will be built at the microscopic level for a diamond interchange. Mathematical expressions of various performance measures,

such as vehicle delay, queue length and number of stops will be developed. These models will be used in the DP algorithm and therefore influence the optimization performance.

To evaluate the proposed optimal signal control for diamond interchanges using micro-simulation.

One of the existing commercially available simulation models will be selected to simulate the proposed algorithm and the signal plans from other models with respect to various traffic scenarios and operational conditions. The model will be calibrated using field data. The proposed DP algorithm will be evaluated and compared to other signal optimization models using pre-selected performance measures.

The proposed DP algorithm will attempt to provide global optimal solutions to both phase sequence and phase duration, which would be a new contribution in the field of real-time signal control. The microscopic vehicle projection, arrival and discharge models to be developed will eliminate limitations and inadequacy of existing models. One of the most challenging tasks is the DP formulation of the problem. It must accommodate the optimization of both phase order and phase duration while taking into account the complexity of microscopic arrival-discharge dynamics in view of computational burden for real-time implementation. The evaluation and simulation of the proposed DP algorithm is another challenging task. It requires building an interface between the optimal signal plan from the algorithm and simulation model. To do so, the capabilities of individual simulation models and their programming implications must be carefully examined. The internal data structure of the simulation models has to be examined in order to select an appropriate model. A further challenging task is the modeling of the arrival trajectories and discharge process because they directly affect the optimization performance of the proposed DP algorithm.

The primary output of the dissertation will be a new and enhanced methodology for optimal signal control of diamond interchanges, including the decision network expression of the problem, the DP solution methodology, and the models for vehicle trajectories from detector arrival to discharging of the stop-line. The secondary output is the corresponding implementation procedure and algorithms.

1.5 Thesis Overview

A critical literature review on intersection traffic signal control and optimization, and diamond interchange signal control and optimization is included in Chapter 2. Chapter 3 presents the proposed methodology for optimal traffic adaptive signal control of diamond interchange using DP, and the model for vehicle arrival-discharge projection dynamics at the stop-line. This is followed by a discussion on the algorithm implementation in Chapter 4. The evaluation of the algorithm using simulation along with the selection of a micro-simulation model and the algorithm coding are given in Chapter 5. Chapter 6 conducts three types of sensitivity analysis to study the characteristics of the DP algorithm. The performance of dynamic weights and fixed weights is also discussed. Chapter 7 compares the performance of the DP algorithm with other two off-line signal optimization tools under various demand scenarios. Finally, Chapter 8 summarizes the contributions, findings and conclusions drawn from the research and also presents the future research.

CHAPTER 2
LITERATURE REVIEW

This chapter reviews previous research on traffic signal control and optimization for intersections and diamond interchanges in Section 2.1 and 2.2, respectively. In Section 2.1, various traffic signal control practices, delay models, and signal optimization methods are described. Optimization models for adaptive signal control are further discussed according to three categories – binary choice decision-making process, signal sequencing process by DP and parameters optimization model. Section 2.2 reviews the characteristics of common diamond interchange signal plans – three phase and four-phase, and existing diamond interchange signal optimization models. A summary of the literature review is presented in Section 2.3.

2.1 Intersection Traffic Signal Control and Optimization

2.1.1 Intersection Traffic Signal Control

Three are three basic forms of traffic signal control at intersection(s) in current practice: pre-timed, actuated and adaptive control.

The HCM (2000) provides the following definitions for the pre-timed and actuated controls:

a. Pre-timed control: Pre-timed controllers have a pre-set sequence of phase displayed in repetitive order. Each phase has a fixed green time and change interval that are repeated in each cycle to produce a constant cycle length.
b. Actuated control is divided three types: semi-actuated, fully actuated and adaptive:
   - Semi-actuated: Semi-actuated controllers operate with some approaches (typically on the minor street) having detectors. The earliest form of semi-actuated control was designed to maintain the green on the major street in the

absence of a minor-street actuation. Once actuated, the minor-street green is displayed for a period just long enough to accommodate the traffic demand.

- Fully actuated: Fully actuated controllers operate with timing on all approaches being influenced by vehicle detectors. Each phase is subject to a minimum and maximum green time, and some phases may be skipped if no demand is detected. The cycle length for fully actuated control will vary from cycle to cycle.
- Adaptive signal control (ITE Traffic Control Systems Handbook, 1985): Allows for on-line optimal signal timings based on prevailing demand. The system collects data on traffic flow conditions, determines a desired timing plan and then implements the timing plan in a time period such as each cycle, or every 5 minutes.

Pre-timed and semi-actuated controls are used primarily in the urban arterials where a network of signals must be coordinated. When properly calibrated, traffic actuated signals respond to the traffic demand in certain ways, therefore, providing considerable advantages over fixed-time equipment and being widely used for isolated intersections. The gap-seeking logic is usually applied in the actuated control. The logic attempts to terminate a current green duration when the gaps between arriving vehicles on the approaches currently with green are longer than the prescribed value or a maximum green time is reached.

2.1.2 Performance Measures – Delay Models

Generally, analytic models available for optimizing signal control are based on minimizing the vehicle delay for the fixed-time signal control. One of the earliest attempts to study delays at fixed-time signals was made by Winsten (1956). He used a model with fixed interval of vehicle departure, but with a binominal distribution of arrivals. This model predicts delays fairly well when the traffic is low. However, when the traffic is high, the delay is substantially underestimated.

Webster (1958) developed a better delay model by using Poisson arrivals. Webster's model is based on the assumption that the intersection is operating under capacity, i.e., no queues are built continuously.

Miller (1963a) derived a more general delay formula based on the model proposed by Winsten (1956). In Miller's model, the arrival patterns are accounted for variance-to-mean ratio, while the departure pattern is still assumed to be regular. In terms of numerical calculations, Miller's model is nearly the same as Webster's model. A number of field sites in Birmingham were studied using Miller's model and Webster's model. When arrivals are nearly random, both models predict almost the same delay that is very close to field-observed delays. The study also showed that when the arrival patterns with high variance-to-mean ratio, the Miller's model gives more accurate prediction than the Webster' model.

Dunne and Potts (1964) developed time-varying control algorithms for an undersaturated intersection with constant arrivals and departure rates which guarantee that for any initial state the system will eventually achieve a limit cycle that minimizes the average delay. This algorithm is represented by computed control functions that are related to the number of vehicles queued and demanding service at the intersection. This control is more similar to actuated control since it can react to the demand dynamics.

The optimal control models described above make use of either the expected flow rate, or a statistical distribution such as Poisson, binomial distribution while no time-variant features of traffic arrivals are taken into account. The traffic arrival patterns and departure patterns are assumed and none of these controls considers the real-time traffic flow process.

2.1.3 Optimization Methods for Signalized Intersection Operation

2.1.3.1 Optimal Phase Splits Using the Time Budget Concept

A traffic signal must provide optimum right-of-way availability to all movements at the intersection. The "time budget" concept considers the maximum sum of critical lane volumes (or capacity) that a signal can accommodate, given that the prevailing saturation headway and lost times are known. The following equation (Roess, McShane and Prassas, 1998) presents this principle,

$$V_c = \frac{T_G}{h} = \frac{1}{h}\left[3600 - (N)(t_L)(\frac{3600}{C})\right] \quad (2.1)$$

Where
V_c = the maximum sum of critical lane volume, or capacity
C = cycle length
N = number of phases
t_L = total (start-up plus clearance) lost time per phase
h = saturation headway

Critical lane is defined as the one with the highest flow ratio (demand flow rate /saturation flow rate) for a given signal phase (HCM, 2000). Each signal phase has one and only one critical lane that determines the green-time requirements for the phase. The time budget allocates 3600 seconds/hour to each of the signal phases based on the critical lane and its flow, and on lost times. The maximum sum of critical lane volumes is calculated by firstly determining how much total lost time in one hour, and then the time remaining in the hour to allocate to effective green time for the critical movement, denoted as T_G, finally dividing this remaining time by the saturation headway. The relationship between cycle length and capacity is provided from the above equation. It shows that as cycle length increases, capacity increases since there are fewer cycles and thus less lost time within the hour.

Equation (2.1) calculates the maximum sum of critical lane volumes that the signal plan can accommodate within the hour. The total amount of effective green times in the cycle is obtained based on the desired cycle length. Finally, the optimal phase splits can be determined in terms of effective green time allocation. The effective green times

for each signal phase can be allocated in proportion to the critical lane volumes in the corresponding phases.

The time budget concept allocates the effective green time based on the sum of critical lane volume that a signalized intersection can accommodate in one hour. It does not take into account the variation of traffic demand in real time. Green time will not be fully utilized when the traffic volume is lower than the capacity. Moreover, the time budge concept does not address the delay estimation. Equation (2.1) shows cycle length increases capacity increases. However, as cycle length increases, average delay increases. The results from the time budge concept cannot guarantee the optimal signal plan in terms of capacity (v/c) and delay. Both capacity (v/c) and delay are needed to evaluate operations at a signalized intersection. Finally, the time budget concept doesn't consider the impact of right- and left-turning vehicles. The signal plan provided by the time budget concept should thus be fully analyzed and fine-tuned using other models.

2.1.3.2 Optimization Methods for Adaptive Signal Control

Adaptive signal control uses algorithms that perform real-time optimization of traffic signals based on the current traffic conditions, demand, and system capacity (Hicks and Carter, 2001). Adaptive control attempts to optimize signal operations and to implement optimal signal-switching sequences based on short-term or long-term future flow provided by surveillance.

Some adaptive signal optimization models intend to optimize parameters such as cycle time, splits and offsets while others move away from these parametric models. The non-parametric models rely on the signal timing decision-making process to optimize the operation. According to their decision-making logic, these non-parametric optimization strategies can be classified into two categories. One category is the binary choice decision-making process to adjust signal timings while the other uses a signal sequencing process. The characteristics of each type are reviewed in the following sections.

- Binary Choice Process

The binary choice process studies a look-ahead horizon for a small interval of up to two seconds each. In each interval, the optimization process compares the benefits of extending the green by another interval against the loss of terminating it. Usually an objective function of vehicle-seconds of delay is selected to represent the difference between the gain made by the extra vehicles that can be discharged during the green extension and the loss to the queuing vehicles in the cross direction resulting from that extension.

Miller (1963b) initiated the adaptive signal control concept and proposed an algorithm for adjusting signal timings in small time intervals. A decision to be made is whether to extend the current green duration or terminate it immediately. The decision is repeatedly made at a very short fixed interval (of 1~2 sec) by the examination of a delay-based control function. This function provides a trade-off analysis that weights the benefits of extending the green by another interval against the drawbacks of terminating it. The study used a four-way intersection to describe the algorithm. Miller's algorithm doesn't consider more complicated cases, such as left-turn protected phase, right turn phase, etc.

Following Miller's original principle, several binary choice adaptive systems have been developed, such as TOL - the Traffic Optimization Logic (Bang, 1976), the control strategy by Michalopoulos and Stephanopoulos (1979), and MOVA - the Modernized Optimized Vehicle Actuation Strategy (Vincent and Young, 1986; Kronborg, P., Davidson, F., 1993).

The Swedish TOL method was developed by Bang (1976). The method is a further study of Miller's model. In the TOL, the extension of the green light is based on calculations of a control function at regular intervals of h seconds. This function represents the gain or loss in community cost resulting from extension of the prevailing green light with h seconds. The objective is to minimize the sum of travel time and vehicle operating costs as well as environmental costs caused by the traffic. Like Miller's

algorithm, TOL does not guarantee that an overall optimal control is obtained, since it is based on a short-term optimization.

Michalopoulos and Stephanopoulos (1979) developed a control strategy for critical intersections (defined as ones that saturate frequently) that can be implemented on a real-time basis since the only on-line information that should be provided to the controller is the demand estimation one cycle ahead of time. The objectives for control policy are to minimize total intersection delay during one cycle ahead of time and to maintain the queue lengths at reasonable levels so that upstream intersections are not blocked. The control problem under consideration can be described mathematically by inspection of the cumulative input and output curves for each approach to the intersection. However, accurate estimations of the queue length are hardly obtained by the simple input-output analysis. A realistic model describing queue dynamics at signalized intersections is needed for further research.

The British MOVA system was developed by the Transport and Road Research Laboratory (TRRL) between 1982 and1988. The control strategy of MOVA consists of a mathematical optimization algorithm and a heuristic algorithm. There are two detectors used – one at 40 m and one at 100 upstream. Firstly the vehicles queued up to the first upstream detector (40 m) are discharged by providing adequate green time. Then the headways at which vehicles cross the detector are examined to determine whether there is a reduction in saturation flow rate. Finally, the algorithm is applied to determine extending the current green or terminating it. The calculation is made in every half-second to determine whether the total delay will be minimized if the current phase is allowed to continue to be green during 0.5 sec., 1.0 sec., 1.5 sec.… up to a preprogrammed time or if the phase is directly forced to terminate. MOVA is also able to handle oversaturation scenarios. MOVA switches from the normal delay-and-stops minimizing process to a capacity-maximizing routine to clear the congested approaches. The MOVA optimization process is also based on Miller's algorithm.

All the above binary choice process models make the decision based on a very short future horizon and corresponding optimization time interval for optimizing signal operations and thus, this method does not ensure overall optimality of the control strategy over a long period of time.

- Signal Sequencing Process Using DP

Signal sequencing process in the real-time traffic adaptive control problem is to determine the optimal signal timing sequence for a relatively long future period of time. Dynamic programming (DP) is a mathematical technique for making a sequence of decisions to determine the optimal combination (Hillier and Lieberman, 2001). It has found several applications in signal sequencing process as follows.

Roberston and Bretherton (1974) first applied DP in the optimization of signal control. The study presented the so-called DYPIC (Dynamic Programmed Intersection Control) algorithm. It finds the optimal signal control policy for a finite look-ahead horizon and its objective is the minimization of the total delay aggregated over all intervals of a finite horizon. The algorithm was designed for an intersection of two identical, single-lane, one-way roads. In the DP model, the time is divided into short intervals of 5 seconds; the arrival of future traffic at the end of each queue, or at the stop-line is known; the maximum rate of discharge is 2 vehicles every 5 second interval (i.e., the rate of discharge of 1440 veh/h); the queue length are calculated at the end of each interval. The DP for optimal signal control policy was solved by backward dynamic programming (BDP) and the optimal policy was defined as a set of signal change times over a period of 10 minutes.

Roberston and Bretherton (1974) also employed DYPIC to examine the sensitivity of the optimal policy to changes in the traffic situation. They attempted to determine what extent optimal decisions are affected by future arrival rates. It was found that in most cases, it is only vehicle arrivals in the immediate future that affect the control decision, and a near optimal signal change decision could be taken on the basis of the

vehicles expected to arrive in the next 10 to 20 seconds without being affected by later vehicle arrivals.

The DYPIC algorithm deals only with a two- phase intersection, with two one-way approaches. The algorithm needs an exact knowledge of vehicle arrival times for several minutes into the future and places substantial demands on detection, computer power and data storage.

As an extension of DYPIC, OPAC (Optimization Policies for Adaptive Control) by Gartner et al (1983, 2001) identifies the optimal signal timing sequence for a relatively long future period of time that is defined as a projection horizon. The projection horizon is usually in the range of 50-100 seconds and is divided into an integral number of intervals, typically 2-5 seconds. Signal timings are calculated to directly minimize performance measures such as vehicle delays and stops. Although the OPAC algorithm has a dynamic programming formulation, due to limited computational capabilities in the early 1980s, in an effort to adopt a practical solution method for the online application, Gartner applied a restricted optimal sequential constrained search (OSCO) instead of the DP solution method. The OSCO evaluates total delay sequentially for restricted feasible switching sequences. At each iteration, the current objective function is compared with the previously stored value and, if lower, replaces it. Thus, although OPAC defines its problem as DP, this optimization methodology can be classified as trial-and-error and try enumeration.

In the OPAC algorithm, a combination of measured and modeled demand is employed to determine optimal phase switching sequences at the signal where the phase durations are constrained by minimum and maximum green times. To reduce the requirements on future arrival information for the entire optimization period, which in practice is difficult to obtain with reliability, OPAC implementes a so-called "rolling horizon" concept, as shown in Figure 2.1. From upstream detectors, actual arrival data are obtained for r intervals at the "head" of the projection horizon. For the next $(k-r)$

intervals, the "tail" part, flow data are obtained from a simple model. The model consists of a moving average of all previous arrivals on the approach. An optimal policy is calculated for the entire horizon but implemented only for the head section. The projection horizon is then shifted (rolled) *r* units ahead, new flow data are obtained for the stage (head and tail), and the process is repeated. In this way, the algorithm can dynamically revise the switching decisions as more recent real-time data continuously become available. However, since a great portion of arrival information in the projection horizon is modeled as the average flow rate, whether the optimal solution is responsive to the real demand needs to be further investigated.

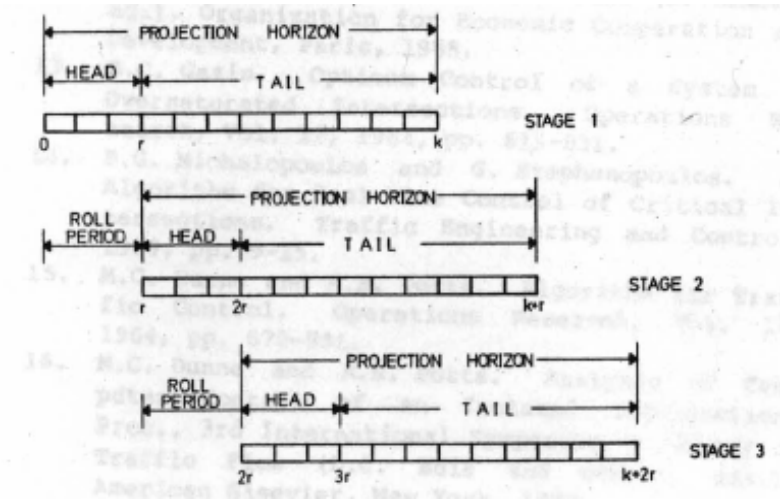


Figure 2.1 Rolling Horizon Concept (Gartner, 1983)

Furthermore, OPAC uses the simple arrival projection and queuing-discharge process to calculate the total delay of each approach for the optimization period, which is the sum over all intervals of the initial queue length plus the arrivals minus the departure during each interval. This functional relation implies that the vehicles detected in an interval can be transformed into arrivals at stopline after *t* seconds if average travel time between the stop line and the detector is *t* seconds. However, this travel distance changes when initial queues for an interval exist and the queues length varies. Thus the number of vehicles arriving and stopping at the stopline during an interval is not equal to the number of vehicles detected upstream. The arrival projection at the stopline is a very

complicated process, and the models adopted by OPAC are not suitable for diamond interchanges which often experience serious queuing problems.

RHODES, Real-time Hierarchical Optimized Distributed Effective System, developed at University of Arizona (Sen and Head, 1997; Mirchandani and Head, 2001), is also a DP-based real-time signal control system. Different from OPAC specifications of DP formulation, RHODES uses signal phases as stages, the amount of green-time as decision variables, and the total number of time-steps as state variables. This DP formulation requires a fixed sequence of phases and a much longer projection time. Other downsides are: unbounded decision variables, stage-dependent (varying) decision space, unmanageable number of stages over a prescribed optimization period. Due to their DP formulations, both OPAC and RHODES could not optimize phase sequence under phase constraints that are involved in the signal control for diamond interchanges.

- Parameters Optimization Models

There are two notable real-time systems deployed based on the optimization of parameters (e.g., cycle time, splits and offsets). They are SCOOT - Split, Cycle and Offset Optimization Technique developed by Transportation and Road Research Laboratory (TRRL), UK (Hunt, et al, 1982), and SCATS – Sydney Coordinated Adaptive Traffic System developed by New South Wales Roads and Traffic Authority (RTA) (Luk, 1984; Charles, 2001). Both systems attempt to select the best parameters to maintain a pre-defined degree of saturation for the critical intersection in the network.

Based on predictions using data from detectors upstream, the SCOOT program firstly establishes and models the development of queues of vehicles at the stop-line and so adjusts phase lengths and offsets to minimize delay. The split optimiser selects the phase that will be closest to achieving a 90 percent degree of saturation without changing the phase starting time by more than 4 seconds. The offset optimiser changes the offset of an intersection from its adjacent intersection once every cycle. The demand profiles obtained upstream and downstream of each intersection are used to assess whether a change in offset would minimize the overall disutility of delay for that cycle. The cycle

length optimiser tries to maintain the degree of saturation for the heaviest intersection at 90 percent. A change in cycle length can only be made in every 5 minute. The optimization model used by SCOOT is the hill-climbing procedure used in TRANSYT-7F, which is a network signal timing optimization program developed by TRRL. The hill-climbing procedure makes changes to signal timings (such as splits and offsets) and determines whether or not the performance index (a combination of stops and delay, fuel consumption or operating cost) is improved. The hill-climbing is an iterative, gradient search technique. Firstly an initial signal timing plan and the initial performance index is given. Then an optimized parameter (i.e., offset) is increased by one pre-specified "step size". The new performance index is compared with the previous value: if the new performance index is less than the previous value, the program continues to increase the offset by the same amount, as long as the performance index continues to decrease; if the new performance index is greater than the previous value, the program will decrease the offset by the same amount and continue to decrease the offset by this amount as long as the performance index continues to decrease. The above procedure will be repeated for all pre-specified optimzation "step size". The disadvantage of hill-climbing procedure is that it still requires extensive numerical computations, and it provides no guarantee that the global solution will always be found.

SCATS uses information from vehicle detectors located in each lane ahead of the stop bar to adjust signal timing in response to variations in traffic demand and system capacity. It determines suitable signal timing based on average traffic conditions. It anticipates the arrival of vehicles at a certain intersection from traffic data collected at the preceding intersection and responds accordingly. Flow and occupancy are calculated during the green phase on each intersection approach. This information is transmitted to a control center where SCATS attempts to maintain a user-specified degree of saturation on the intersection downstream of the collected traffic data. Signal phases can be set to the equalize degree of saturation on all approaches or they can be arranged to give priority to a particular direction (Pearce, 2001).

Compared to the parameters optimization models, signal timing sequencing decision-making logic models that make the signal decision to switch between phases are based on actual arrival data at the intersection, thus making them more relevant for real-time traffic-adaptive signal control.

2.2 Diamond Interchange Signal Control and Optimization

2.2.1 Diamond Interchange Signal Control

The basic feature of a diamond interchange is the two closely spaced intersections that connect the freeway entrance and exit ramps to the arterial. Signalization of these two intersections is necessary when demand flows are sufficiently high. The signalization of a diamond interchange presents several major challenges. Most of these involve the heavy turning movements and the possibility of spillback queuing of vehicles between the internal link connecting the two intersections. Figure 2.2 illustrates the interchange traffic movements and phasing number convention established by NEMA (Bonnesson and Lee, 2000).

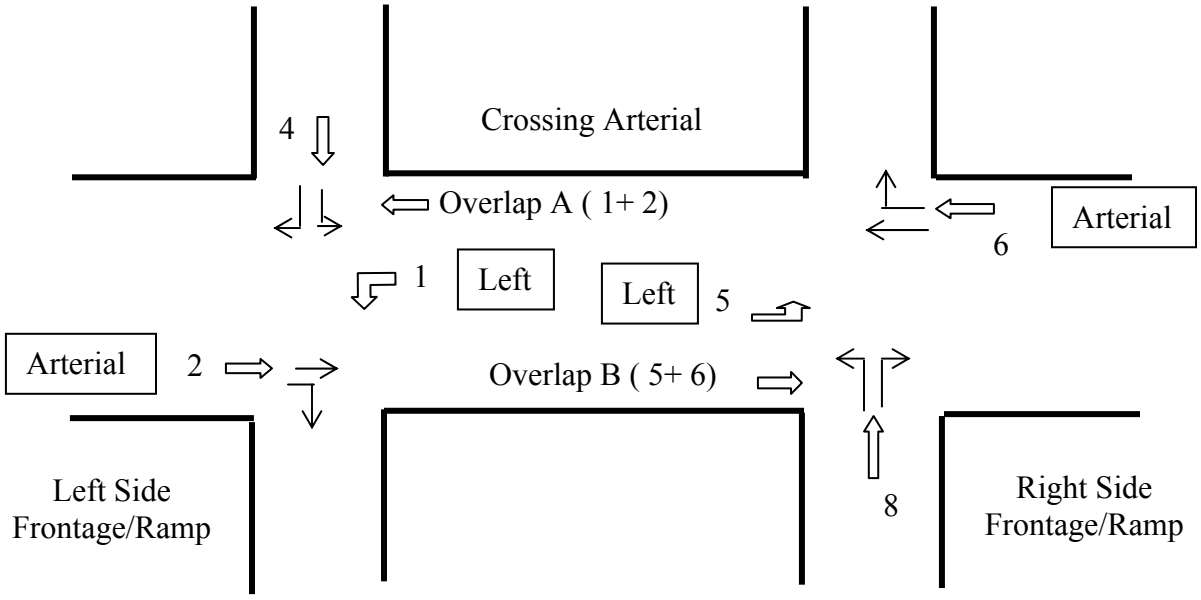


Figure 2.2 NEMA Phasing for Diamond Interchange Movements (Bonnesson and Lee, 2000)

Various phasing sequences have been developed to improve the traffic operation of interchanges. None of these plans, however, can provide the "optimal" or "best" strategy for every geometry and traffic condition. The three-phase and the four-phase scheme are the most often used at diamond interchange; and are briefly described below:

- Three-phase Plan

The three- phase plan treats the diamond interchange as two separate intersections, with each having three phases. One phase is for through movements on the surface arterial, the second for through and left movements from the internal links, and the third for ramp movements. As shown in Figure 2.3, each ramp junction is uniquely controlled by the phase timing functions in one ring. The plan can have 5 possible signal phase pairs 26, 25, 16, 15 and 48 while it requires the simultaneous termination of arterial phases before switching to the ramp phases.

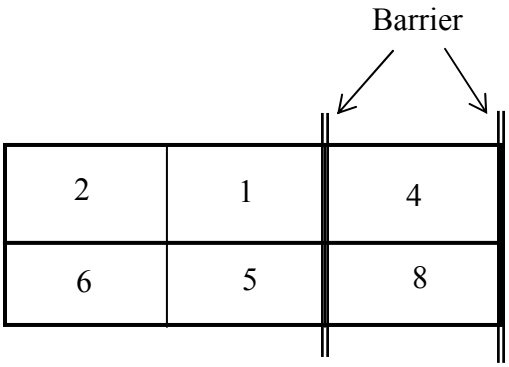
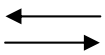
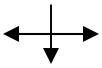
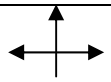
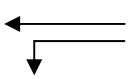
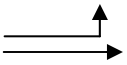
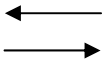
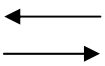
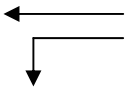
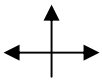
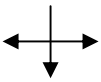
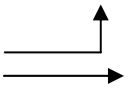
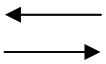
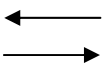
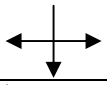
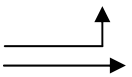
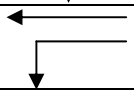
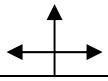
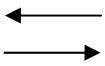
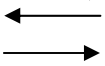
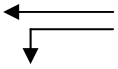
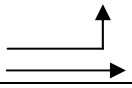
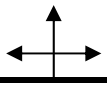


Figure 2.3 Basic Three-Phase Diamond Interchange Ring Structure (Bonneson and Lee, 2000)

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

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

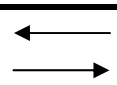
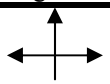
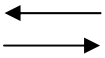
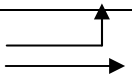
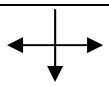
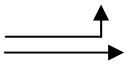
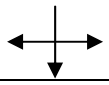
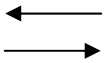
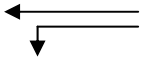
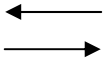
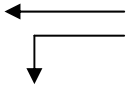
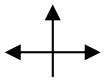
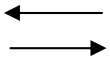
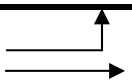
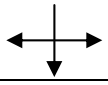
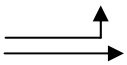
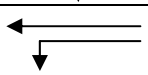
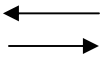
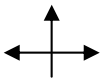
Phase Plan		Phase Number	Left Side	Right Side
3-Phase	1 (Lead –Lead)	Phase 1		
		Phase 2		
		Phase 3		
	2 (Lag-Lead)	Phase 1		
		Phase 2		
		Phase 3		
	3 (Lead – Lag)	Phase 1		
		Phase 2		
		Phase 3		
	4 (Lag- Lag)	Phase 1		
		Phase 2		
		Phase 3		

- Four-phase Plan

Table 2.2 presents the most common four-phase pre-timed diamond interchange plans. In the four-phase with overlaps plan, vehicles entering and exiting from the internal link can happen at the same phase except that at overlaps Phase 1 and 4, through vehicles are allowed to enter the internal link while not be able to exit the link during the phase. The four-phase with overlaps strategy can possibly reduce queuing in the internal link of the interchange. However, because the plan requires four phases, longer cycle

lengths are required which reduce the capacity of the intersection. The four-phase plan decreases the effective green time per cycle (g/C) ratio for the major movements and increases delay.

Table 2.2 Common Four-Phase Signal Plan (HCM, 2000)

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

2.2.2 Performance Measures of Diamond Interchanges

The following lists some performance measures that have been considered in the literature (Elefteriadou et al, 2002):

- ***Average Control Delay Per Vehicle***: The average control delay per vehicle is incurred by all vehicles traversing the interchange. Average control delay is estimated for each

movement, and the overall interchange control delay is estimated as the weighted-by-demand average of all movements.

-***Combined Average Interchange Travel Time***: The combined interchange travel time was defined as the sum of the control delays experienced at each signal traversed, plus the link travel time between the upstream and downstream stop line (Roess, 1999). The combined average interchange travel time is the weighted average of all individual movement values, where weighting is by movement demand.

-Likelihood of Queue Storage Failure: This measure is defined as the probability that the vehicle queue will exceed the available storage. Various probability levels could be used, with a reasonable range being from 85 percent - to 95 percent.

-Demand/Capacity Ratio (V/C ratio): The v/c ratio is a measure of the degree of saturation. It is a performance measure used in the analysis of signalized intersections, and it can evaluate the utilization of the effective green for a lane group, for a given signal timing plan.

- Number of Vehicles Discharged: This refers to the number of vehicles exiting the system during the control period, and is a very common performance measure used in microscopic simulation models. It can provide a measure of capacity, but cannot by itself describe oversaturated conditions.

None of the existing performance measures by itself can fulfill all the requirements for the evaluation of diamond interchange.

2.2.3 Optimization of Diamond Interchanges Signal Operations

One of the early works on the optimization of signalized diamond interchanges was conducted by Messer et al (Messer, et al, 1977). They developed PASSER III (Messer, et al, 1977, Fambro, et al, 1991) to identify the best strategy for a pre-timed signalization diamond interchange. Interchange performance is evaluated by using average vehicle delay based on a combination of Webster's equation for external delay and a delay-offset relationship developed by Gartner (1973) for internal delay. The principle of PASSER III is to evaluate all possible alternatives and output the results for

the signal plan that minimizes the average delay per vehicle. The delay is calculated by Webster's delay equation. PASSER III can choose from several interchange phasing patterns including combinations of "leading" and "lagging" green of three-phase plan and four-phase with overlaps. The optimization technique used by PASSER III is to search for the option with the minimum delay from the restricted alternatives provided to the program. PASSER III requires as input the traffic volume for each approach Poisson arrivals are assumed for the external link. The program is not responsive to real-time demand, but could be used for off-line optimization.

As stated early, two closely-spaced traffic signals with heavy turning movements and high levels of demand create a complex signal control scenario. Although existing control strategies that use off-line optimization programs can be changed by time of day as daily traffic conditions change, they are not sensitive to day-to-day fluctuations in demand or to unexpected changes.

2.3 Summary of Literature Review

Several signal timing strategies have been developed for diamond interchanges, but none works optimally for every set of traffic and geometric conditions. PASSER III is the only existing signal timing optimization model for diamond interchanges, but it provides the optimal solution among a restricted number of alternatives and may not provide the globally optimum solution. Moreover, PASSER III determines the signal plan based on off-line demand and is not responsive to real time demand fluctuation. Hence, there is a need to develop an optimal strategy for adaptive control of diamond interchange.

The literature review on the state of the art of optimal signalized intersection(s) has shown that there are two adaptive signal optimization logics that is not based on the optimization of parameters (e.g., cycle time, splits and offsets). They are the binary choice decision-making process and the signal timing sequencing process. The binary choice logic looks at only a very short future time interval (i.e., 2 second) for carrying out

the optimization, thus it is unlikely to ensure the overall optimum solution. Moreover, the frequent computing (every interval, i.e., 2 seconds) is not appropriate for on-line implementation. The sequencing approach considers a much longer future period for signal optimization, and a much larger number of alternative signal timing sequences are evaluated in the optimization process. The sequencing process can achieve better signal operation than the binary choice. Thus, this dissertation will consider a longer future period). DP is selected to carry out this optimization process because it provides great computational savings over using exhaustive enumeration to find the best combination of decisions, especially for large problems with a huge number of alternatives. Past research has shown that DP is a promising optimization technique for adaptive signal control problems.

By reviewing the past research on the application of DP in signal optimization, OPAC appears to be a more practical method. The OPAC model (Gartner, 1983 and 2002) is a discrete-time optimal control formulation for a single intersection. However, the optimization problem is not solved by a real DP solution procedure, but instead, uses a restricted search heuristic that cannot guarantee global optimization. The global optimality represents the real optimal results over the entire horizon. Unlike OPAC, the optimization solution in this dissertation will be performed by a DP procedure. The goal of this study is to develop a computationally efficient algorithm that can be implemented in real time.

Phase optimization has not been examined in both OPAC and RHODE. The optimization method used in this study will allow phase sequencing optimization without which the impact of real-time optimization may be very limited.

The vehicle arrival projection and queue discharge functions were oversimplified in previous algorithms (including OPAC and RHODES). It was assumed that the traffic information at stop line could be predicted in advance for the vehicle travel time from the detector to the stop line. However, the queues present at stop-line complicate this

projection relation, because the vehicles detected within an interval cannot simply be projected into the corresponding future interval at the stopline. Microscopic arrival-discharge models that provide more reliable vehicle information for the optimization algorithm will be examined, modified as appropriate, and used in this study.

The arrival information plays a significant role in adaptive signal control since the system should be responsive to the real demand. The surveillance, usually loop detectors, is relied on to provide information on arrivals. But it is not practical to place detectors very far upstream to obtain the desired amount of information due to the geometry limits of diamond interchanges. For example, given an arrival speed of 30 mph, to obtain arrivals information 25 seconds in advance, the detector should be set 1100 ft upstream. However, diamond interchanges internal links sometimes are not long enough to meet this requirement. Conventional diamond interchanges have spacing of 800 ft or more (Roess, 1999). Thus, associated with upstream detector data, the sensitivity of the entire performance of optimal adaptive signal control to the arrival prediction model needs to be further investigated.

Whereas DP has been applied for signal control optimization, different DP formulations result in algorithms that require different computational loads. For the signal control problem, it is essential to recognize whether the formulated DP procedure is a good representation of the real problem, i.e., how the problem can be solved by DP efficiently and effectively. Alternative ways to define stage, state and recursive relationships leads to alternative DP methodologies. This study will pursue a DP formulation that especially suits diamond interchanges and also leads to an efficient solution to the to the optimal signal plan.

CHAPTER 3
OPTIMIZATION METHODOLOGY

In this chapter, the methodology proposed for providing the optimal solution for signal adaptive control at diamond interchanges is described. Firstly, the adaptive signal control of diamond interchanges is formulated as a decision network problem in Section 3.1. This representation of the problem suggests that dynamic programming (DP) is an appropriate solution. The DP formulation is then presented in Section 3.2. Its major notations such as stage, state variables, decision variables, and immediate return functions are defined in the context of the signal optimization problem. For the solution of the optimal decision network, the forward recurrence relation, signal state transition equation, immediate return equation and optimal value function are explicitly expressed. Next, section 3.3 develops a model to study vehicles arrival projection and discharge process under different signal decision. The model is needed in the DP optimization to determine return function value. The consideration of queued vehicles at the stop-line, vehicle travel time and vehicle projection dynamics as well as detection ranges are addressed in each subsection. Finally, several performance measures including queue length, storage ratio and delay are defined in Section 3.4. These performance measures will serve as the optimization criteria in the DP method.

3.1 Decision Network Formulation

3.1.1 Adaptive Signal Plan for Diamond Interchanges

As shown in Figures 2.2 and 2.3, a basic NEMA three-phase ring structure signal plan diamond interchange, has a total of 5 feasible phases, i.e., phases 26, 25, 15, 16 and 48, which are subject to the ring structure constraints. An adaptive signal plan is a sequence of signal phases of varying durations. Using this basic three-phase signal plan as an example, such an adaptive signal plan is shown in Figure 3.1. The time can be discretized in a constant time stage (i.e. interval) Δt. The duration of a phase (i.e., P_{26},

P_{25}, P_{15}, P_{16}, P_{48}) is, therefore, the sum of successive stages Δt. The example starts with a given initial phase 26,

26 → 25 → 15 → 48 → 26 → 15 → 48 → ---

in which the first cycle of the phase sequence has phases 26, 25, 15 and 48 and the second cycle has phases 26, 15 and 48. In this example the phase sequence is not fixed, some of the phases can be skipped, and the lengths of individual phases may vary. Referring to the NEMA ring structure (Figure 2.3), there are three possible phases after phase 26: phase 25, phase 16 or phase 15. Thus an adaptive signal plan will operate as the phase extensions or switches at the boundary between stages (i.e. discrete time t1, t2, t3...) only. The decision whether to extend the current phase or switch to one of the other permissive phases must be made in each t.

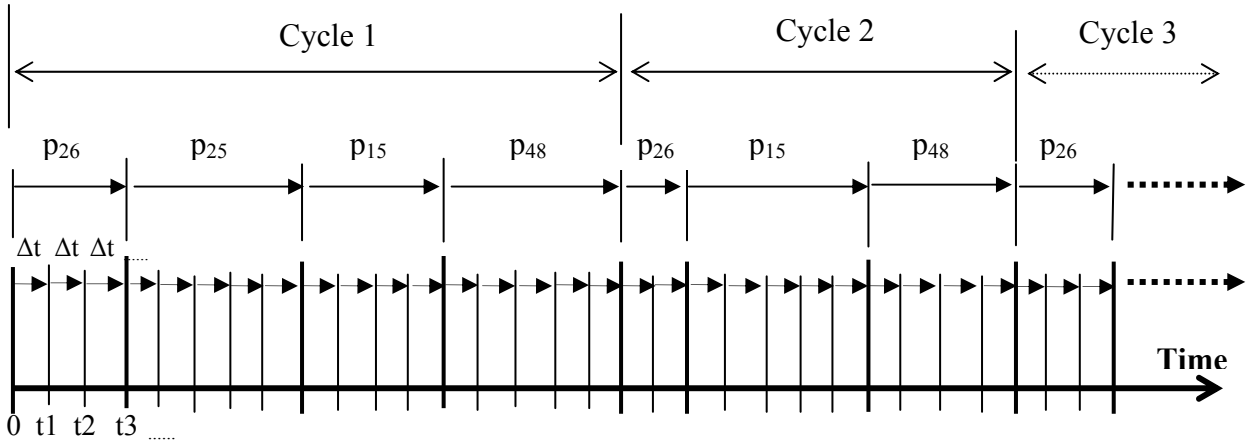


Figure 3.1 Example of an Adaptive Signal Plan

3.1.2 Decision Network Formulation of Diamond Interchanges Signal Control

Adaptive signal optimization problem for a diamond interchange can be represented by a decision network (Figure 3.2). A decision network is a network of interconnected nodes that are arranged in layers. Departing from one node (i.e., phase) are lines representing possible decisions and pointing toward the next phases. A node with a label in it is the signal phase status. Time is discretized at a constant stage Δt. At

each discrete time (t1, t2, t3....) there are five possible signal phases, i.e., phases 26, 25, 15, 16 and 48. Thus a layer has a total of 5 nodes representing possible signal phases at a time. Single input to a node is the decision coming from one of the previous possible nodes, and multiple outputs departing from a node result in the signal status at next discrete time. Figure 3.2 shows an example decision network with an initial phase 26. In the context of a decision network, a signal plan is a decision trajectory that is composed of one node in each layer. A sequence of nodes are linked from the left most layer to the infinite.

With respect to an individual movement a decision network can have two possible signal states (green and red) in a layer. Figure 3.3 presents an example of signal plan and the signal status change for each movement at a diamond interchange.

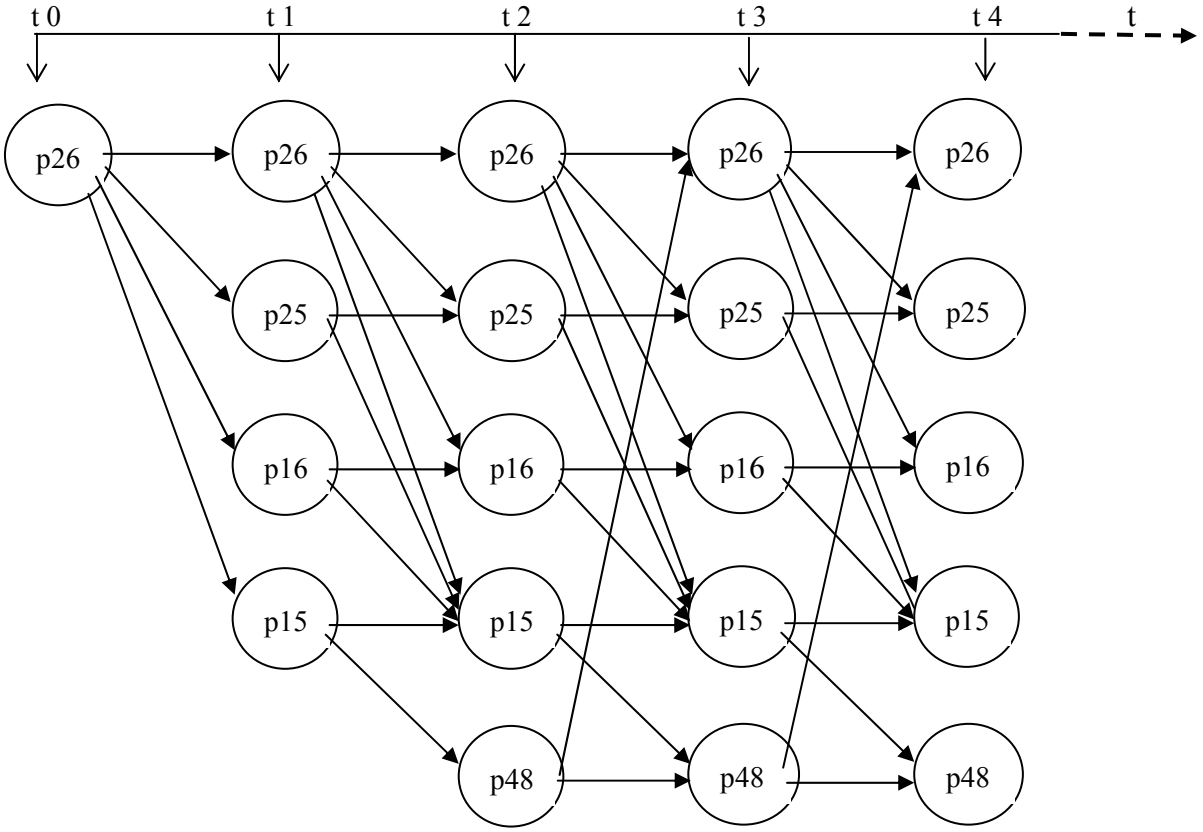


Figure 3.2 Diamond Interchange Signal Control Depicted in a Decision Network with Initial Phase 26

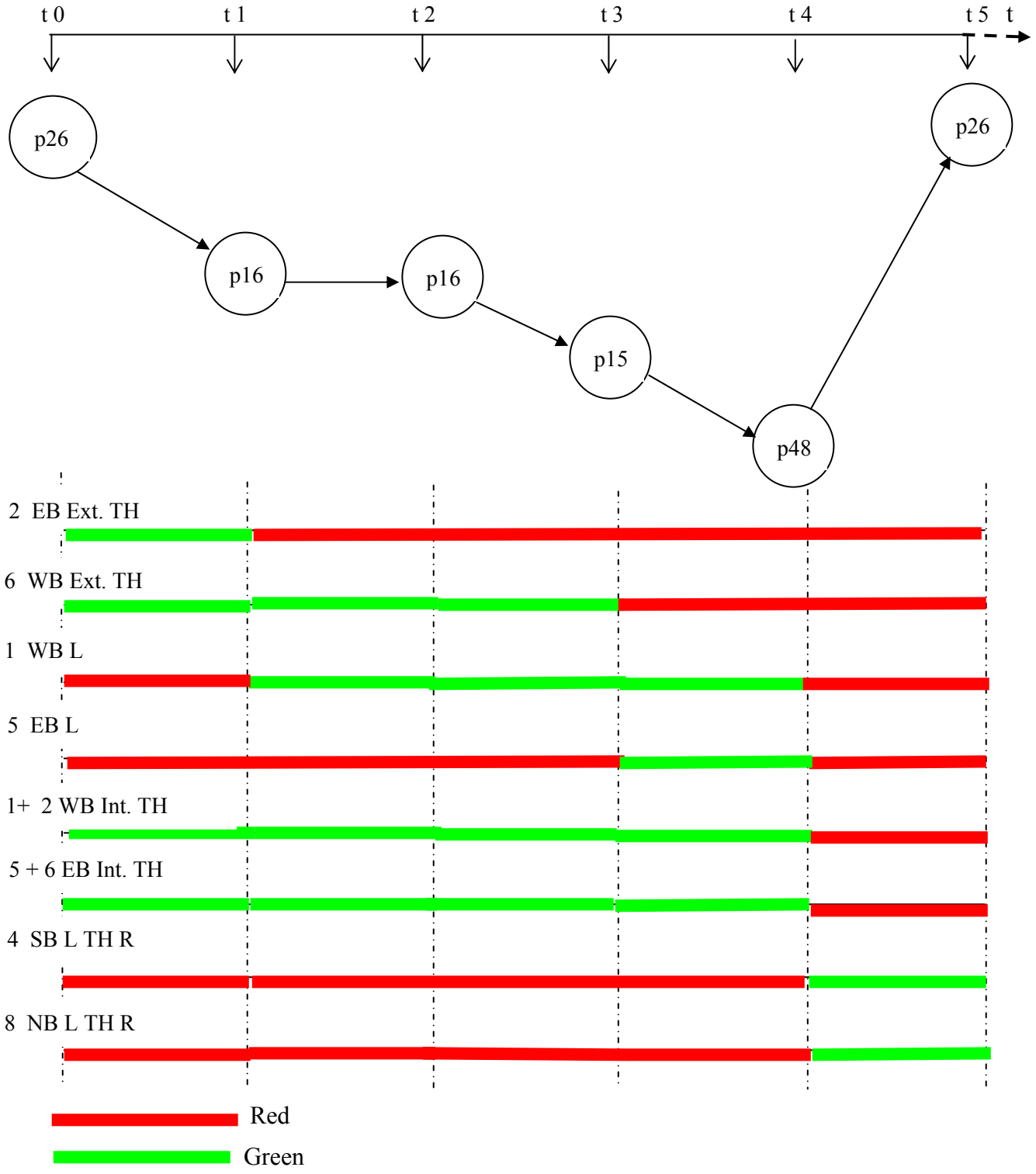


Figure 3.3 A Sample Signal Plan with Signal Indication for Each Movement

3.1.3 Optimal Decision-Making Process

In the adaptive signal control problem, the objective is to determine an optimal decision trajectory, or a signal plan of a sequence of phases at discrete times over a prescribed optimization period. A global optimal solution is often pursued with respect to a certain objective function that specifies operational performance of a diamond interchange. Performance measures may include delay, travel time, queue length, stop and other variables. These performance measures are defined over a single stage Δt and evaluated over the entire optimization period.

Major inputs to the optimization are traffic arrivals on individual approaches, together with initial signal phase and initial queue length while outputs are signal phases at discrete times over the entire optimization period. Traffic arrivals are predicted from upstream detectors.

Given an initial queue length and signal phase in the beginning of a stage, the queue length at the end of a stage is dictated by the state transition function (i.e., a function about how the signal state is switched) that is based on arrival-discharge dynamics, and the decision to extend or switch from the initial phase. Thus, an objective function for optimal signal control is a function of signal phase, traffic condition and state transition function.

Accordingly, with a decision network representation, the optimization solution procedure for an adaptive signal control problem involves the development of state transition function over a stage and the optimization of the objective function over an optimization horizon.

3.2 Dynamic Programming for Interchange Signal Control

3.2.1 Background

As presented in section 3.1, adaptive signal control of diamond interchanges is as a matter of fact a sequential decision making process. One straightforward approach to solving this problem is to enumerate all possible phase sequencing paths for the optimization horizon. However, the number of possible paths is very large, and having to calculate the performance for each path is neither an appealing task nor appropriate for real-time implementation. Thus, dynamic programming (DP) is adopted to solve this signal optimization problem as DP is a highly effective and efficient mathematical technique for making a sequence of interrelated decisions. Compared to exhaustive enumeration methodology, DP finds an optimal solution with much less effort and time. Figure 3.4 shows a decision tree (with respect to a specific movement) and the optimal signal switches, in dark line, found by DP.

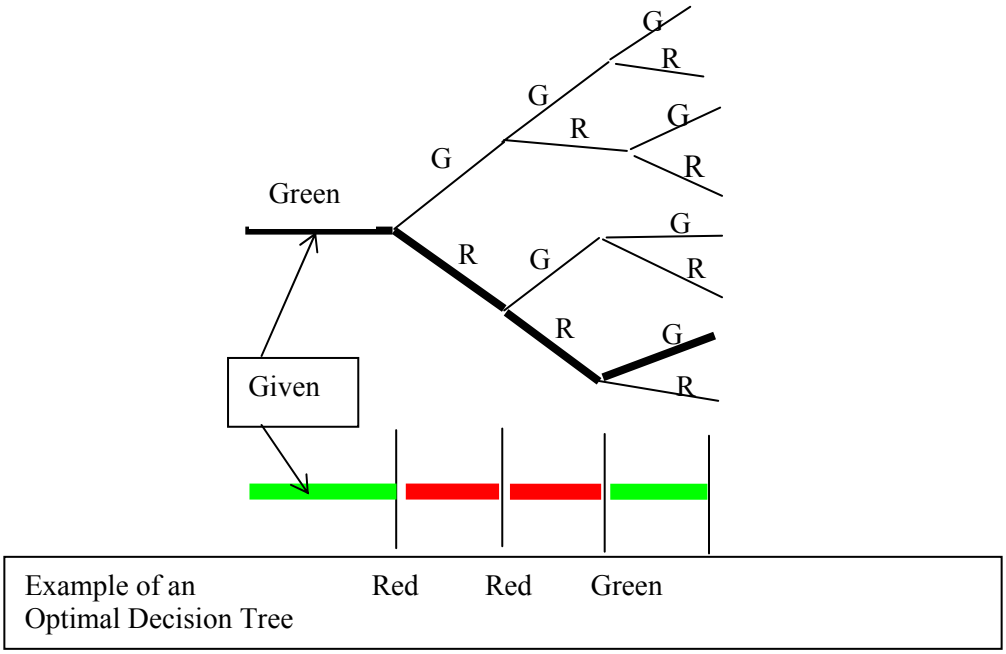


Figure 3.4 Decision Tree of Dynamic Programming Process

3.2.2 Dynamic Programming Formulation

3.2.2.1 Introduction

In this section a solution approach to the optimal decision trajectory based on DP is formulated. The central issue of signal control is to find a decision trajectory (or a sequence of phase extensions or switches at discrete times) that optimizes an objective function defining performance measures index (PMI). PMI is a combination of performance measures such as average delay, queue length, number of stops, or others. More discussions will be made in "Immediate Return Equation" section. In our DP formulation a signal plan is determined for a future time period defined as an optimization horizon. Each horizon consists of a number of stages i.e., the time is discretized at t = 0, t1, t2, ... The signal switching decisions are made at each stage. The signal may be changed at the boundary between stages. The objective function for optimization is the aggregated pre-defined performance measures index of a diamond interchange over the stages.

The DP formulation for an optimal signal plan may be either deterministic or stochastic, depending on the randomness involved in the traffic arrival-discharge dynamics. Deterministic DP defines the formulation where the state at the next stage is completely determined by the state and decision at the current stage. Stochastic, rather, defines that there is a probability distribution for what the next state will be. In this study, deterministic DP is considered since the very large state transition matrix calculation involved in the stochastic DP prevents it from being implemented in real time. Moreover, traffic arrivals pattern at the stop-line is deterministic since the arrivals are projected from the detector upstream in real time.

Deterministic DP could be solved in either backward recurrence or forward recurrence. To make the backward recurrence possible, given initial queue in the beginning of each optimization period, the queue lengths of all nodes throughout the entire decision network have to be calculated firstly. The separation of the queue length calculation in forward manner and the value iteration for optimal solution in backward manner for DP backward recurrence is a major disadvantage, which might double the computational amount and thus prevent it from being implemented in real time. Thus a

forward DP formulation is applied to our adaptive signal control problem due to its efficiency.

The forward DP formulation procedure suggested by Hillier and Lieberman (2001) and Hastings (1973) is adopted in this study. The following Sections define the DP terminology and interpret each term in the context of the signal optimization problem, together with the symbols used to denote the relevant quantities. The equations that are needed for the solution of the optimal decision trajectory are formulated. They include:

(a) Recurrence relation
(b) State transition equation
(c) Immediate contribution equation
(d) Optimal value function
(e) Terminal values

3.2.2.2 Terminology and Symbols for DP Adaptive Signal Control Problem

DP formulation terminologies have the general definitions, as described in textbooks. However, with respect to a specific problem - forward recurrence DP solution to the signal control problem, it is required to define and formulate these terms specifically. Table 3.1 lists both definitions.

The following DP formulation uses the basic three-phase diamond interchange ring structure (Figure 2.3) as an example, without loss of generality. Since diamond interchanges signal operation is not limited to this ring structure, the complete phase definitions and structure would be considered for the DP formulation in the future work.

Table 3.1 Terminology Definition and Symbols for Forward DP Formulation

Terminology	**General DP Definitions** (Hastings,1973)	**Definitions for Forward DP Adaptive Signal Optimization** (Refer to Figure 3.5)	**Symbol**
Stage	The original problem is divided into N stages	An interval (for example: 2 ~ 5 seconds) *n* = the index of stage, also the number of stages from the stage 0 to current stage	*n*
State	Each stage has a number of states associated with it. The states are the various possible conditions in which the system might be at each particular stage of the problem.	Signal phase state ***i*** for stage *n* *i=1*, phase 2 and 6 (p26): Green for movement 2,6, A and B; Red for others *i=2*, phase 2 and 5 (p25): Green for movement 2,5, A and B; Red for others *i=3*, phase 1 and 6 (p16): Green for movement 1,6, A and B; Red for others *i=4*, phase 1 and 5 (p15): Green for movement 1,5, A and B; Red for others *i=5*, phase 4 and 8 (p48): Green for movement 4 and 8; Red for others	*(n, i)*
Decision Variable	At each state there is a set of decisions from amongst which a choice must be made	Decision *k* is a signal phase leading to the state *i* at stage n. The decision departs from the state *j=k* at stage *n-1*. *k* is on decision space K_i ={1,2,3,4,5}={p26,p25,p16,p15,p48} Use the three-phase diamond interchange ring structure as an example: $K_i = \{1,5\}, \quad i=1$ $K_i = \{1,2\}, \quad i=2$ $K_i = \{1,3\}, \quad i=3$ $K_i = \{1,2,3,4\}, \quad i=4$ $K_i = \{4,5\}, \quad i=5$	*k*
State Transition	A function that shows the state for the next stage changes based on the current state, stage, and optimal decision	Signal state transition $j = f(n,i,k) = k$	

Immediate Return	Something which the system generates over one stage given the starting state and the decision	Queue length, delay or user-defined performance measures over stage *n*, due to state (*n-1, j*), queue length *q* and decision *k* (leading to state *i* at stage *n*). The defined performance measure(s) is named as performance measure index (PMI). It is the sum of the PMI of 8 approaches in a diamond interchange.	PMI *(n, i, q, k, arrivals)*
Recurrence Equation	Identifies the optimal policy at stage n, given that the optimal policy at stage n-1 is available	Equation (3.2)	
Optimal Value Of a State	Best total value from stage 0 to stage n, given that the starting state at stage 0, and a sequence of optimal decision is made	Minimum PMI at a given state, from stage 0 up to stage n under an optimal plan	$f^*(n, i)$

A **stage** is an interval of time. Its index value *n* stands for the number of stages completed from the beginning of the planning period. Also, "stage *n*" is for "interval *n*". An optimization horizon is made up of N stages.

A ***state*** is a signal phase for diamond interchanges. State *i* at the end of stage *n* is the state value of stage *n* and the signal phase over stage *n*, often denoted by state *(n,i)*. Initial state (or signal phase) in the very beginning of signal optimization period is given. Possible values of a state *i* are (using the three-phase ring structure as an example):

i =1, when phase 2 and 6 have the green

i = 2, when phase 2 and 5 have the green

i = 3, when phase 1 and 6 have the green

i = 4, when phase 1 and 5 have the green

i = 5, when phase 4 and 8 have the green

A **decision** is whether to extend the current signal state or switch it to one of the permissive signal states for the stage that follows. The decision *k* made at state *(n-1,j)* is a signal phase that leads to state *(n,i)*. The set of possible decisions K_i (i.e., the decision space) associated with state *(n, i)* depends on the value of state *i* (the current signal phase). According to the three-phase signal plan operational characteristics of a diamond interchange, decision space K_i are constrained by

$$
\begin{aligned}
K_i &= \{1,5\}, && i=1\\
K_i &= \{1,2\}, && i=2\\
K_i &= \{1,3\}, && i=3\\
K_i &= \{1,2,3,4\}, && i=4\\
K_i &= \{4,5\}, && i=5
\end{aligned}
$$

There are two barriers existing in this ring structure (Figure 2.3): one between phase pair 15 and 48, and the other between phase pair 26 and 48. This requires that the length of phase 2 plus phase 1 equals the length of phase 5 plus 6. This constraint has to be met logically when the time is discretized in a fixed interval, and the signal phase is extended or switched in a way the decision network specifies. This is one of the advantages of using a decision network to represent the signal control problem.

An **immediate return**, associated with state *i*, state *j*, decision *k*, initial queue length *q*, is defined as the performance measures index (PMI) that vehicles experience at an interchange over an interval. It is a sum of individual PMI that vehicles experience at each approach for a stage *n*. The PMI can be delays, queue lengths, stops, or a combination of them.

The **optimal value** $f^*(n, i)$ is the minimal PMI from stage 0 to stage *n* under an optimal signal switch plan (or a optimal decision path). The value is given by the sum of the vehicles' PMI at all approaches over the discrete intervals that make up the decision trajectory.

3.2.2.3 Forward Recurrence Equation

Forward DP is applied to determine the smallest PMI over stage 1 to stage N and the signal plan associated, given initial signal state and initial queue lengths at all approaches.

Figure 3.5 shows a decision network (the low half of the diagram) representing the signal control problem of the diamond interchange three-phase ring structure to be solved by forward recurrence DP (the upper part of the diagram). As pictured forward recurrence DP, the network nodes stand for phase states, arrowed lines for possible transitions of signal status and numbers against the lines for possible decisions.

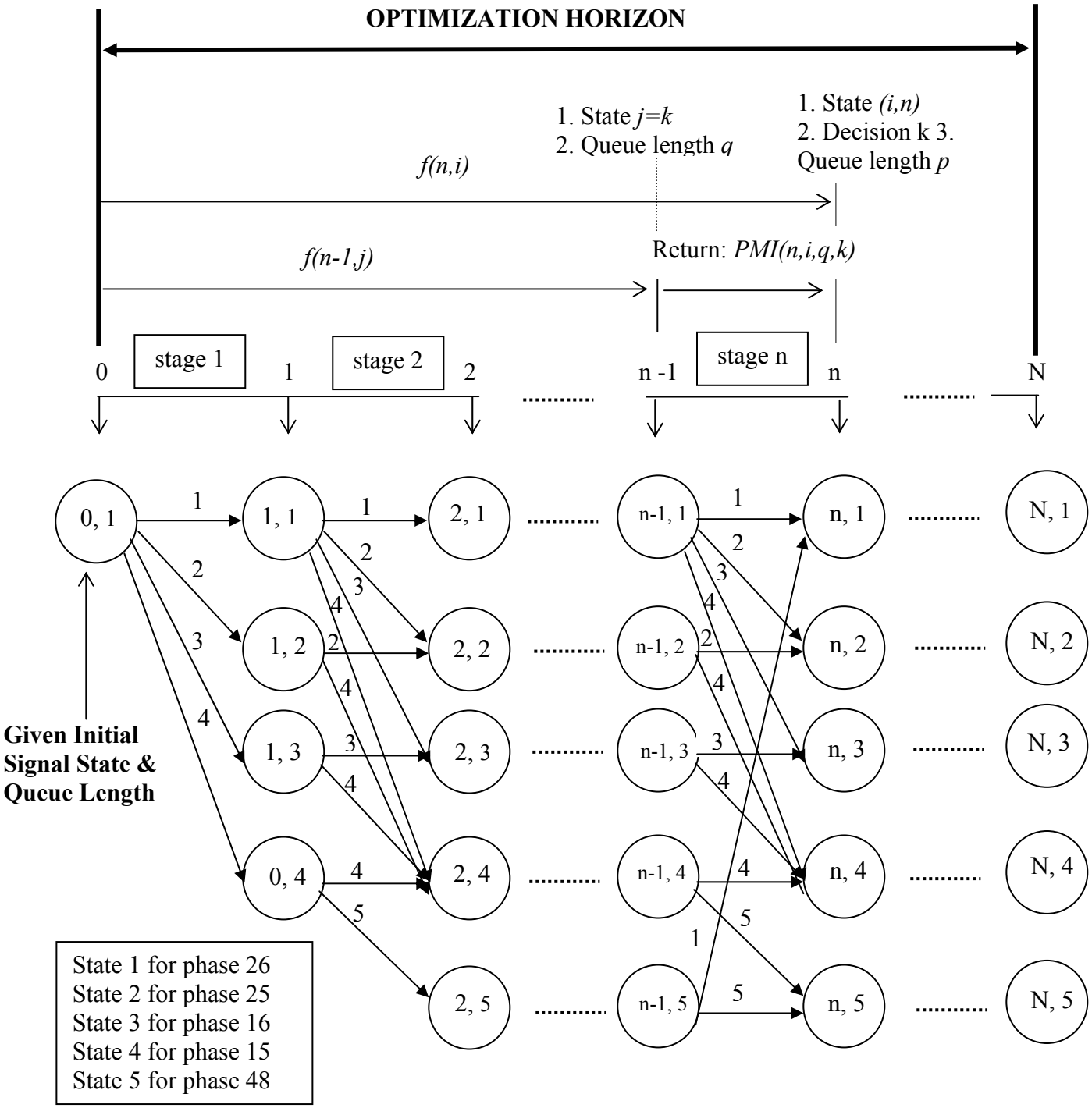


Figure 3.5 Signal Control in Decision Network Solved by Forward Recurrence DP

According to Hillier and Lieberman (2001), the forward recurrence relation is expressed as,

Minimal PMI value from stage 0 to stage n, at state *(n.i)*	=	Minimal PMI value from stage 0 to stage *n-1*, at state *(n-1, j)* that results in state *(n,i)* due to decision *k.*	+	**Minimum over all relevant decisions k_i at state *(n-1,j)* leading to state *(n,i)*** | PMI value due to decision 1 PMI value due to decision 2 ⋮

and mathematically,

$$f^*(n,i) = \underset{k \in K_i}{Min}\{PMI(n,i,k,q) + f^*(n-1,j)\} \tag{3.1}$$

Where

n = Stage number

i = Current state at stage n (or signal phase over the stage n)

q = Initial number of queue lengths at stage n. It is a vector of having the number of queue lengths of individual approaches as components for totally 8 approaches.

k = Signal phase switch decision at stage *n-1* that leads to state i at stage n.

K_i = Set of possible decisions leading to state i

j = State at stage *n-1* (or signal phase over the stage *n-1*)

PMI (n ,i ,k, q) = Return over stage n, due to decision k, state *(n-1, j)* changing to state *(n, i),* given initial queue lengths at stage *n-1*. More details see " immediate return equation" section.

$f^*(n\text{-}1, j)$ = Minimum PMI values from stage 0 to state *n-1*

$f^*(n, i)$ = Minimum PMI values from stage 0 to state n

An Example: Referring to Figure 3.5, suppose the current state (n=3,i=1) and the values associated with all nodes before n=3 have been calculated. Thus, $K_i=\{1,5\}$, that is, j=1 or 5 at stage 2 leads to i=1 at stage 3. According to the above equation, we have the recurrence,

$$f^*(3,1) = \underset{k \in K_1}{Min}\left\{PMI(n,q,k) + f^*(n-1,j)\right\}$$
$$= Min\left\{(PMI(3,q,1) + f^*(2,1)),(PMI(3,q,5) + f^*(2,5))\right\} \quad (3.2)$$

Where

$f^*(3,1)$	= Optimal value of state (3,1)
$f^*(2,1)$	= Optimal value of state (2,1)
$f^*(2,5)$	= Optimal value of state (2,5)
$PMI(3,q,1)$	= The return associated with a transition of node (2,1) to node (3,1), and initial queue length q at node (2,1)
$PMI(3,q,5)$	= The return associated with a transition of node (2,5) to node (3,1), and initial queue length q at node (5,1)

3.2.2.4 Signal State Transition

A decision k is the extension or switch of a phase state *(n-1,j)* that leads to a state *(n,i)*. The transition equation relates the predecessor state *(n-1,j)* to the current state *(n,i)* under decision *k*. It is expressed as

$$j = t(n,i,k) = k \quad (3.3)$$

Where

t = State transition function

n = Stage number

k = Decision at state *(n-1,j)* leading to state *(n,i)*

i = State value at stage *n*

j = State value at stage *n-1*

3.2.2.5 Immediate Return Equation and Terminal Values

Immediate return is associated with the performance measures used for the signal control of diamond interchanges. A general performance measures index (PMI) is defined to consider queue lengths, delays, stops or a combination of them. The return associated with state *(n,i)* and decision *k* is denoted by *PMI (n,q,k)*. It is a sum of each PMI value at 8 approaches over stage *n* due to a transition to the current state *(n,i)* from the previous state *(n-1,j=k),* given the initial queue length *q* at stage *n-1*,

$$PMI(n,q,k) = \sum_{i=1}^{8} PMI[i] \tag{3.4}$$

Where

$$PMI[i] = PMI(n,q,k), \quad i = 1,2,...,8 \tag{3.5}$$

Where

PMI[i] = queue length, delay or other performance measures at approach *i* over stage *n* due to the signal status *k* for approach *i* over the stage, given the initial queues length *q* of approach *i* in the beginning of stage *n*.

Among all performance measures, queue lengths can be the most direct "return" generated from different signal decision. The following presents the concept of queue transition over a stage. The detailed calculation of each performance measure will be given by traffic arrival-discharge dynamics in Section 3.3.

Individual approach is treated independently in terms of vehicles queuing and discharge. Given the initial queue length *q* at the beginning of stage *n*, for a decision *k* leading to state *(n, i)* from state *(n-1, j=k),* the resultant queue length *p* at stage *n* is calculated by

$$p_a = qt(n,i,q_a,k_a) \qquad a = 1,2,\cdots,8 \tag{3.6}$$

Where

qt = Queue transition function. It is given by the traffic dynamics in Section 3.3.

n = Stage number

p_a = Queue length of approach a in the end of stage n

q_a = Queue length of approach a in the beginning of stage n

k_a = Signal status for approach a over the stage n

A terminal value is the value of a state at stage 0. It is assumed that a zero terminal value for all possible states at stage 0 is used, i.e.,

$$f(0,i) = 0, \qquad i = 1,2,\cdots,5 \tag{3.7}$$

3.2.2.6 Algorithm Discussion

Figure 3.6 presents the block diagram for forward DP algorithm. It involves three major loops. The inner loop (decision loop of 2 or 4 iterations) corresponds to finding the best decision at a state. The middle loop (phase state loop of 5 iterations) takes the procedure from one state to another at each stage. The outer loop (stage loop of N iterations) then does the recurrence by forwardly progressing the stage to determine the optimal decision and value for every state.

The value iteration in forward manner involves starting from terminal states of known values and working through the problem from the beginning to end of an optimization period. Thus, the forward recurrence is advantageous as the state transition and queue length updating are calculated and progressed simultaneously with the value iteration. The value of a state, state transition and queue length transition can be extended forward in time without earlier calculation having to be repeated and also without a need of advance calculation of queue transition prior to the value iteration.

Both forward queue calculations and value iterations are performed recursively, and as a consequence, any errors and imprecision involved in traffic dynamics and the DP numeric algorithm seemingly accumulate. But this will not become an actual problem because the initial queue of each optimization horizon is calculated based on real detected vehicular arrivals. Nevertheless, to avoid the accumulated errors unacceptable the DP program may also be re-set after a certain time (e.g., 60 minutes) has elapsed.

The forward recurrence DP formulation of the diamond interchange adaptive signal control has been completed. The equations for recurrence relation, state transition, and immediate return contribution (i.e., performance measures index), as well as optimal value function, terminal values have been formulated. The following sections will further develop the mathematical functions for calculating performance measures index by

taking into account phase definitions, detector placement and arrival-discharge projection dynamics.

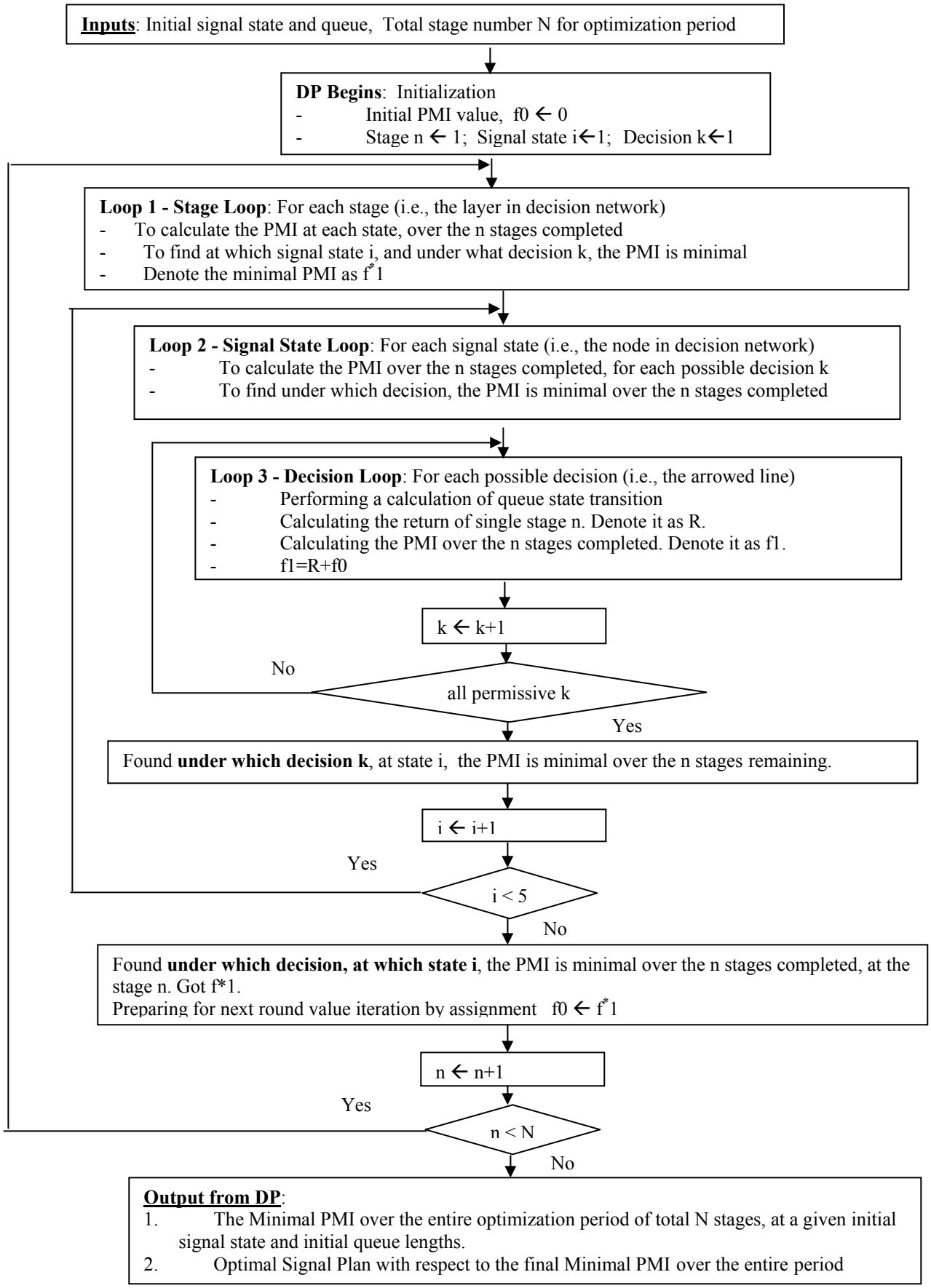


Figure 3.6 Block Diagram for Forward Recurrence DP

3.3 Vehicle Arrival-Discharge Projection Dnamics

3.3.1 Introduction

As discussed in Section 3.1, real-time signal control for an optimal signal plan has been represented in a decision network. The time is discretized in a constant time interval and a signal plan (or, a decision trajectory) is a sequence of phases of the interval length that are concatenated. The DP solution approach to an optimal signal plan, as discussed in Section 3.2, is performed over stages (or, intervals) and the return function (or, the objective function for optimization) that is a function related to queuing process is calculated on an interval-basis. Thus, the fundamental issue for optimization of interchange signal control is to calculate delay, queue length and other performance measures over a single interval under a certain signal phase. This can be further simplified into the calculations of arrivals; queue length; delay associated with individual approaches under signal Green or Red.

In what follows we will focus on an individual approach at a microscopic level and study its arrival projection, discharge process and performance over a single interval t. It is assumed the queue length at the beginning of an interval is given and signal status for the concerned interval is Green or Red.

3.3.2 Queued Vehicles at the Stop-line

Adaptive signal control requires reliable vehicle arrival and departure information at the stop-line for optimization. Usually this arrival information is predicted from arrivals detected at the upstream detector line. According to one of the first models for arrival projection (i.e., OPAC), the number of vehicles arriving at and stopped at the stop-line during an interval t is equal to the number of vehicles detected by detectors during an interval T= t. This holds only for an approach that is given green signal phase without initial queued vehicles. This projection scheme is not applicable for a red or a green approach with initial queued vehicles. Due to the queued vehicles present at the stop-line, the travel distance becomes the distance between the back of queue and the detector line, which is shorter than the distance between the stop-line and detector line. Therefore,

more vehicles would be queued during the interval T (or t). An extreme scenario is illustrated in Figure 3.7, where $T >> t$.

As stated above, queued vehicles change the distance between the back of queue and the detector, and the arrival projection process becomes more complicated. In the case of a signalized diamond interchange, queuing is the major problem involved in the traffic operation and queues particularly exist at the internal links. Hence, it is necessary to explore the vehicle arrival and discharge at the stop-line in a more microscopic way. Depending on the queue length at the stop-line, the detection range varies for a particular stop-line interval t. To precisely predict the vehicle arrivals at the stop-line, it is necessary to determine each detection range (T) corresponding each interval (t). The following develops the arrival projection and queue discharge dynamics for one approach of an interchange.

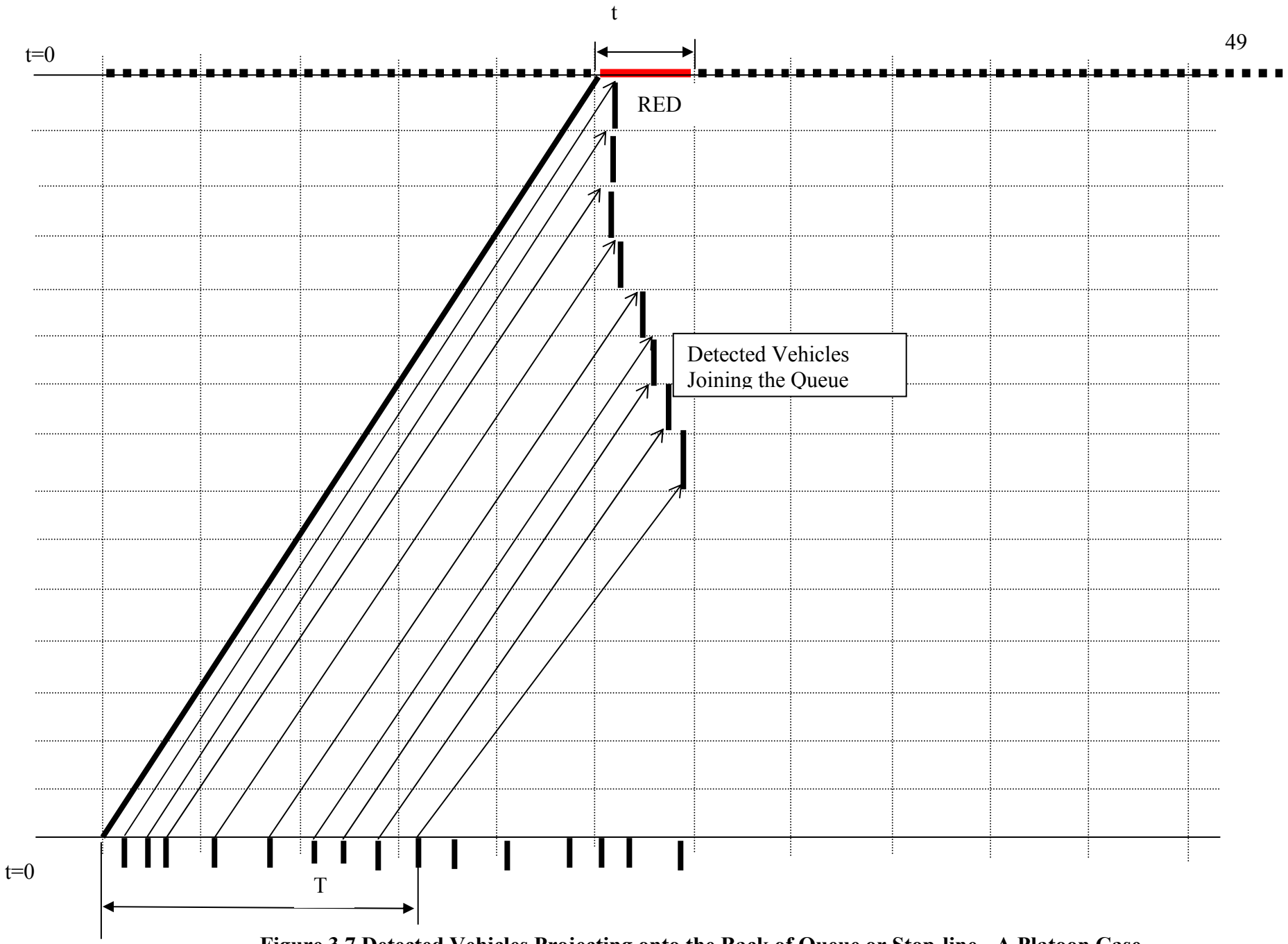


Figure 3.7 Detected Vehicles Projecting onto the Back of Queue or Stop-line - A Platoon Case

3.3.3 Vehicle Travel Time

Figure 3.8 presents the calculation scheme of vehicle travel time. Detectors are set upstream of the stop-line. The distance between the detector and the stop-line is denoted as *D*. It is assumed that a vehicle passes the detection line and travels with a constant speed (detected speed *V*) for T_1, then it starts to decelerate with a deceleration rate *a* to join the back of queue at the stop-line. The deceleration period is denoted as T_2.

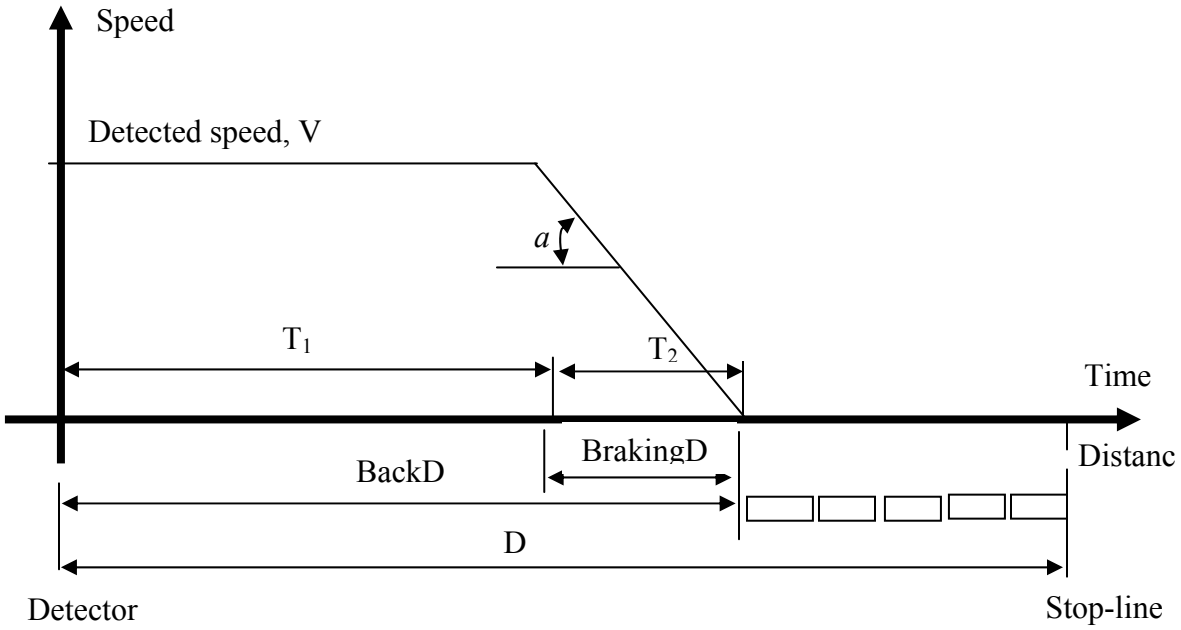


Figure 3.8 Travel Time Model

A vehicle's travel time from the detector to stop-line or the back of queue is calculated as follows, with respect to different signal and queuing situation at the stop-line.

Case a: If there is no queue and the signal is green, the vehicle travels with speed V:

$$T = \frac{D}{V} \tag{3.8}$$

Case b: If there is no queue and the signal is red, the vehicle is decelerating:

$$T = T_1 + T_2 = \frac{D - \frac{V^2}{2a}}{V} + \frac{V}{a} \tag{3.9}$$

Case c: If there are initial queued vehicles present:

$$T = T_1 + T_2 = \frac{D - (q * l) - \frac{V^2}{2a}}{V} + \frac{V}{a} \qquad (3.10)$$

Where

V: The detected speed of a vehicle at the detector, ft/s

q: The number of vehicles queued per lane at the stop-line

a: The deceleration rate, ft/s^2

l: The queued vehicle spacing, ft

D: The distance between detectors and stop-line, ft

T_1: The travel time of a vehicle with constant speed, sec

T_2: Vehicle braking time, $T_2 = \frac{V}{a}$, sec

Based on Eq. (3.10), the travel time difference between two consecutive vehicles joining the back of queue is calculated as below, assuming no queued vehicles discharged,

$$\text{Travel Time Difference} = \{\frac{V}{a} + \frac{D - (q * l) - \frac{V^2}{2a}}{V_{sa}}\} - \{\frac{V}{a} + \frac{D - (q-1) * l - \frac{V^2}{2a}}{V_{sa}}\}$$

$$= \frac{-q * l + (q-1) * l}{V} = \frac{l}{V} \qquad (3.11)$$

Assuming an average vehicle spacing of 23ft and average vehicle speed of 30 ~ 35mph (44 ~ 51.3 ft/s), then the travel time difference calculated from the above equation is about 0.5s (0.523s ~ 0.448s). That is, one more (or less) queued vehicle at stop-line decreases (or increases) the travel time duration by 0.5s.

3.3.4 Vehicle Projection Dynamics and Detection Range

The detection range (or detection period) (*T)* is the time duration in which the vehicles arrive at detectors and will be projected onto the back of queue or the stop-line during the corresponding interval (t). For an interval at the stop-line, the queue length change is estimated for either a green or a red signal. As the queue length changes, the

travel time changes. As a result the detected vehicles at the detector line which are able to arrive at the stop-line or back of queue changes. Thus, the detection period that matches the DP interval varies dynamically.

This section develops a relationship between time interval at the stop-line (t) and corresponding detection range (T). As shown in Figure 3.9, the detection range for vehicle projection in an interval is adjusted due to vehicle projection dynamics. Vehicle projection dynamics states that the net increase in queue length shortens the vehicle travel time, therefore, with one more (or less) vehicle queued at the stop-line, the detection period is 0.5 second longer (or shorter). This conclusion is reached based on the travel time estimation in Section 3.3.2.

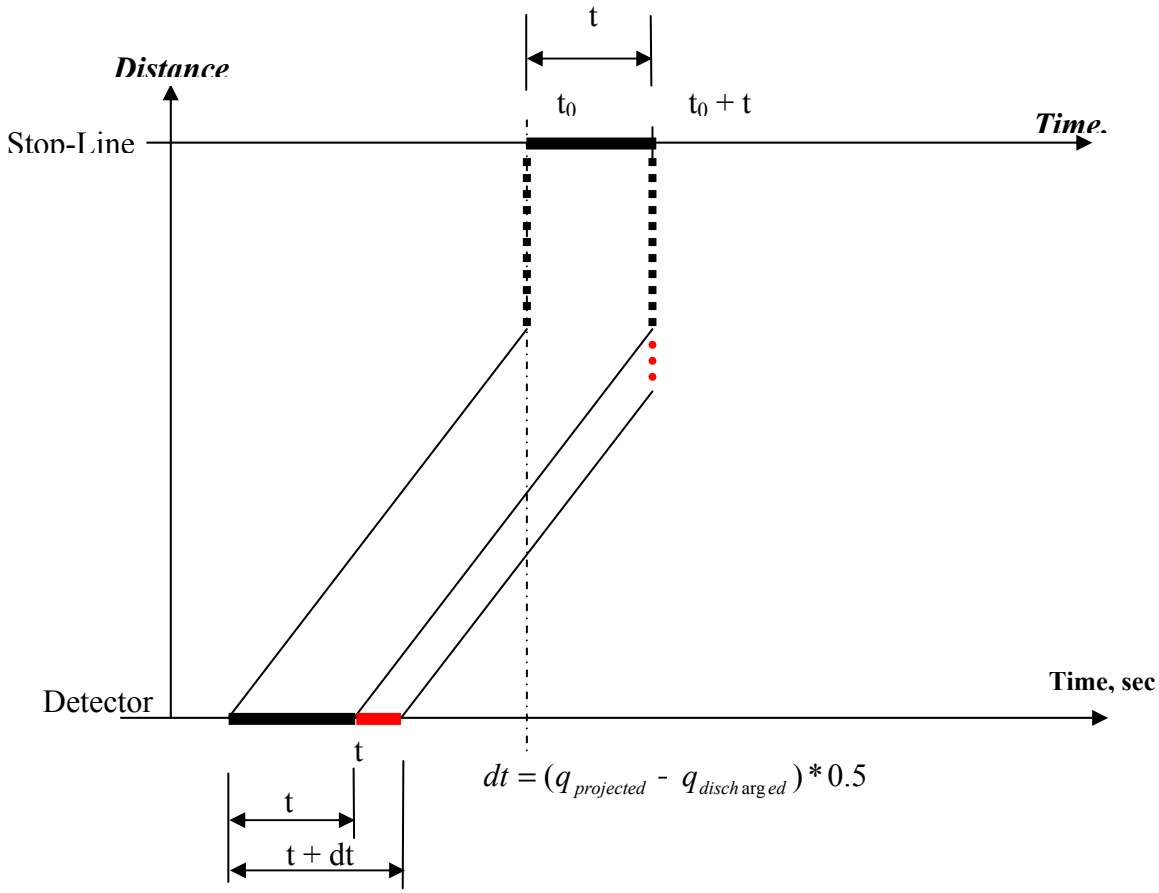


Figure 3.9 Vehicle Projection Dynamics

The following presents the mathematical relationship between the time interval at the stop-line (t) and the corresponding detection range (T) in terms of the change of queue length at the stop-line.

If the queue length remains unchanged during an interval (t),

$$\Delta T = \Delta t \tag{3.12}$$

Where

t: A time interval at the stop-line. It also represents a DP interval.

T: The corresponding detection range for this DP interval

If the queue length changes during an interval (t),

$$\Delta T = \Delta t + dt \tag{3.13}$$

Where

dt: An adjustment of detection range due to vehicle projection dynamics at the stop-line.

$$dt = (q_{projected} - q_{discharged}) * 0.5 \tag{3.14}$$

Where

$q_{projected}$: The number of vehicles detected during [T_0, T_0+ t] will be projected to the stop-line or the back of queue during an interval

$q_{discharged}$: The number of vehicles discharged during an interval

If queue length becomes longer (when $q_{projected} > q_{discharged}$), then the detection range is larger than t. Otherwise, the detection range is shorter than t.

The starting time of each detection range, T_0, is illustrated in Figure 3.10 . It can be determined using the following equation:

$$T_0 = T - i * \Delta t \tag{3.15}$$

Where,

T_0: The starting time of detection range, expressed as the time in advance of the optimization horizon. The vehicles detected during [T_0, T_0+ t] will be projected to the stop-line or back of queue during an interval

T: Travel time of a vehicle

i: DP interval index, i =1, 2, 3, 4

t: An interval at the stop-line.

In summary, the above vehicle projection scheme is developed at the microscopic level involving the vehicle detections, vehicle arrivals and departures. Therefore, the actual traffic flows at the stop-line for each optimization horizon can be predicted with a certain degree of accuracy.

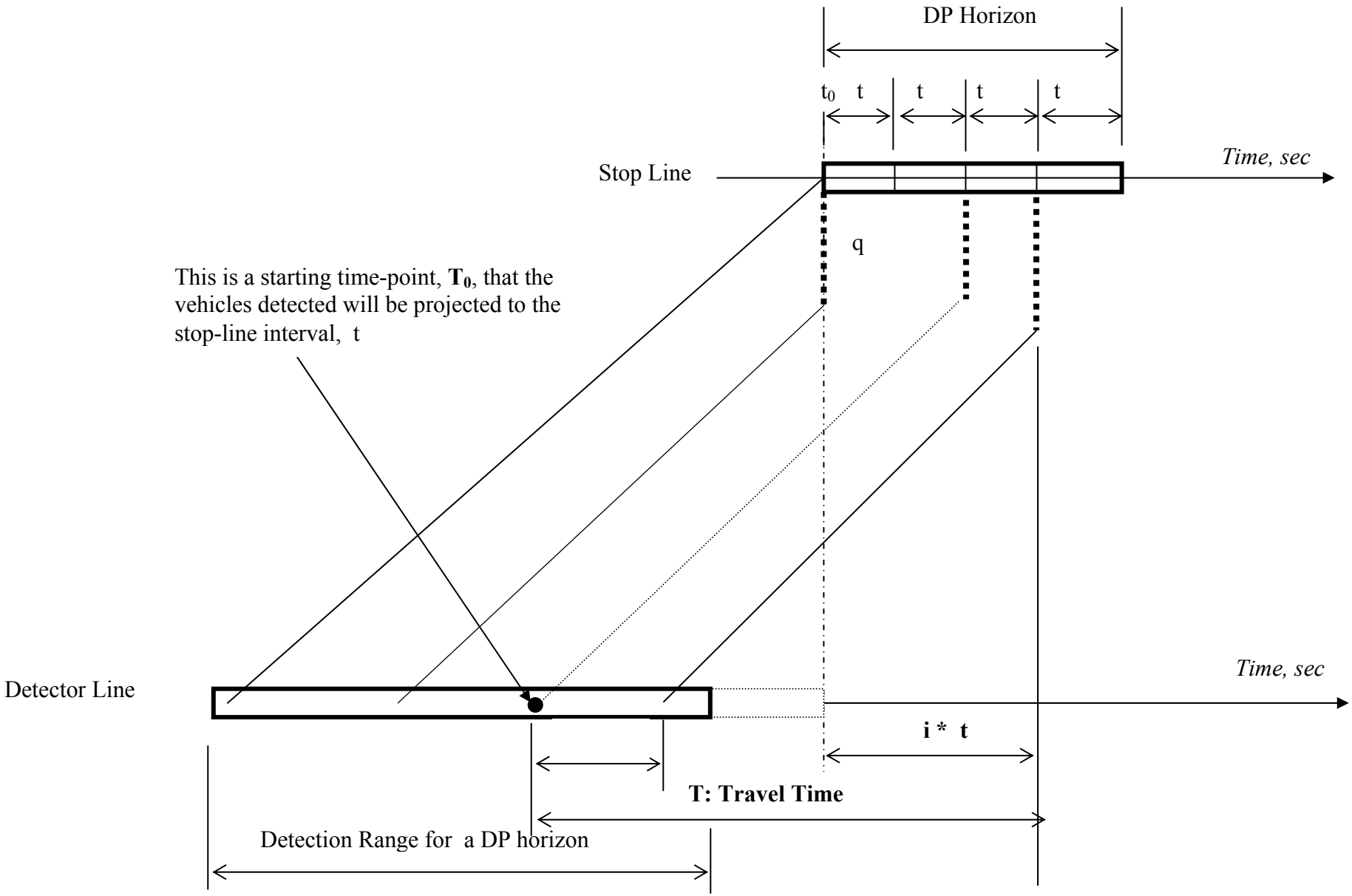


Figure 3.10 Vehicle Projection Interactive

3.4 Calculation of Performance Measures for one DP Interval

Various performance measures can be calculated for one DP interval due to different signal decision, either green or red. They include queue length, stop, delay, storage ratio and a combination of them. This study has defined performance measures related to queue length, delay and storage ratio, as follows,

1. The Average Number of Queued Vehicles Per Lane during a DP Interval

The average queue length per lane at the end of an interval is calculated by

$$q_{final} = q_{initial} + q_{projected} - q_{discharged} \tag{3.16}$$

Where,

q_{final}: The number of queued vehicles in the end of an interval

$q_{initial}$: The number of queued vehicles in the beginning of an interval

$q_{projected}$: The number of vehicles arrived during an interval

$q_{discharged}$: The number of vehicles discharged during an interval

If the signal is red,

$$q_{discharged} = 0 \tag{3.17}$$

If the signal is green,

$$q_{discharged} = t / adhw \tag{3.18}$$

Where,

adhw: average discharge headway

Note that q_{final} is also the initial queue $q_{initial}$ of the next interval.

The queue length calculation has to be performed at each node in the decision network, which depends on the previous final queue and the current signal status. This final queue is further dependent on the previous initial queue and the previous signal status. Consequently, any error in calculation or estimation might accumulate. In order to improve the accuracy, the initial queue in the very beginning of a horizon is detected in this study.

2. Average Delay Per Lane during a DP Interval

During an interval, the total delay is the sum of delays experienced by the queued vehicles and the back-of-queue catching vehicles. The following presents the equation to calculate the total delay that all vehicles experience at the stop-line:

$$Delay = \Delta t * q_{initial} + 0.5 * (q_{projected} - q_{discharged}) \quad (3.19)$$

It is assumed that if a vehicle arrives during a DP interval (either green or red signal) and unable to be discharged, then its delay time during this interval is 0.5* t.

3. Total Delays during a DP Interval

$Total\ Delay = NL * Delay$

Where,

NL: the number of lanes of an approach

Delay: Average delay per lane, calculated by Eq. 3.19

4. Storage Ratio during a DP Interval

It is defined as the ratio of the queues during a DP interval to the available space for queue storage.

$$Storage\ Ratio = \frac{q_{final}}{L / S} \quad (3.20)$$

Where

q_{final}: The number of queued vehicles in the end of an interval

L: Link length of an approach

S: Average spacing of queued vehicles

CHAPTER 4
ALGORITHM IMPLEMENTATION

This chapter describes the DP algorithm implementation. Section 4.1 provides the guidelines on how to determine several basic parameters using an example. Section 4.2 addresses detector setup along with a layout of detectors placement at a diamond interchange. Section 4.3 discusses the relationship between the DP optimization horizon and the corresponding detection range. This is followed by a description of optimal signal plan implementation in Section 4.4. The majority signal rolling technique is also explained in this section. Finally, a framework of algorithm implementation is provided in Section 4.5.

4.1 Basic Parameters for Implementing the DP Algorithm

To implement the DP method developed in Chapter 3, it is essential to determine the values of some parameters such as the distance of detectors to the stop-line, DP horizon and each interval, and detection range. Those values vary with the field detected data such as average vehicle speed, queue length at the stop-line, etc. This study uses an example illustrated as below to provide guidelines on how to determine these basic parameters for implementing the DP algorithm

As stated in Section 3.3, when there are queued vehicles present at the stop-line, the vehicle travel time from the detector to the back of queue can be estimated using equation 3.10. The following presents a look-up table for vehicle travel time, T (the time taken for a vehicle to travel and stop at the back of queue) with respect to various queue length at the stop-line, assuming that vehicle average speed is 30 mph (i.e., 44 ft/sec), average vehicle deceleration rate is 19.5 ft/s^2 and queued vehicle spacing is 23 ft.

Table 4.1 Travel Time at Various Queue Length Present

Number of Queued Vehicles	T_1: Travel Time with Constant Speed (sec)	T_2: Vehicle Braking Time (sec)	Travel Time $T=T_1+T_2$ (sec)	Distance from Detector to the Back of Queue
0	2.26	13.78	**16**	656
1	2.26	13.26	**16**	633
2	2.26	12.74	**15**	610
3	2.26	12.21	**14**	587
4	2.26	11.69	**14**	564
5	2.26	11.17	**13**	541
6	2.26	10.64	**13**	518
7	2.26	10.12	**12**	495
8	2.26	9.60	**12**	472
9	2.26	9.08	**11**	449
10	2.26	8.55	**11**	426

Based on the above estimation, by considering different queue length at the stop-line, the following parameters are determined in order to implement the DP algorithm:

a) Detectors setback at 656 ft from the stop-line for all approaches

   This setup is selected in order to ensure reliable traffic information of more than 10 seconds in advance even when there is certain number of queued vehicles present (e.g., 10). The distance is also corresponding to the vehicle travel distance with zero queued vehicles at the stop-line.

b) DP horizon at stop-line is 10 seconds and it is divided into 4 intervals. Each DP interval is 2.5 seconds

   This takes into account the minimum travel time that can be achieved with the pre-determined detector setup and detected vehicle speed, and the minimal green time that can be implemented. Minimum travel time represents the minimum time during which traffic information can be obtained in advance.

c) Detection range for vehicle projection is 16 to 2 seconds in advance of the corresponding DP horizon.

   This predicts the maximum detection range for one optimization horizon. Depending on queue length at the stop-line, the corresponding detection range varies. The boundaries of maximum detection range is corresponding to the

scenarios when there is no queued vehicle in the beginning of the DP horizon and there are maximum queued vehicles (e.g., 10) in the end of the horizon.

4.2 Detector Setup

Figure 4.1 presents a detectors placement layout for implementing the DP algorithm. A diamond interchange has 8 lane groups including NB, SB, EB external, WB external, EB internal, WB internal, EB left turning and WB left turning. Detectors are located upstream some distance (e.g., 656 ft) for all lane groups. These detectors should have the capabilities of measuring the number of vehicles passing through and their speeds.

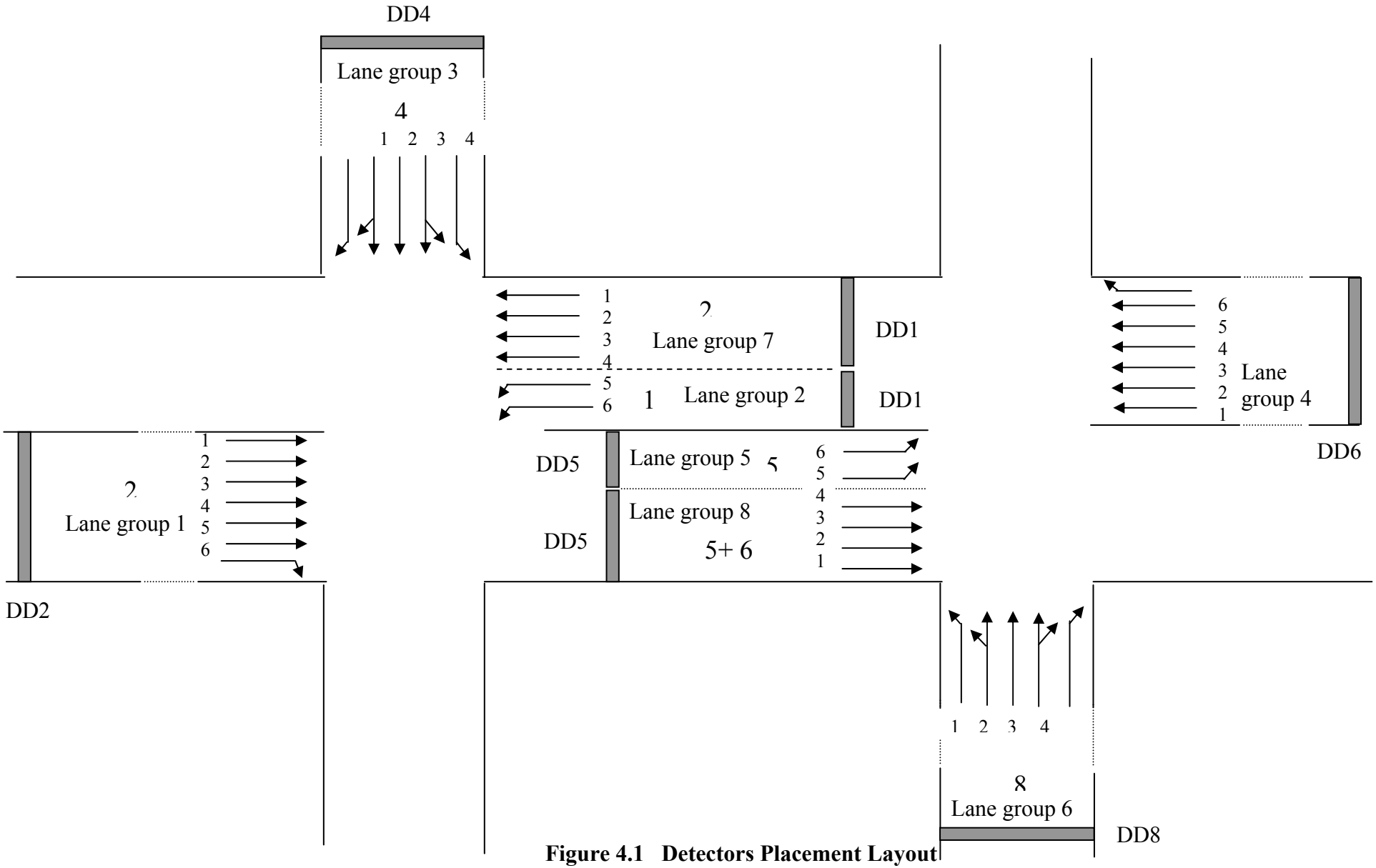


Figure 4.1 Detectors Placement Layout

4.3 DP Optimization Horizon and the Corresponding Detection Range

Figure 4.2 shows the relationship between DP optimization horizon and the corresponding detection range using an example stated in Section 4.1. Detectors are setback some distance (e.g., 565 ft) from the stop-line. During the detection period, the detectors detect the presence of a vehicle and its speed, which in turn estimates its arrival time at the stop-line sometime during the 10 seconds optimization horizon. Depending on the queue length at stop-line, detection periods are not constant and vary from 16 to 2 seconds in advance of the corresponding optimization horizon. Each vehicle's detection time is also located according to the vehicle projection dynamics in section 3.3.3. After determining the detection range, the number of vehicles detected during that period are projected to the horizon intervals (0-2.5), (2.5-5), (5-7.5) and (7.5-10).

To increase the measurement precision and properly implement the majority signal rolling technique (see section 4.4), two consecutive DP optimization horizons are designed to have one interval of 2.5s overlap. Consequently, there is also an overlap between detection periods. As shown in Figure 4.3, two consecutive DP horizons can share the same detection information for up to 6.5s (-8.5s ~ -2s).

.

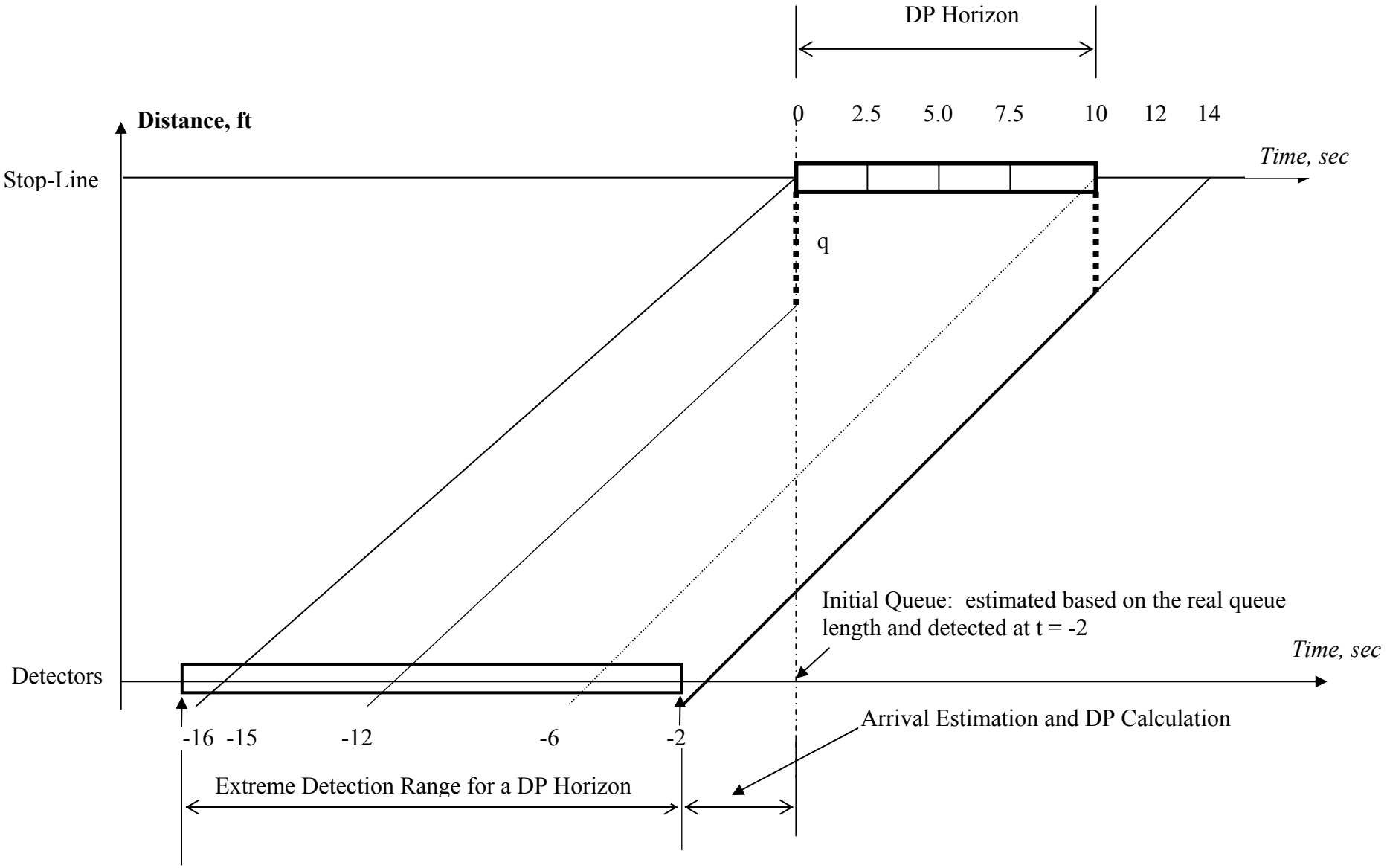


Figure 4.2 Detected Vehicle Projection Scheme I

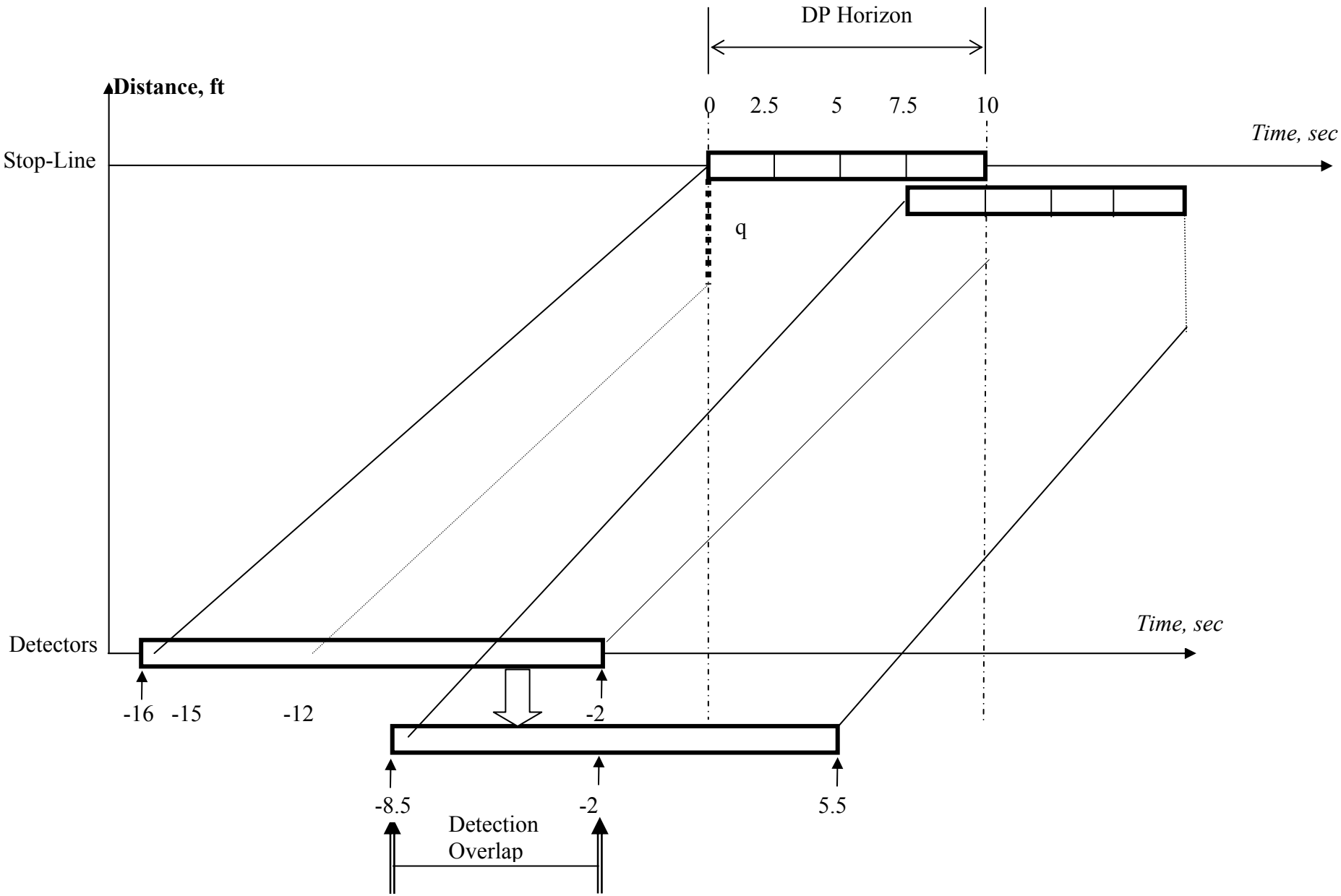


Figure 4.3 Detected Vehicle Projection Scheme II

4.4. Implementation of Optimized Signal Plan

To implement the optimized signal plan according to practical scenarios, a so-called majority rolling technique was developed. This technique can be stated as follows: over a single horizon of 10 seconds, the signal phase chosen most frequently is implemented for 7.5 seconds followed by a yellow-and-all-red clearance time of 2.5 seconds; and if two consecutive horizons hold the same majority signal then no clearance time is inserted between them.

The majority signal phase is defined as the one option that appears most among four DP optimal solutions for a DP horizon. For each optimization horizon of 10s, a majority signal phase is implemented for either 5.0 or 7.5 seconds, depending on the following two conditions:

if this majority signal phase is the same as the previous phase,

    it is implemented for 7.5 seconds

Otherwise,

    implement a 2.5s yellow-and-all-red clearance time followed by 5s green of the majority signal phase

An illustrative example is given in Figure 4.4. Suppose phase 26 and phase 48 are the majority signal option for the two consecutive horizons (0-10s and 7.5-17.5s). Thus, phase 26 is implemented for time 0 – 7.5s, followed by 2.5 seconds yellow-and-all-red and then by phase 48 for 7.5-15s. The signal option between 15 and 17.5s will be decided by the majority signal of the horizon 15-25s.

The majority signal rolling allows for implementation of sufficient long green and yellow-red intervals by avoiding the frequent switches of the signal phases of small intervals. It also tends to reduce the accumulation of arrival projection errors due to an overlapped interval of 2.5s. However, this scheme sacrifices some of computations in order to be implemented practically and does not fully represent optimal solutions.

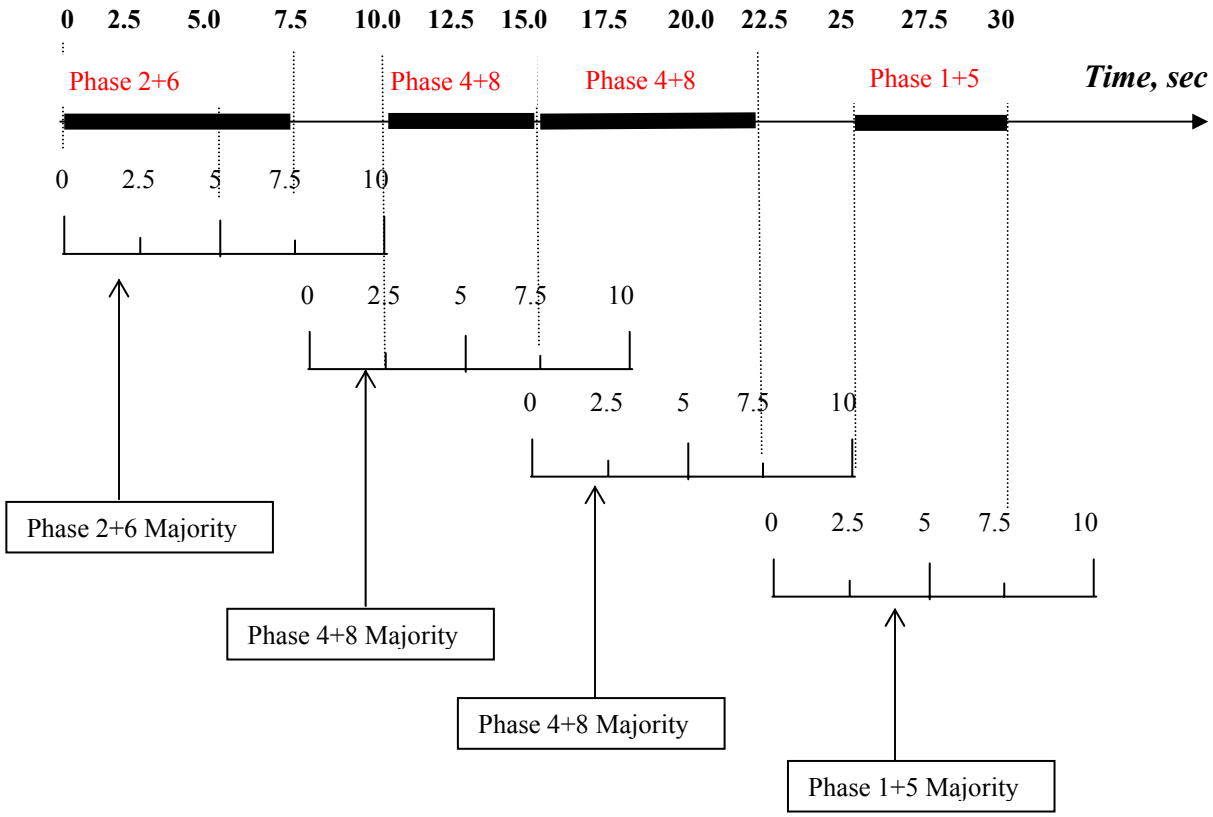


Figure 4.4 Signal Implementation - Majority Rolling Technique

4.5 Framework of Algorithm Implementation

The framework and procedure for implementing the DP-based adaptive signal control is presented in Figure 4.5. Inputs include inital signal phase, inital queue length, and continuously undated vehicle information at detectors. The arrival projection model is employed to estimate the vehicle arrivals over the entire optimization horizon and the performance measure model is used to calculate queeu length, delay, etc. for each interval. Initial queues in the beginning of each horizon can be estimated based on queueing information and signal phase at the end of detection range, as shown in Figure 3.12. Based on this information, the DP algorithm works at one horizon and the optimization is completed before the implementation time starts. For example, as shown in Figure 3.12, arrival estimation and DP calculation is done during 2 seconds before time 0. The signal phase switch sequencing is then updated at the controller in due time. With

the time rolling one horizon forward, the DP algorithm restarts optimization and the optimal signal phases are then implemented.

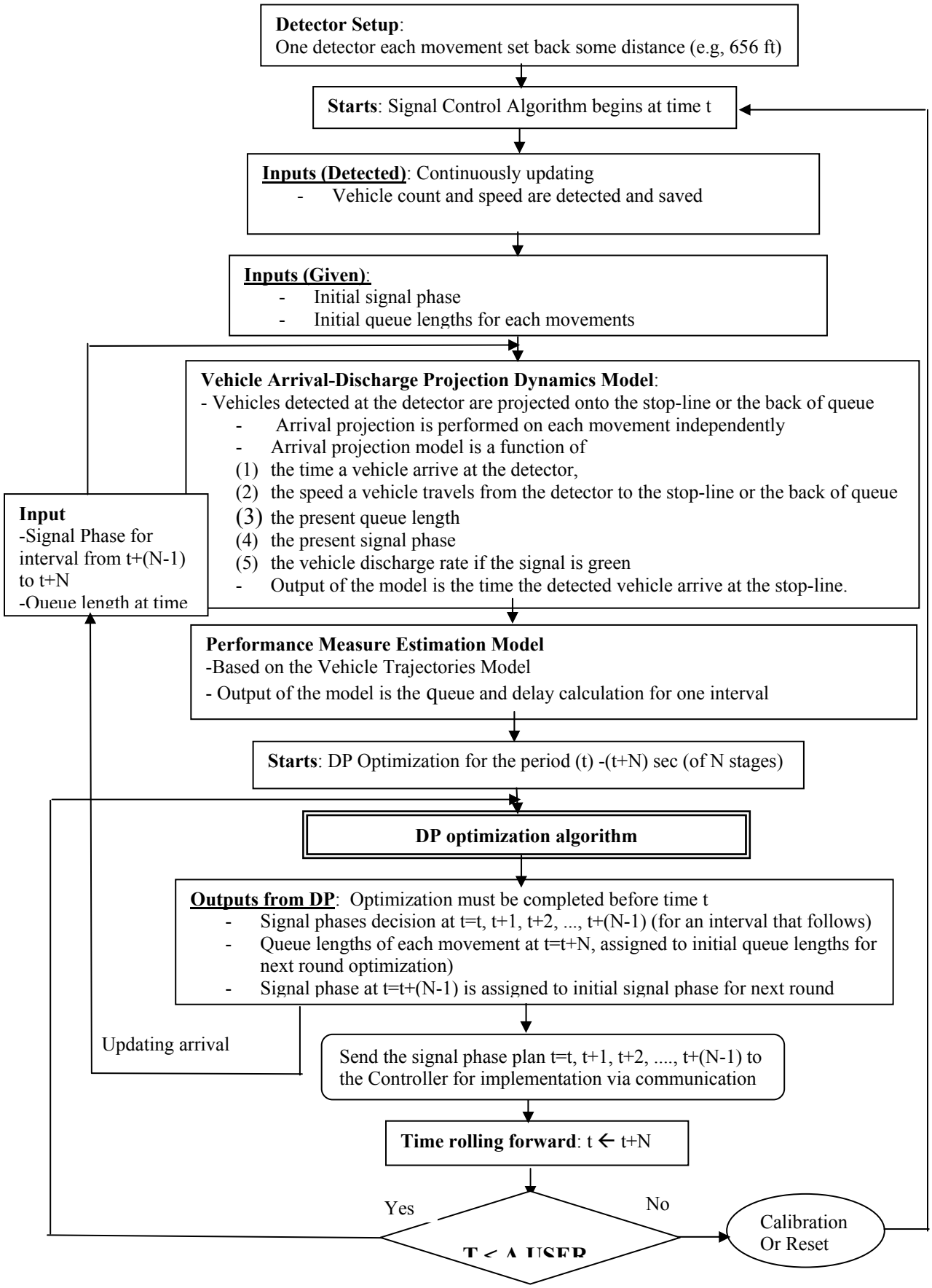


Figure 4.5 Framework of Algorithm

CHAPTER 5
USING SIMULATION TO EVALUATE THE DP ALGORITHM

Simulation is a valuable tool for the evaluation of various traffic improvement scenarios in the transportation system and is used here to assess the adaptive signal control algorithm developed. This chapter is organized in two parts. First, section 5.1 presents a procedure for using simulation to evaluate different signal plans obtained from the proposed DP algorithm and other optimization models. The second part of the chapter (Section 5.2) discusses the simulation of the DP algorithm using AIMSUN. Each subsection of Section 5.1 describes one aspect of the evaluation procedure. Section 5.1.1 states a general procedure of simulation evaluation. Section 5.1.2 describes the selection of an appropriate simulation model, where the capabilities of major traffic simulation models (AIMSUN, CORSIM, and VISSIM) are reviewed and the reasons of selecting AIMSUN are discussed. Section 5.1.3 presents the calibration of the selected simulation model and the field data of selected field diamond interchange. Section 5.1.4 addresses alternative signal optimization models. This is followed by a discussion on the methods for simulating the proposed DP algorithm in Section 5.1.5. In section 5.2, the function of GETRAM extension module is firstly introduced, DP implementation scheme including detectors capabilities and vehicle projection is described next and the flow chart of how the DP is coded along with time rolling is discussed finally.

5.1 Using Simulation for Evaluation

5.1.1 Simulation Evaluation Procedure

The optimal signal control algorithm developed in this study is different from the traditional plan concept. The simulation of diamond interchange operation provides a cost-effective way to evaluate the performance of this new signal plan strategy before it is implemented. Several optimized signal plans are evaluated in the same environment and their results are compared accordingly. As illustrated in Figure 5.1, a procedure for conducting this simulation evaluation study is proposed in this research. First, a

microscopic simulation model that can simulate networks and adaptive signal control is selected as an evaluation model. Secondly, the selected model is calibrated using diamond interchange field data. The calibrated diamond interchange is then used to compare the performance of three different optimal signal plans under the same scenarios. Signal plans are obtained from three signal optimization models: PASSER III, TRANSYT-7F and the DP algorithm developed in this study. The values of the calibrated parameters are used in the comparison study, by assuming that driver behavior, vehicle characteristics, simulation seed numbers, etc. keep the same for all comparison cases. Finally, this study compares delays of each movement and entire diamond interchange from three signal optimization algorithms, namely, 1) PASSER III; 2) TRANSYT-7F, and 3) proposed DP algorithm.

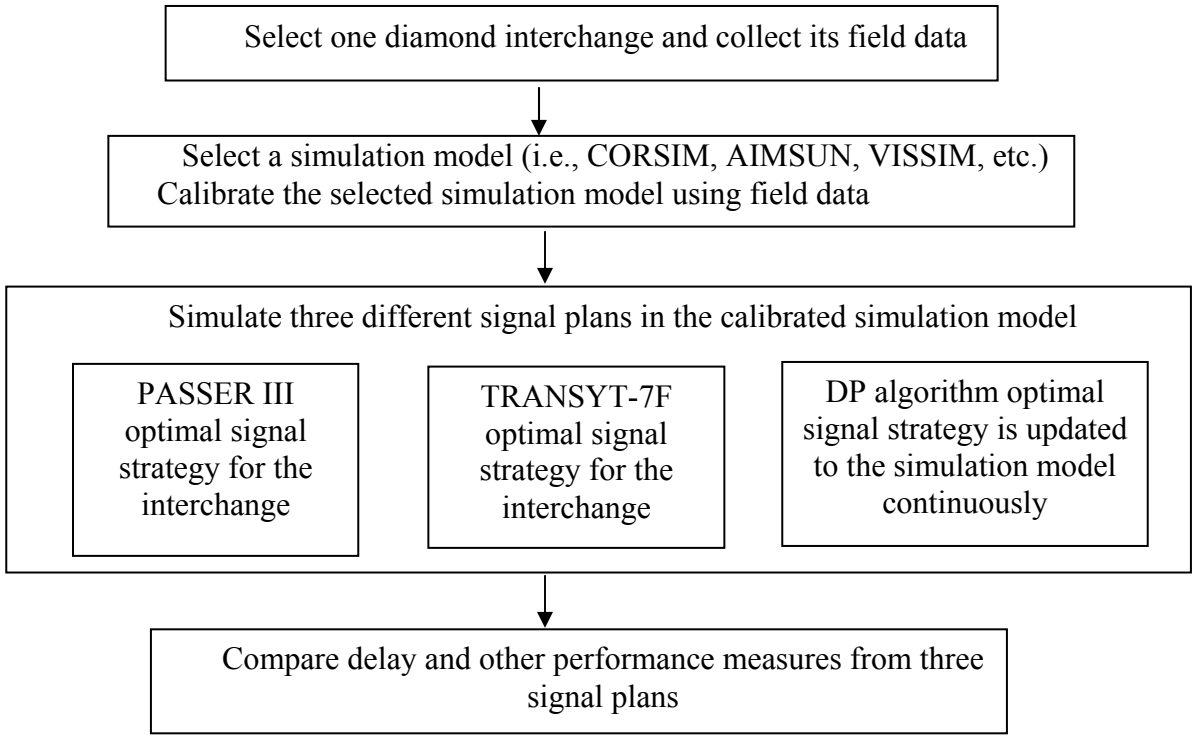


Figure 5.1 Simulation Evaluation Procedure

5.1.2 Selection of a Simulation Model

One of three widely used traffic simulation models – CORSIM (FHWA, 1997), AIMSUN (TSS, 2002) or VISSIM (PTV, 2000) is selected as an evaluation model after the capabilities of each model are examined. All three simulators can simulate arterial and freeway operations, and a wide variety of traffic control and signalization functions. The specific characteristics of each model and its application are briefly described as follows:

AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks) is a microscopic traffic simulator that evaluates the full range of functionally classified roadways, such as freeways, urban streets, ring roads and any combination of thereof. The model, based on Gipps car-following logic, provides detailed representation of traffic networks. It deals with various types of traffic controls (e.g., signs, pre-time, actuated, adaptive and ramp metering) and measurable traffic detector (e.g., counts, occupancy, presence, speed and density), analyze a wide range of geometries, vehicles and drivers, and models public transportation, incidents and VMS. AIMSUN is also designed to simulate advanced traffic management systems such as adaptive signal control, bus priority, vehicle guidance systems, etc. through its GETRAM Extensions function. AIMSUN was developed by TSS (Transportation Simulation Systems) in Spain. Networks in AIMSUN are constructed by a user-friendly TEDI (Traffic Network Graphic Editor) while simulation is executed in the GETRAM Environment (Generic Environment for Traffic Analysis and Modeling). AIMSUN provides statistical output for flows, speeds, travel times, delay, queue, etc.

CORSIM (CORridor SIMulation) is a microscopic simulation model designed to study traffic operations of freeways, surface streets and basic transit systems. It was developed and updated by the Federal Highway Administration (FHWA) in the middle of 1970s. CORSIM consists of two models: FRESIM and NETSIM. FRSIM is a model for freeway operations while NETSIM is for street networks. CORSIM networks are based on a link-node representation. Each link represents a one-directional roadway segment while nodes represent intersections and entry/exit points. CORSIM is capable of simulating different type of traffic controls (e.g., signs, pre-time and actuated signals),

various approach geometry with maximum 7 lanes per approach, and four different categories of vehicles (e.g., passenger cars, trucks, mass transit, and carpool vehicles). CORSIM provides a variety of numerical output that is link specific, aggregated for multiple links, and network wide. The running of CORSIM is supported by TSIS (Traffic Software Integrated Shell) that provides an interface and environment.

VISSIM serves as a microscopic simulation model developed by PTV AG, Germany and is capable of analyzing surface streets, freeways and public transportation operations. Different from CORSIM and AIMSUN and other typical microscopic simulation model, VISSIM employs links and connector to form its network. This structure allows flexibility and greater representation of actual network condition, and provides the capability of grouping signal components in different intersections/locations into once controller. VISSIM can handle various traffic controls (e.g., signs, pre-time and actuated). However, actuated control has to be modeled externally using a special signal state generator that is usually coded by VAP program. Other capabilities of VISSIM include the comprehensive and detailed representation of vehicle characteristics, static and dynamic traffic assignment and various traffic flow conditions. VISSIM provides a wide range of user-defined outputs file, such as volume, speed, delay, queues and signal switch information, etc.

To select a simulation model for evaluation purposes, it is necessary to exam the characteristics, functions and flexibilities of each candidate model, with special respect to the potential being used to simulate the proposed adaptive signal control. AIMSUN has been selected as the simulation model in this study because it provides a function, GETRAM Extension module, to implement advanced traffic control applications. The module is an API (Application Programming Interface) that enables users to develop external applications that may need access to internal data of AIMSUN during simulation run time. The user can program them either in C/C++, as a DLL library, or using Python scripting language. The extension module can be used to develop an interface to other external control applications (e.g., SCOOT) or any other real-time adaptive traffic control system.

5.1.3 Calibration of the Selected Simulation Model

The selected simulation model has been calibrated using field data to ensure that it can accurately replicate field conditions. Calibration is the process of quantifying model parameters using real-world data in the model logic so that the model can realistically represent the traffic environment being analyzed (Elefteriadou et al, 1997). The traffic simulation model usually provides default values of some parameters that represent average conditions. These parameters describe vehicles, drivers and traffic model (i.e., car-following, lane-change) characteristics. Rather than accepting these default values, the calibration process adjusts the model by quantifying these default values with site-specific data to the extent practical. In other words, with a given input data set (such as geometry and volumes), the calibrated model outputs the same performance measures (such as delay and queue length) results as field observation.

The calibration of the diamond interchange for evaluating the DP algorithm and conducting comparison study in this study is based on the field data of a site obtained from Arizona DOT: Diamond Interchange – Indian/I-17, Arizona. The data available include a video-images of the vehicle arrivals and others that are classified into four types: 1) geometric layout as shown in Figure 5.2; 2) signal timing plan; 3) traffic demand; and 4) other features. The field measure of control delay for each movement is also provided. The detailed data are assembled as follows,

The average free-flow speeds were obtained from videotapes of the study sites. It was also determined from the videotapes that the diamond interchange had approximately 2 percent heavy vehicles. The diamond interchange is operated by a dual-ring actuated NEMA controller. The signal plan follows a Lead-Lead three-phase signal scheme. Arterial through movement phase has longer duration of minimum and maximum green time than other phases. Pedestrian signal phase can be recalled during arterial and ramp phases at the interchange. Dual entry is activated in left turn phase for the diamond interchange.

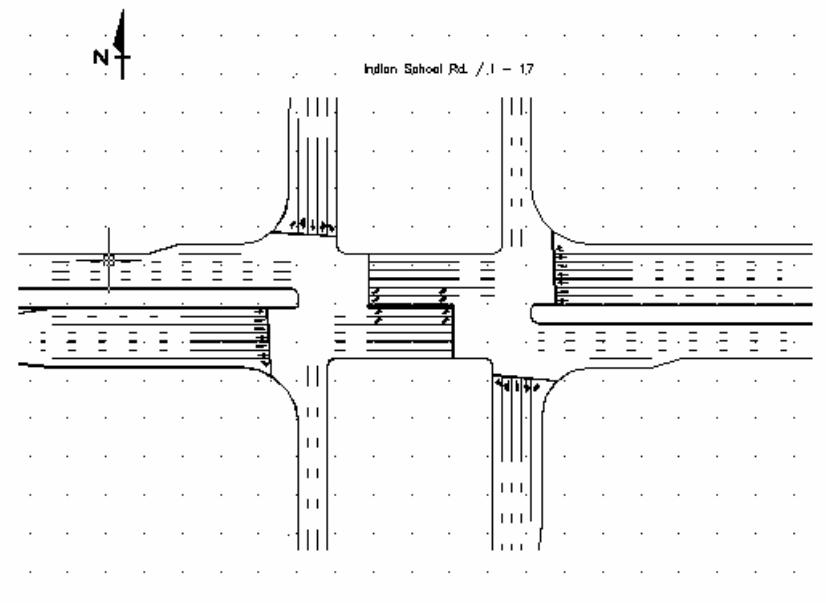


Figure 5.2 Geometric Layout for the Diamond interchange (Indian/I-17, Arizona)

Table 5.1 also presents an overview of data requirements for calibrating the model and simulating optimal signal plans.

Table 5.1 Data Requirements for Calibrating the Model and Simulating Optimal Signal Plans

Geometric Characteristics	- Layout - Number of lanes - Lane widths - Pocket length - Grades - Lane use and Lane restriction
Signal Timing	- Cycle length - Phase duration: green, yellow, red - Other parameters for actuated: max and min recall, max and min green, min vehicle extension, etc. - Detector Information – type, location, length, etc.
Traffic Demand	- Traffic volume- PHF, peak 15 minutes volume - Free-flow speed for passenger car and heavy vehicle in all approaches - Percentage of heavy vehicle - Saturation flow rate
Other Features	- Vehicles – acceleration and deceleration - Driver – probability of left turn lagger, probability of joining spillback

The following presents the parameters modified from their default values for calibration purpose:

Driver's Reaction Time: 0.5s, (default: 0.75s)

This is the time a driver takes to react to speed changes in the preceding vehicle. It is used in the car-following model. The calibration desires to use driver's reaction time of 1.5s. However, as driver's reaction time is also taken as the simulation step or cycle, this study uses a 0.5s of the simulation step to increase the accuracy of obtaining traffic information for simulating the DP algorithm. In addition, vehicle information can be updated in each 0.5s for a DP interval of 2.5s rather than fractional data are retrieved.

Response Time at Stop: 0.75s, (default: 1s)

This is the time taken for a stopped vehicle to react to the acceleration of the vehicle in front, or to a traffic light changing to green.

Vehicle Maximum Desired Speeds**:**

This set of inputs defines the distributions of the maximum speeds for each type of vehicle (i.e., passenger cars, trucks, etc.) present in the traffic stream. Decreasing these speeds results in a decrease of travel time delay. The distribution of maximum desired speeds for passenger cars is described as follows:

Mean: 32.5mph
Deviation: 2.5 mph
Minimum: 30mph
Maximum: 35mph

Vehicle Maximum Acceleration:

This set of inputs defines distributions of the maximum acceleration that each vehicle type can achieve under any circumstance. This acceleration is used in the Gipps car-following model that AIMSUN employs. The distribution of maximum acceleration for passenger cars is entered as follows:

Mean: 7 ft/s^2 (default: 9.19 ft/s^2)
Deviation: 0 ft/s^2 (default: 0 ft/s^2)
Minimum Value: 7 ft/s^2 (default: 9.19 ft/s^2)
Maximum Value: 7 ft/s^2 (default: 9.19 ft/s^2)

Vehicle Speed Limit Acceptance Parameter:

This parameter (>0) can be interpreted as the "level of goodness" of the drivers or the degree of acceptance of speed limits. >1 means that the vehicle will be assigned as maximum speed for a section a value greater than the speed limit while <1 means that the vehicle will be assigned a lower speed than the speed limit. The distribution of the vehicle speed acceptance parameter for passenger cars is modified as follows:

Mean: 1.1, (default: 1)
Deviation: 0.1, (default: 0)
Minimum: 1, (default: 1)
Maximum: 1.2, (default: 1)

Speed Limit of Links (Sections)

It is the maximum speed allowed for the vehicles traveling through a section. Depending on the value of the Vehicle Speed Limit Acceptance Parameter, each vehicle may or may not follow the speed limit recommendations. The values entered are as follows for both the SPUI and the diamond:

Arterials: 35mph (default: 31mph)
Ramp: 35 mph (default: 37 mph)

Speed Limit of Turning:

It is the maximum speed at which a vehicle will travel when making a turn. Depending on the value of the Vehicle Speed Limit Acceptance Parameter, each vehicle may or may not follow the speed limit recommendations. The values entered are as follows for both the SPUI and the diamond:

Arterials:	Through: 45 mph	Right turning: 40mph	Left turning: 40mph
Ramps:	Through: 35mph	Right turning: 40mph	Left turning: 35mph

Defaults:

Arterials:	Trough: 31 mph	Right turning: 16mph	Left turning: 10mph
Ramps:	Through: 37mph	Right turning: 23mph	Left turning: 34mph

5.1.4 Signal Timing Optimization Models for Diamond Interchanges

The models that can provide the signal timing optimization include PASSER IV, TRANSYT-7F and PASSER III. Their capabilities and limitations are reviewed as follows.

PASSER IV is a deterministic model for optimizing arterial signal timing. It is based on bandwidth efficiency optimization to select the best offsets, phase duration and cycle length. The Webster model is used to optimize the phase splits with searching the minimum delay for arterial intersections. The limitations of PASSER IV used as a simulation tool for diamond interchanges include: no explicit consideration is given to spillback. Left turn lanes have same length as through lanes. Bandwidth optimization can result in cycle lengths that can cause high degree of saturation and delay.

TRANSYT-7F is a macroscopic, deterministic model for optimizing arterial and network signal timing. Compared to PASSER IV, the bandwidth is unimportant for TRANSYT-7F, instead, TRANSYT-7F uses the "disutility index" which is linear function of the delay and stops, or excess fuel consumption or others to optimize cycle length, offsets and phase duration. A hill-climbing search technique is applied to search the minimum/maximum objective function. This is an iterative, gradient search technique that requires extensive numerical computations by the computer. The disadvantages of TRANSYT-7F include: it does not consider spillback/oversaturation explicitly. Optimization based solely on "Disutility" may result in poor progression. Optimization based on static flows that do not vary over time. Phase sequence is not optimized. The "hill-climbing" technique cannot guarantee that the global optimum will always be found.

PASSER III is the only existing model designed for evaluating and optimizing pre-timed signal timings at diamond interchange. Interchange performance is evaluated

by using average vehicle delay based on the combination of Webster's equation for external delay and delay-offset technique for internal delay. As stated in Section 2.2.1, PASSER III can only search for a signal plan or a value of the signal parameter with the minimum delay from the restricted alternative. Every search focuses on one parameter such as cycle length or phase duration.

The capabilities and performance measures outputs of the signal optimization models are summarized in Table 5.2. After studying each model's characteristics, this research considered the optimal signal results from PASSER III and TRANSYT-7F as the comparison alternatives. PASSER IV is not included because it develops signal timing by maximizing progression on an arterial; therefore, the minimum delay may not be attained. Furthermore, PASSER IV's signal plan is not appropriate for a diamond interchange because its operation cannot be simply recognized as two regular signalized intersections.

Table 5.2 The Capabilities of Signal Optimization Models

(Excerpt from: Elefteriadou et al, 2002)

	PASSER III	PASSER IV	TRANSYT-7F
Diamond Interchange	Yes (Specifically for diamond interchange)	Yes	Yes
Signal Timing Optimization	Offset; Phase duration and Cycle length	Offset, Cycle length and Phase duration based on Bandwidth	Cycle length, Offset and phase duration
Delay Definition	Based on the HCM 1985 stopped delay equation, the delay (called as Total Delay: sec/veh by PASSER III) is calculated as 1.3 Stopped Delay.	Based on the HCM 1985 stopped delay equation, the delay (sec/veh) is calculated as 1.3 Stopped Delay.	Based on HCM 1994 while considering the variations of vehicles arrivals from cycle to cycle.
Queue	Maximum Queue Length (veh/ln)	Maximum Queue length is estimated from the "queue profile" documented in several TRB papers. A "queue profile" is a process which uses a complex algorithm that identifies 15 distinct "events" in the cycle which define timing and queue formation and dissipation.	Maximum back of queue is defined in link-wise simulation to represent the maximum queue length in the uniform arrival distance-time diagram. It doesn't consider the queue spillback.

5.1.5 Simulation of the Proposed DP Algorithm

When a traffic simulation package is used to evaluate the DP optimal signal control algorithm, as shown in Figure 5.3, an interface connecting from the simulation software to the algorithm is necessary. The connection has to allow the DP algorithm to optimize signal timing in real time on user-defined simulated network provided by the simulation package. It is the major advantage of testing the proposed algorithm in the simulated network since the simulation software is able to reproduce the real traffic and geometric conditions of traffic network on a computer. However, the possibility of accessing traffic simulation internal data structure to make the successful connection needs to be investigated. The University of Utah traffic laboratory has explored the performance of SCOOT real-time adaptive control in a CORSIM environment (Hansen, Martin and Perrin, 2000). A software called SCOOT-CORSIM was developed to support the detector data transfer and the signal decision communication. Since SCOOT uses entirely different concept from our DP algorithm, the possibility of using CORSIM for our algorithm needs to be further investigated.

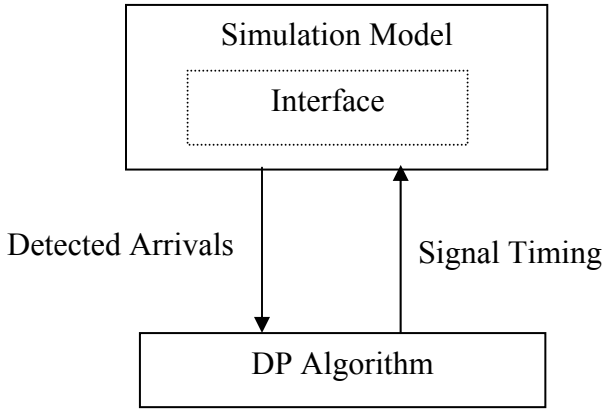


Figure 5.3 Connection from Simulation Model to the DP Algorithm

In addition to the software link, another alternative is the Controller Interface Device (CID) that provides the link between simulation software and actual traffic controller. CID can react to the detector actuation from the simulation model by updating phase indications in the controller, then these phase indications are subsequently read back from the controller hardware to the simulation model and assigned to the simulation traffic signal. For the application of CID in this study, whether our DP based algorithm

can be programmed into CID and controller to carry out all these connections still remains to be investigated.

After carefully investigating possible tools to evaluate the DP algorithm, as discussed in Section 5.1.2, AIMSUN has been selected as the simulation model in this study because its unique API (Application Programming Interface) function named as GETRAM extension module can be used to implement advanced traffic control applications.

5.2 Simulation of the DP Algorithm in AIMSUN

5.2.1 GETRAM Extension Module

The simulation of the DP algorithm is carried out by the GETRAM extension module embedded in AIMSUN. The module serves as an interface to link AIMSUN to an external application (i.e., DP algorithm) that requires to access to some internal data of AIMSUN such as vehicle information during simulation run time. Figure 5.4 shows the process of the information exchange between AIMSUN and the DP algorithm via GETRAM extension module. First, the simulated interchange network equipped with detectors is modeled in AIMSUN. Next, during simulation, the real-time traffic measurements provided by those detectors feed the DP algorithm that, after processing, makes the decision of signal controls, e.g., implement phase 26 for next 7.5s. Finally, these decisions are transferred back to the simulated network, which emulates their operations through the signal controller.

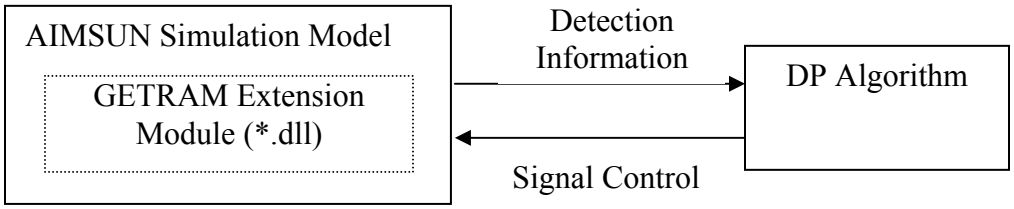


Figure 5.4 Simulation of the DP Algorithm in AIMSUN via GETRAM Extension Module

In order to achieve this exchange between the simulator and the DP algorithm, and have the network dynamically implementing the DP signal plan, the external algorithm is required to be compiled into a DLL (dynamic link library) file. In this research, Microsoft Visual C++6.0 is used to code the algorithm by combining the functions available from GETRAM extension module. The module has a number of functions available for users to obtain detection information from AIMSUN model, such as the number of vehicles that have crossed a detector during a simulation step or a aggregated time interval, vehicle speed, and detector occupancy, etc.

5.2.2 DP Algorithm Simulation Scheme

To implement the DP algorithm in AIMSUN, as stated in Section 4.1, detectors are placed 656 ft upstream of the stop-line for each approach. These detectors can generate four types of vehicle data at every simulation step. They are:

- Count: number of vehicles
- Speed: mean speed for vehicles crossing the detector during the step
- Occupancy: percentage of time step the detector is pressed
- Presence: whether a vehicle is over the detector or not

Among them, count and speed are the required data gathered for implementing the DP algorithm.

A simulation step of 0.5s is used to collect vehicle detection information because smaller values means more accurate detected data, and a DP interval 2.5s is 5 (an integer) times 0.5s. As shown in Figure 5.5, detection data are collected in each simulation step of 0.5s for extreme detection range.

The simulation time horizon is arranged so that the simulation starts at time 0 and the interchange is controlled in a fixed time signal control, when the time goes to 300 seconds (5 minutes), the signal switches to adaptive signal control. Figure 5.5 also shows the simulation scheme during this transition period. The first DP horizon is located at time 300 to 310s. Its corresponding detection period is 16 to 2 seconds in advance, i.e.,

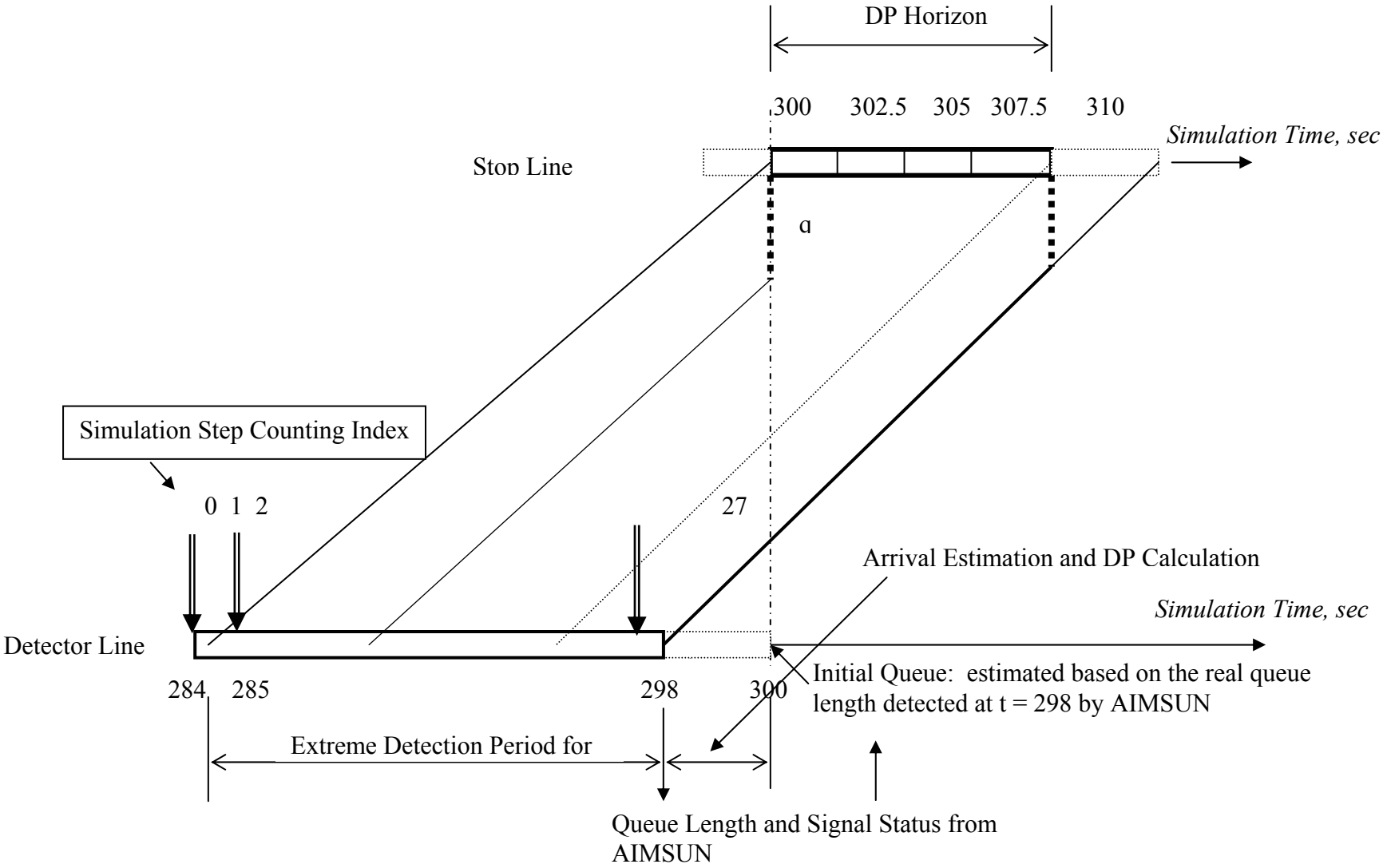


Figure 5.5 Arrival Detection and DP Calculation Timing Scheme

the maximum detection time is 284 to 298 seconds. The arrival projection and DP calculation starts to be processed immediately after the detection period, but has to be completed before the time 300. Initial queue length and signal status for the beginning of DP horizon is obtained from the simulation time at 298s. Figure 5.6 describes the detection overlap for two consecutive optimization periods in terms of simulation time. The detailed relationship between optimization horizon and detection range is addressed in Section 3.3.4.

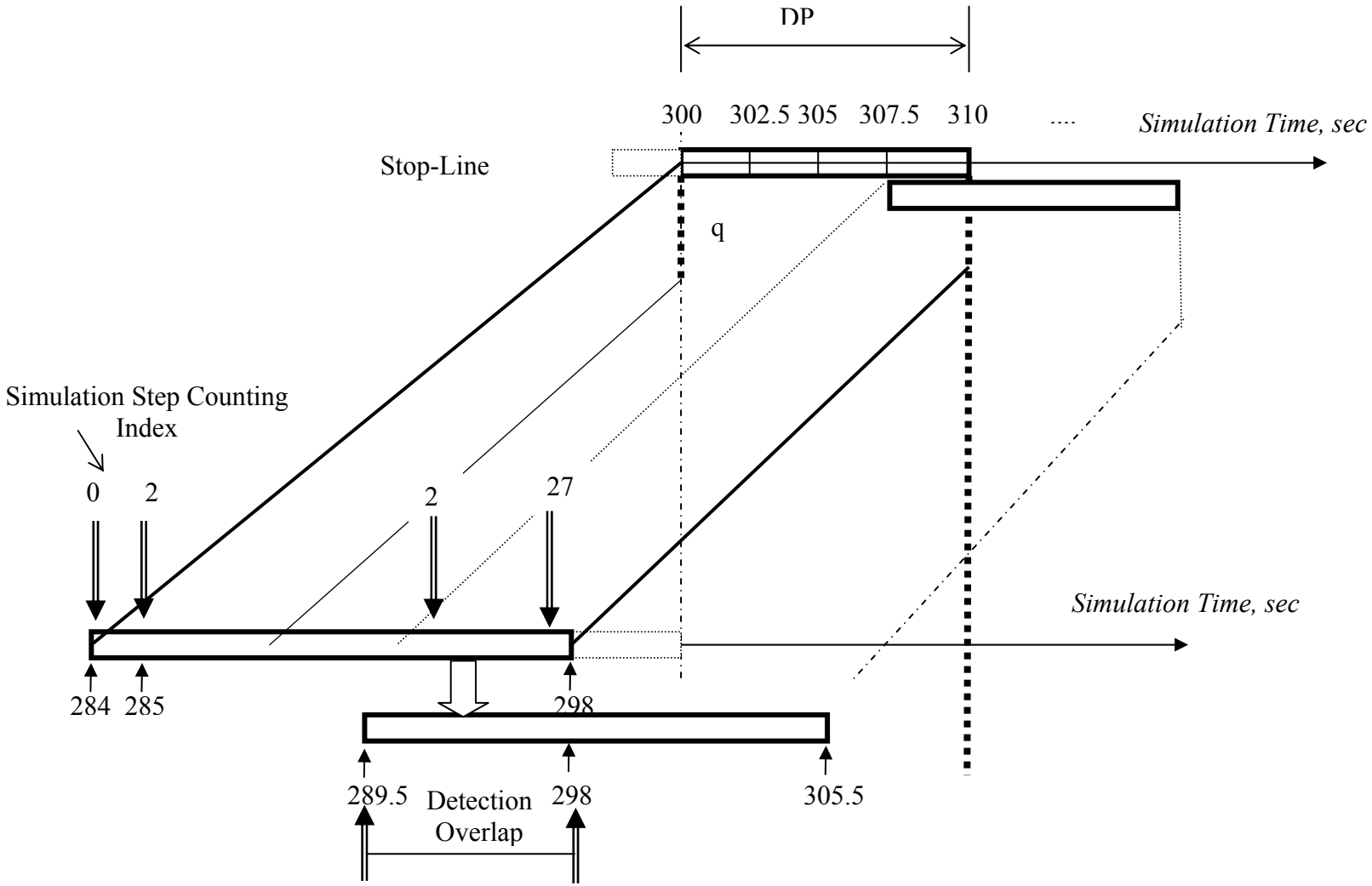


Figure 5.6 Arrival Detection and DP Horizon Rolling Forward Scheme

5.2.3 Flow Structure and Timing Logic of the DP Algorithm Coding

The DP algorithm is coded in C++ computer language to generate a dynamic link library (DLL) file that can be connected to the simulation. Figure 5.7 presents the flow chart of the algorithm along with the time rolling horizon. The DP algorithm is coded according to the procedure of six routines defined by GETRAM extension module to build the communication between AIMSUN and the DP adaptive signal control. They are GetExtLoad, GetExtInit, GetExtManage, GetExtPostManage, GetExtFinish and GetExtUnLoad. With reference to Figure 5.8, their functions are described as follows,

1. GetExtLoad:

It is called when the GETRAM extension is loaded by AIMSUN

2. GetExtInit

It is called when AIMSUN starts the simulation and initialize some parameters, such as the simulation/detection step, DP rolling horizon and DP interval counts, as well as find displays the current signal plan (fixed time control)

3. GetExtManage

It is called in every simulation stop at the beginning of the cycle. It is used to request and update detector measures, vehicle information and interaction with junctions. The DP optimization, vehicle arrivals-discharge projection model and signal implementation are embedded in this routine.

4. GetExtPostManage

It is called in every simulation step at the end of the cycle.

5. GetExtFinish

It is called when AIMSUN finishes the simulation

6. GetExtUnLoad

It is called when the GETRAM extension is unloaded.

Time Logic

Among six routines, GetExtManage is the one that essentially contains the code for implementing the algorithm. The time logic of each block in GetExtManage is shown in Figure 5.7.

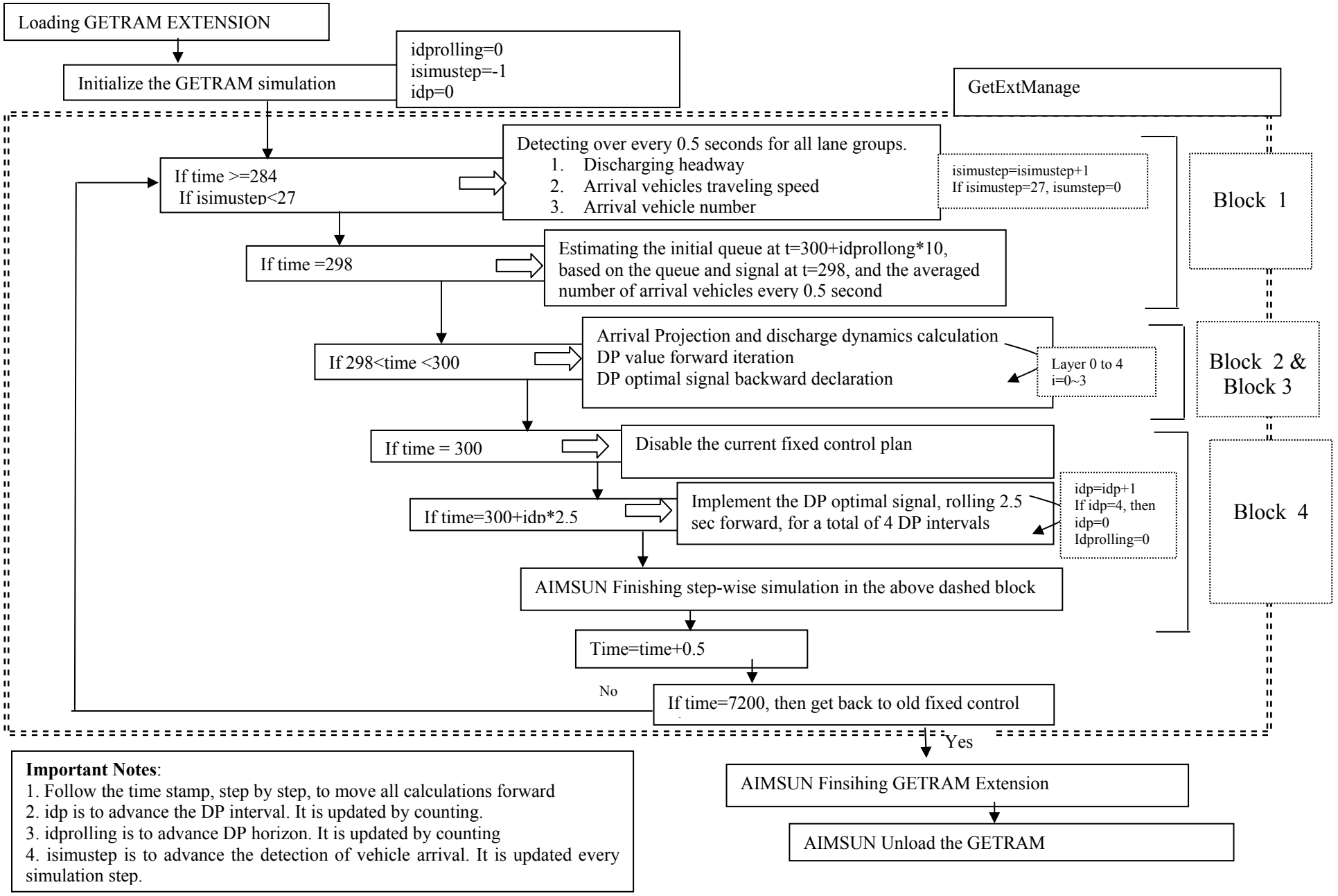


Figure 5.7 Flow Chart and Timing Logic of DP Algorithm Coding

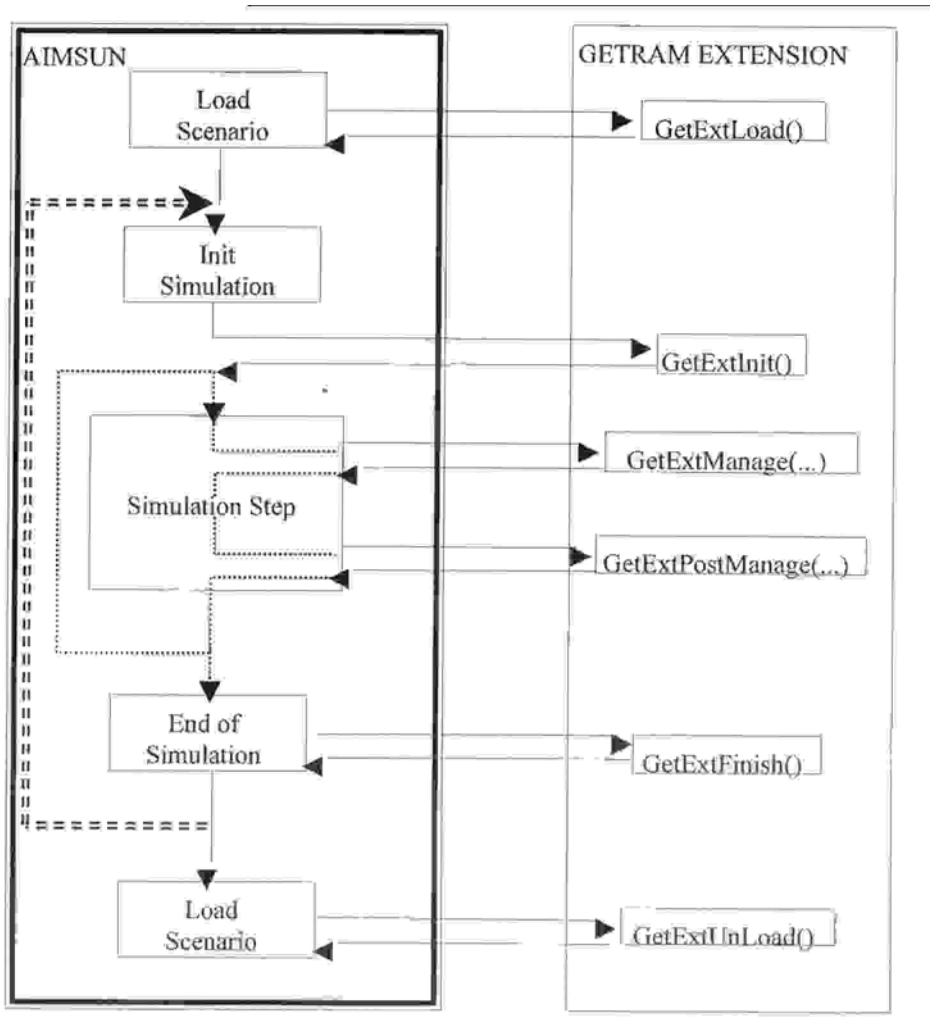


Figure 5.8 DP Algorithm in Relation to AIMSUN/GETRAM Extension (Excerpt from: TSS -GETGRAM Extension User Manual, 2002)

The simulation is arranged in such a way that when simulation time is less than 300s (5 minutes), the interchange is operated as fixed time control; when time reaches to 300s, fixed time control is disabled and DP adaptive control is applied; when 300 < Time < 7200 (2 hours), the interchange is operated as DP control and when time > = 7200 (2 hours), fixed time control is enabled. The timing of the DP signal optimization is detailed below,

When 284s <= time <= 298s

Detectors start to gather information for every simulation step 0.5 seconds for each 8 approaches including the number of vehicles crossing the detectors and vehicles traveling speed.

When time = 298
Estimate the initial queue at t = 300 which is the initial queue for optimization cycle based on the queues and signal at t = 298.

When 298 <time <300
Conduct arrival projection and discharge dynamic calculation, DP value forward iteration and DP optimal signal backward declaration

*When 300 < time < 310 (4*2.5 = 10)*
Implement the DP optimal signal by rolling 2.5 sec forward for 7.5s by adopting majority signal rolling

Time is forwarded for each 10 seconds until 2 hours when it switches back to fixed time control.

Block Structure

The code flow structure of the algorithm consists of several blocks, as presented in Figure 5.7. The following illustrates each block function:

Block One – Vehicle Detection and Initial Status

During this block, detectors start to gather information for every simulation step 0.5 second for each approach, initial queue for each DP horizon is calculated, and the signal control at time of 2 seconds in advance is used for the initial signal plan for DP calculation. If it is the first DP cycle, vehicles arriving from time 284 – 298s at detectors are saved, otherwise, the detection overlap is considered.

Block Two – Forward DP Value Calculation

During this block, the DP forward recurrence method is applied. Based on the optimal state values of the previous stage, for all nodes in current stage, the PMI (i.e., return) value for each node is calculated, and the optimal decision made at each node is obtained in current stage which results in the total PMI value minimal from the DP beginning to the current node. These optimal state values also serve as the initial PMI value of the next stage.

Block Two also includes a subroutine developed for the vehicle projection and DP value calculation over a single DP interval (2.5s). The subroutine, firstly, calculates the vehicle travel time T required to reach the back of queue at the stop-line for each approach in terms of three scenarios: a) no queue and signal green; b) no queue and signal red and c) initial queues present; Secondly, it locates each vehicle detection starting time point (T_0) at the detector line; Thirdly, it computes the number of vehicles projected without taking into account the projection interactive dynamics. This is defined as "nominal $q_{projected}$" vehicle projection (it projects 5 simulation steps of the detected vehicles into the corresponding DP interval 2.5s starting from T_0 for each approach); This step is followed by calculating the number of vehicles discharged ($q_{discharged}$). Next, $q_{projected}$ is adjusted from "nominal $q_{projected}$" by considering projection interactive dynamics as follows,

- If $q_{projected} > q_{discharged}$, the queue becomes longer, i.e., the detection line projecting point moves to the right
- If $q_{projected} < q_{discharged}$, the queue becomes shorter, i.e., the detection line projecting point moves to the left

Every added or reduced queued vehicle increases or decreases 0.5s for detection time at the detector, i.e., the projection point moves to the right or the left. Finally, the subroutines calculates different PMI including the sum of final queues, q_{final}, for all approaches, the sum of average delay per lane of all approaches, the sum of total delays and the sum of storage ratios.

Block Three – Optimal DP Signal Declaration

Block three finds the optimal phase in the last interval that will be the initial phase for a new round DP horizon and declare the optimal signals to be implemented.

Block Four – Optimal DP Signal Implementation

During this block, at time 300s, fixed time control is disabled and the DP control is implemented with the time goes on. When the time reaches to 7200s, fixed time control is switched back.

CHAPTER 6
SENSITIVITY ANALYSIS

This chapter conducts three types of sensitivity analysis to characterize the proposed adaptive signal control algorithm. First, the simulated interchange is described in Section 6.1. Section 6.2 discusses how the interchange delay is affected by different definitions of performance measure index (PMI). The performance of dynamic weights and fixed weights is also presented in this section. Section 6.3 describes how the interchange delay is affected by the weight on ramps, arterials and internal left turns, respectively. This is followed by a discussion on the performance of the DP algorithm under various demand scenarios in Section 6.4. Finally a summary of these sensitivity analyses is provided in the end of the chapter.

6.1 Overview

As shown in Figure 6.1, the field interchange network as indicated in Section 5.1.3 is used to conduct the sensitivity analysis. The DP algorithm is dynamically linked to the interchange and controls the interchange's signal operation in real time. These real-time generated signal plans vary with the optimization criteria defined in the DP algorithm. The formulation of optimization criteria is associated with the PMI and the weights assigned to each approach. Therefore, the following sections will investigate how the different PMI definitions and the weights change the interchange operational performance. The capability of how the DP algorithm accommodates various demand scenarios is also examined.

The interchange controlled by the DP adaptive signal plan is simulated in AIMSUN under various demand scenarios. Figure 6.1 also shows an example of O-D demand scenario.

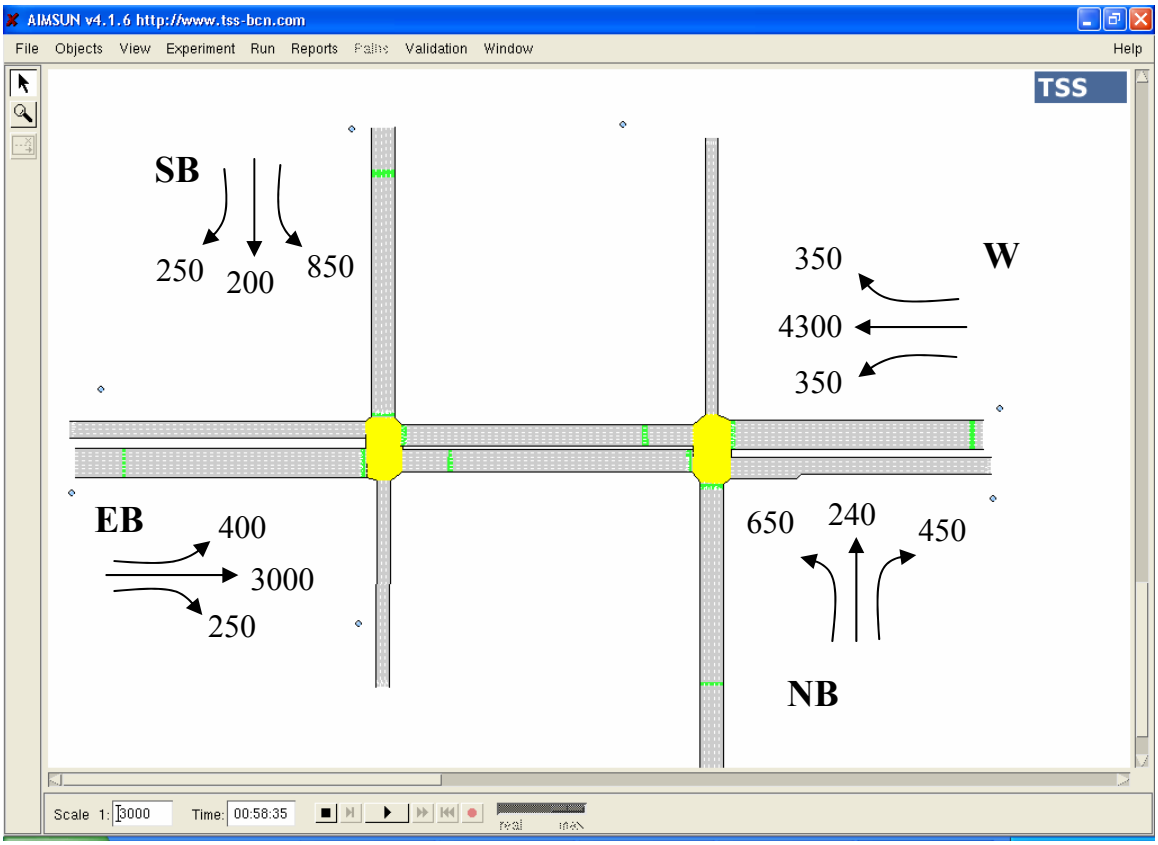


Figure 6.1 The Simulated Interchange with a O-D Demand Scenario

6.2 Delay vs. Performance Measure Index (PMI)

6.2.1 PMI Definitions

This section studies how the interchange system delay is affected by the different definitions of PMI in the DP algorithm. PMI is served as optimization criteria in the DP algorithm. Therefore, different PMI will result in different signal control plan that subsequently affects the interchange operational performance.

A PMI is defined for each interval, but calculated over the optimized decision trajectory that consists of 4 DP intervals. The result from the DP procedure is the total optimal value of the PMI over the horizon. This research has used four types of PMI that are described as follows,

PMI 1: Sum of Average Queues Per Lane For All Approaches

$$PMI = \sum_{j=1}^{4} \left(\sum_{i=1}^{8} w[i] * Qfinal[i] \right)_j \quad (6.1)$$

Where,

w[i]: The weight associated with approach *i*

Qfinal[i]: Average queue length per lane on approach *i* in the end of a DP interval *j*. It is calculated as follows,

$Qfinal[i] = Qinitial[i] + Qprojected[i] – Qdischarged[i]$

PMI 2: Sum of Average Delay Per Lane For All Approaches

$$PMI = \sum_{j=1}^{4} \left(\sum_{i=1}^{8} w[i] * Delay[i] \right)_j \quad (6.2)$$

Where,

w[i]: The weight associated with approach *i*

Delay[i]: Average delay per lane on approach *i* for total vehicles for a DP interval j. It is calculated as follows,

$Delay[i] = 2.5*Qinitial[i] + 0.5*2.5*(Qprojected[i] – Qdischarged[i])$

The delay calculation assumes that, for each DP interval of 2.5 seconds, the initial vehicles are delayed for 2.5 seconds and arrived vehicles are delayed for half of 2.5 seconds.

PMI 3: Sum of Total Delays for All Approaches

$$PMI = \sum_{j=1}^{4} \left(\sum_{i=1}^{8} NL[i] * Delay[i] \right)_j \quad (6.3)$$

Where,

NL[i]: The number of lanes of approach *i*

Delay[i]: Average delay per lane on approach *i* for total vehicles for a DP interval *j*

PMI 4: Sum of Storage Ratio for All Approaches

$$PMI = \sum_{j=1}^{4} \left(\sum_{i=1}^{8} SR[i] \right)_j \tag{6.4}$$

Where,

SR[i]: Storage ratio on approach *i* in the end of a DP interval *j*. It is calculated as follows

$$SR[i] = \frac{Q_{final}[i]}{L[i]/S[i]}$$

Where,

Qfinal[i]: Average queue length per lane on approach *i* in the end of a DP interval

L[i]: Link length of approach i

S: Average space headway of queued vehicle

6.2.2 Fixed Weights

Fixed weights mean that its value associated with an approach is pre-determined before simulation. The value will not change during the simulation.

Table 6.1 presents how each of the PMI definitions affects the system delay resulted from AIMSUN simulation for various demand scenarios. Six types of demand scenarios ranging from low demand to high demand are considered in this research. The results have shown that generally the simulated interchange experiences less delay in "PMI 1: Sum of Average Queues Per Lane For All Approaches" and "PMI 4: Sum of Storage Ratio for All Approaches" than PMI 2 and 3.

The results of PMI 1 and PMI 4 are very similar in this study due to the geometric configuration of this particular interchange. Each approach of the simulated interchange has the same link length (i.e., same maximum storages for queued vehicles). Therefore, the definition of the sum of storage ratio can be regarded the same as the sum of average queue length for this particular interchange. However, in the case of a diamond

interchange that has left turn pocket lane or each approach has different length, PMI 1 should result in different result from PMI 4. There is a discrepancy in the system delay for scenario 6 (high demand) between PMI 1 and PMI 4. This could be further developed delays under over-saturated conditions happened on some approaches.

In the following studies, the optimization criteria of " PMI 1: Sum of Average Queues Per Lane For All Approaches" will be applied to conduct sensitivity analysis and make comparisons among several optimization methods.

Table 6.1 System Delay (sec/veh) of Each Performance Measure Index (PMI) Using Fixed Weight

#	Movement Type			O-D Demand (veh/h)	Demand Level*	PMI 1. Fixed Weight Sum of Average Queues Per Lane For All Approaches	PMI 2: Fixed Weight Sum of Average Delay Per Lane for All Approaches	PMI 3: Fixed Weight Sum of Total Delays for All Approaches	PMI 4: Fixed Weight Sum of Storage Ratio for All Approaches
1	Arterial	EB/WB	L	400	H	25	31	36	26
			TH	3000					
			R	250					
	Ramp	NB/SB	L	650					
			TH	300					
			R	650					
2	Arterial	EB/WB	L	400	H	33	42	55	34
			TH	3300					
			R	250					
	Ramp	NB/SB	L	650					
			TH	300					
			R	650					
3	Arterial	EB/WB	L	200	M	16	16	17	16
			TH	2000					
			R	250					
	Ramp	NB/SB	L	500					
			TH	300					
			R	500					

(Continued in the next page)

(Table 6.1 Continued from the previous page)

4	Arterial	EB/WB	L	200	M	17	17	16	18
			TH	2500					
			R	250					
	Ramp	NB/SB	L	400					
			TH	250					
			R	400					
5	Arterial	EB/WB	L	250	L	14	15	16	15
			TH	1000					
			R	250					
	Ramp	NB/SB	L	350					
			TH	150					
			R	350					
6	Arterial	EB/WB	L	400	H	44	59	63	45
			TH	3500					
			R	250					
	Ramp	NB/SB	L	750					
			TH	300					
			R	750					

*: H, M and L represents High, Medium and Low Demand, respectively.

6.2.3 Dynamic Weights

Dynamic weights denote that the values of weights associated with each approach vary in each optimization horizon during the simulation. Dynamic weights are designed to be used in high demand (congested) and unpredicted conditions to quickly discharge queues for certain approach(es). In view of "PMI 1: Sum of Average Queues Per Lane For All Approaches" defined for the fixed weight, its corresponding dynamic weight PMI is defined as follows,

PMI 1D: Sum of Average Queues Per Lane For All Approaches using Dynamic Weight

$$PMI = \sum_{j=1}^{4}\left(\sum_{i=1}^{8} dw[i] * Qfinal[i]\right)_j \tag{6.5}$$

Where,

dw[i]: Dynamic weights associated with approach *i.*

Qfinal[i]: Average queue length per lane on approach *i* in the end of a DP interval *j*

The values of dynamic weights are determined based on the storage ratio of an approach, as shown in Table 6.2. When the storage ratio of an approach is greater than certain values (e.g., 0.6), the higher dynamic weight value (e.g., 6) will be automatically assigned to this approach to reduce queue length. Table 6.2 also presents an example where ramps and arterials have the different specification of dynamic weight values. Ramps are assigned heavier weights than other approaches under the same storage ratio. This takes into account that the DP decision network used does not provide enough ramp phase options.

Table 6.2 Dynamic Weight Values

Movement Type	Dynamic Weight Value
Arterial (EB and WB)	*if SR[i]>0.1, then dw[i]=1.0*
	if SR[i]>0.2, then dw[i]=2.0
	if SR[i]>0.3, then dw[i]=3.0
	if SR[i]>0.4, then dw[i]=4.0
	if SR[i]>0.5, then dw[i]=5.0
	if SR[i]>0.6, then dw[i]=6.0
	if SR[i]>0.7, then dw[i]=7.0
	if SR[i]>0.8, then dw[i]=8.0
	if SR[i]>0.9, then dw[i]=9.0
Ramps (NB and SB)	*if SR[i]>0.1, then dw[i]=4.0*
	if SR[i]>0.2, then dw[i]=5.0
	if SR[i]>0.3, then dw[i]=6.0
	if SR[i]>0.4, then dw[i]=7.0
	if SR[i]>0.5, then dw[i]=8.0
	if SR[i]>0.6, then dw[i]=9.0
	if SR[i]>0.7, then dw[i]=10.0
	if SR[i]>0.8, then dw[i]=11.0
	if SR[i]>0.9, then dw[i]=12.0

Several demand scenarios are designed to study the performance of using dynamic weights in the algorithm. As illustrated in Table 6.3, the demand changes in each 15-minute period in order to determine whether or not the dynamic weight has the capability to discharge the increased volume on particular approach(es) quickly enough while still keeping the operation of entire system efficiently. The scenarios include several unbalanced demands, e.g., there is one approach (i.e., arterial or ramp) with higher demand and there are two approaches (i.e., two arterials, or one arterial with one ramp) with higher demand.

Table 6.3 O-D Demand Scenarios for a Comparison between Dynamic Weight and Fixed Weight

#	Scenario	Movement Type			Demand (veh/h)				
					0-15m	15-30m	30-45m	45-60m	Total (1h)
1	High EB Demand (* showing bold)	Ramp	NB	L	120	120	120	120	480
				TH	50	50	50	50	200
				R	120	120	120	120	480
			SB	L	120	120	120	120	480
				TH	50	50	50	50	200
				R	120	120	120	120	480
		Arterial	EB	L	65	65	65	65	260
				TH	**700**	**1200**	**1300**	**1200**	**4400**
				R	60	60	60	60	240
			WB	L	65	65	65	65	260
				TH	500	500	500	500	2000
				R	60	60	60	60	240
2	High SB Demand (* showing bold)	Ramp	NB	L	120	120	120	120	480
				TH	50	50	50	50	200
				R	120	120	120	120	480
			SB	L	**200**	**400**	**350**	**350**	**1300**
				TH	80	80	80	80	320
				R	**150**	**200**	**200**	**200**	**750**
		Arterial	EB	L	65	65	65	65	260
				TH	600	600	600	600	2400
				R	60	60	60	60	240
			WB	L	65	65	65	65	260
				TH	600	600	600	600	2400
				R	60	60	60	60	240

(Continued in the next page)

(Table 6.3 Continued from the previous page)

#	Scenario	Movement Type			Demand (veh/h)				
					0-15m	15-30m	30-45m	45-60m	Total (1h)
3	High EB & WB Demand (* showing bold)	Ramp	NB	L	120	120	120	120	480
				TH	50	50	50	50	200
				R	120	120	120	120	480
			SB	L	120	120	120	120	480
				TH	50	50	50	50	200
				R	120	120	120	120	480
		Arterial	EB	L	65	65	65	65	260
				TH	**700**	**1200**	**1300**	**1200**	**4400**
				R	60	60	60	60	240
			WB	L	65	65	65	65	260
				TH	**700**	**1200**	**1300**	**1200**	**4400**
				R	60	60	60	60	240
4	High EB & SB Demand (* showing bold)	Ramp	NB	L	120	120	120	120	480
				TH	50	50	50	50	200
				R	120	120	120	120	480
			SB	L	**200**	**200**	**200**	**200**	**800**
				TH	80	80	80	80	320
				R	**150**	**200**	**200**	**150**	**700**
		Arterial	EB	L	65	65	65	65	260
				TH	**700**	**1000**	**1000**	**1000**	**3700**
				R	60	60	60	60	240
			WB	L	65	65	65	65	260
				TH	500	500	500	500	2000
				R	60	60	60	60	240

(Continued in the next page)

(Table 6.3 Continued from the previous page)

#	Scenario	Movement Type			Demand (veh/h)				
					0-15m	15-30m	30-45m	45-60m	Total (1h)
5	High EB & NB Demand (* showing bold)	Ramp	NB	L	**120**	**250**	**200**	**200**	**770**
				TH	80	80	80	80	320
				R	**120**	**250**	**200**	**200**	**770**
			SB	L	120	120	120	120	480
				TH	50	50	50	50	200
				R	120	120	120	120	480
		Arterial	EB	L	65	65	65	65	260
				TH	**700**	**1000**	**1000**	**1000**	**3700**
				R	60	60	60	60	240
			WB	L	65	65	65	65	260
				TH	500	500	500	500	2000
				R	60	60	60	60	240

* Bold number in the table represents that the corresponding movement has high demand.

Table 6.4 presents a comparison of the delay of each movement type and whole system between PMI 1D and PMI 1. Both of them are defined as "Sum of Average Queues Per Lane For All Approaches" while PMI 1D uses dynamic weight and PMI 1 employs fixed weight.

The results have shown that the dynamic weight can significantly reduce system delay than the fixed weight does in all demand scenarios. The saving on system delay by using dynamic weight is ranged between 36 percent and 49 percent (i.e., 22s, 13s, 38s, 22s and 19s, respectively). When demand varies in each 15-minute period and unpredictable, dynamic weights can automatically assign different values of weights depending on queue storage ratio on each approach using the criteria defined in Table 6.2. The application of dynamic weight makes the DP algorithm more flexible and adaptive to demand fluctuation.

It should be noted, however, that the lower system delay does not mean that the delay on all approaches is lower. For example, in the case of both EB and WB having high demand, the signal control using dynamic weight results in higher delays on ramps than the one using fixed weight does. But, because the calculation of system delay in AISMUN is averaged to each vehicle, lower volume on the ramp has less contribution to system delay than higher volume on the arterials. Therefore, high ramp delays could still lead to smaller system delay.

The simulation results have demonstrated that dynamic weights works better than fixed weights in terms of discharging vehicles on the high volume approach(es), although the former could still lead to high delays on some other approaches. Nevertheless, by properly adjusting the values of fixed weights, the signal control might also produce an equivalent or even better performance. In the remaining studies of the thesis, the general rules of choosing weights related to the interchange delay will be investigated. The signal controls using dynamic and fixed weights will be tested and compared in various demand scenarios.

Table 6.4 A Comparison of Delay (sec/veh) between Using Dynamic Weight and Fixed Weight

#	Scenario				**PMI 1D: Dynamic Weights** Sum of Average Queues Per Lane For All Approaches	PMI 1. Fixed Weights Sum of Average Queues Per Lane For All Approaches
1	High EB Demand	Ramp	NB	L	0:00:56	0:00:19
				TH	0:00:41	0:00:16
				R	0:00:57	0:00:17
			SB	L	0:01:09	0:00:52
				TH	0:00:38	0:00:17
				R	0:00:43	0:00:12
		Arterial	EB	L	0:00:29	0:00:14
				TH-EXT	0:00:20	0:01:43
				TH-INT	0:00:05	0:00:10
				R	0:00:01	0:00:13
			WB	L	0:01:45	0:00:50
				TH-EXT	0:00:11	0:00:15
				TH-INT	0:00:02	0:00:03
				R	0:00:00	0:00:00
		System Delay			**0:00:30**	**0:00:52**

(Continued in the next page)

(Table 6.4 Continued from the previous page)

2	High SB Demand	Ramp	NB	L	0:00:18	0:00:11
				TH	0:00:18	0:00:11
				R	0:00:20	0:00:11
			SB	L	0:00:47	0:02:02
				TH	0:00:21	0:00:28
				R	0:00:16	0:00:14
		Arterial	EB	L	0:00:13	0:00:20
				TH-EXT	0:00:18	0:00:34
				TH-INT	0:00:06	0:00:10
				R	0:00:00	0:00:00
			WB	L	0:00:14	0:00:37
				TH-EXT	0:00:18	0:00:23
				TH-INT	0:00:04	0:00:06
				R	0:00:00	0:00:00
		System Delay			**0:00:23**	**0:00:36**
3	High EB & WB Demand	Ramp	NB	L	0:03:33	0:02:02
				TH	0:01:00	0:00:15
				R	0:01:28	0:00:17
			SB	L	0:01:46	0:00:53
				TH	0:01:01	0:00:14
				R	0:01:19	0:00:09
		Arterial	EB	L	0:01:26	0:01:34
				TH-EXT	0:00:19	0:01:44
				TH-INT	0:00:05	0:00:12
				R	0:00:01	0:00:12
			WB	L	0:01:37	0:01:35
				TH-EXT	0:00:18	0:01:27
				TH-INT	0:00:04	0:00:10
				R	0:00:01	0:00:14
		System Delay			**0:00:39**	**0:01:17**

(Table 6.4 Continued from the previous page)

4	High EB & SB Demand	Ramp	NB	L	0:00:26	0:00:15
				TH	0:00:26	0:00:14
				R	0:00:29	0:00:14
			SB	L	0:00:59	0:00:47
				TH	0:00:29	0:00:17
				R	0:00:31	0:00:12
		Arterial	EB	L	0:00:13	0:00:19
				TH-EXT	0:00:32	0:01:44
				TH-INT	0:00:05	0:00:12
				R	0:00:01	0:00:11
			WB	L	0:00:28	0:01:25
				TH-EXT	0:00:16	0:00:17
				TH-INT	0:00:02	0:00:05
				R	0:00:00	0:00:00
		System Delay			**0:00:29**	**0:00:51**
5	High EB & NB Demand	Ramp	NB	L	0:00:40	0:00:26
				TH	0:00:29	0:00:17
				R	0:00:50	0:00:25
			SB	L	0:00:31	0:00:30
				TH	0:00:25	0:00:16
				R	0:00:22	0:00:10
		Arterial	EB	L	0:00:21	0:00:19
				TH-EXT	0:00:30	0:01:32
				TH-INT	0:00:05	0:00:15
				R	0:00:01	0:00:09
			WB	L	0:00:28	0:01:25
				TH-EXT	0:00:14	0:00:17
				TH-INT	0:00:02	0:00:05
				R	0:00:00	0:00:00
		System Delay			**0:00:29**	**0:00:48**

6.3 Delay vs. Weights

The effects of ramp weights, arterial weights and internal left turn weights are studied in this section, respectively. Relatively high demand scenarios are chosen in order to better explore their effects.

6.3.1 Delay vs. Ramp Weights

This section discusses about how the changing in the ramp weight will affect the delay of each movement type and the whole system. Figure 6.2 shows the demand scenario used for this sensitivity analysis.

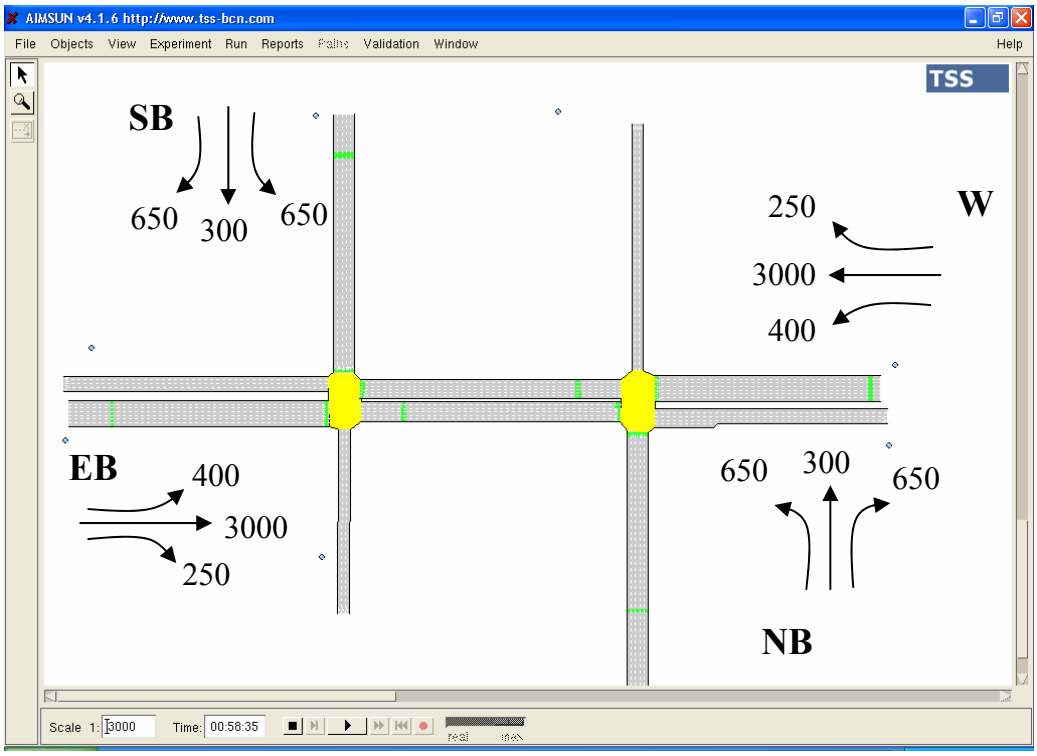


Figure 6.2 O-D Demand Scenario for Ramp Weight

Table 6.5 presents the simulation results of the delay on each movement type and the entire system with increasing ramp weights. The weights for arterial external and internal approaches are kept unchanged with the value of 1. The ramp weights increase from 1 to 3 in an interval of 0.25. It can be found that with larger weights added on the

ramp approach, as a general trend, the ramp left turn delays are reduced and arterial delays are increased, except in the case of “1.25 R, 1 Ext, 1 Int” and "1.5 R, 1 Ext, 1 Int”. The reason why the operational performance of the interchange in the case of “1.5 Ramp” does not follow the general tendency is required to be further investigated. One possible reason might be that the arterial vehicles cannot be discharged quickly due to lower weight in the beginning, thus its value of *w[i]*q[i]* becomes larger which would prevents the discharge from the ramps. Hence, it is important to keep proper weights assigned to both ramps and arterials to improve entire system’s efficiency.

Table 6.5 Delay vs. Ramp Weight

		Demand (Veh/h)	* 1 R 1 Ext 1 Int	1.25 R 1 Ext 1 Int	1.5 R 1 Ext 1 Int	1.75 R 1 Ext 1 Int	2 R 1 Ext 1 Int	2.25 R 1 Ext 1 Int	2.5 R 1 Ext 1 Int	2.75 R 1 Ext 1 Int	3 R 1 Ext 1 Int
NB	L	650	**0:02:25**	**0:00:39**	**0:01:00**	**0:00:37**	**0:00:23**	**0:00:21**	**0:00:20**	**0:00:24**	**0:00:21**
	TH	300	0:00:36	0:00:22	0:00:20	0:00:18	0:00:19	0:00:18	0:00:16	0:00:15	0:00:15
	R	650	0:00:40	0:00:27	0:00:21	0:00:21	0:00:22	0:00:19	0:00:18	0:00:16	0:00:16
SB	L	650	**0:02:25**	**0:00:37**	**0:00:37**	**0:00:27**	**0:00:24**	**0:00:21**	**0:00:22**	**0:00:20**	**0:00:19**
	TH	300	0:00:43	0:00:21	0:00:18	0:00:17	0:00:18	0:00:17	0:00:17	0:00:14	0:00:14
	R	650	0:00:36	0:00:19	0:00:13	0:00:13	0:00:15	0:00:13	0:00:12	0:00:10	0:00:11
EB	L	400	0:00:19	0:00:20	0:01:12	0:00:41	0:00:24	0:00:28	0:00:28	0:01:13	0:01:14
	TH-EXT	3000	0:00:25	0:00:31	0:01:26	0:00:37	0:00:34	0:00:30	0:00:35	0:01:29	0:01:04
	TH-INT		0:00:05	0:00:05	0:00:07	0:00:07	0:00:08	0:00:08	0:00:09	0:00:13	0:00:13
	R	250	0:00:00	0:00:00	0:00:06	0:00:01	0:00:00	0:00:00	0:00:00	0:00:08	0:00:05
WB	L	400	0:00:19	0:00:22	0:02:33	0:00:33	0:00:24	0:00:33	0:00:33	0:01:41	0:00:48
	TH-EXT	3000	0:00:25	0:00:26	0:01:34	0:01:05	0:00:29	0:00:32	0:00:34	0:01:19	0:01:08
	TH-INT		0:00:05	0:00:05	0:00:05	0:00:06	0:00:07	0:00:08	0:00:08	0:00:12	0:00:12
	R	250	0:00:00	0:00:00	0:00:08	0:00:05	0:00:00	0:00:00	0:00:00	0:00:09	0:00:05
System Delay			41s	29s	1m04s	40s	30s	30s	31s	1m01s	50s

*: R: Ramp links Ext: Arterial External links Int: Arterials Internal links

Bold number in the table represents the delays for NB and SB Left turning movements

6.3.2 Delay vs. Arterial Weights

This section investigates how the increase in arterial weights will change the movement delay and system delay. The demand scenario employed for this sensitivity analysis is illustrated in Figure 6.3.

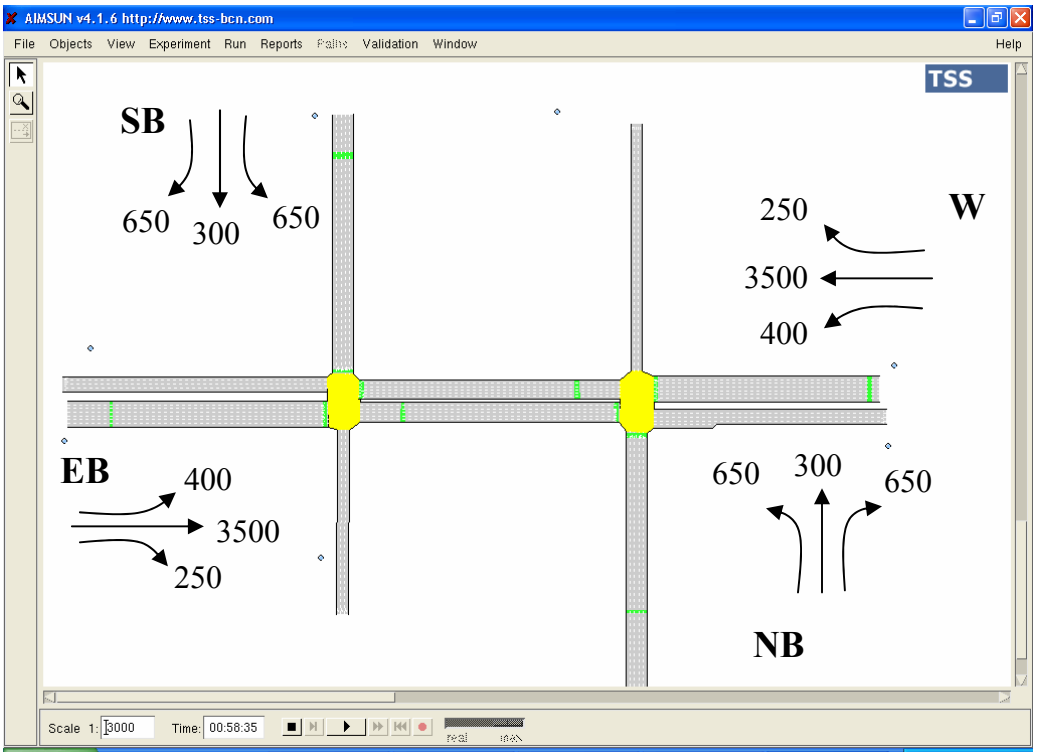


Figure 6.3 O-D Demand Scenario for Arterial Weight

As shown in Table 6.6, with the increase of weights on arterials external links, the delays on these approaches (i.e., both EB and WB external) decrease. When the weight increases from 1 to 5 in a step of 0.25, the external links delay is reduced from more than 2 minutes to around 17 seconds, and the ramp delay is increased from less than 1 minute to more than 5 minutes (over-saturated). Compared to the effects of changing the ramp weight, there is a fair and clear tendency on the relationship between the arterial weight and the arterial delay.

Table 6.6 Delay vs. Arterial Weight

		Demand (Veh/h)	* 2 Ramp 1 Ext 1 Int	2 Ramp 1.25 Ext 1 Int	2 Ramp 1.5 Ext 1 Int	2 Ramp 1.75 Ext 1 Int	2 Ramp 2 Ext 1 Int	2 Ramp 2.25 Ext 1 Int	2 Ramp 2.5 Ext 1 Int	2 Ramp 2.75 Ext 1 Int	2 Ramp 3 Ext 1 Int
NB	L	650	0:00:42	0:01:20	0:00:55	0:01:06	0:01:19	0:03:54	0:01:56	0:03:12	0:05:10
	TH	300	0:00:16	0:00:33	0:00:21	0:00:27	0:00:26	0:01:31	0:00:56	0:01:14	0:02:57
	R	650	0:00:17	0:00:16	0:00:25	0:00:30	0:00:32	0:01:07	0:00:59	0:01:18	0:02:50
SB	L	650	0:00:37	0:00:59	0:01:31	0:01:09	0:00:54	0:03:40	0:01:50	0:03:10	0:05:23
	TH	300	0:00:16	0:00:23	0:00:22	0:00:24	0:00:25	0:01:32	0:00:51	0:01:12	0:03:38
	R	650	0:00:10	0:00:09	0:00:18	0:00:21	0:00:23	0:01:10	0:00:50	0:01:01	0:02:52
EB	L	400	0:03:00	0:01:02	0:03:35	0:01:07	0:00:54	0:00:49	0:01:02	0:00:57	0:00:42
	TH-EXT	3500	**0:02:36**	**0:02:25**	**0:00:48**	**0:00:35**	**0:00:28**	**0:00:28**	**0:00:21**	**0:00:18**	**0:00:17**
	TH-INT		0:00:14	0:00:27	0:00:09	0:00:09	0:00:08	0:00:08	0:00:09	0:00:07	0:00:06
	R	250	0:00:16	0:00:11	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00
WB	L	400	0:03:29	0:01:28	0:02:00	0:00:59	0:01:03	0:00:54	0:01:11	0:00:48	0:00:48
	TH-EXT	3500	**0:02:14**	**0:02:22**	**0:00:38**	**0:00:35**	**0:00:28**	**0:00:24**	**0:00:19**	**0:00:17**	**0:00:16**
	TH-INT		0:00:11	0:00:25	0:00:10	0:00:09	0:00:07	0:00:06	0:00:08	0:00:07	0:00:05
	R	250	0:00:15	0:00:13	0:00:02	0:00:01	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00
System Delay			**1m37s**	**1m37s**	**51s**	**41s**	**36s**	**56s**	**42s**	**48s**	**1m16s**

*: R: Ramp links Ext: Arterial External links Int: Arterails Internal links

Bold number in the table represents the delays for EB and WB Through movements

6.3.3 Delay vs. Internal Link Left Turn Weights

The interchange performance due to the changing of the weights for internal link left turn is examined in this section. O-D demand scenario is given in Figure 6.4, which is the same as the one used for studying the ramp weight.

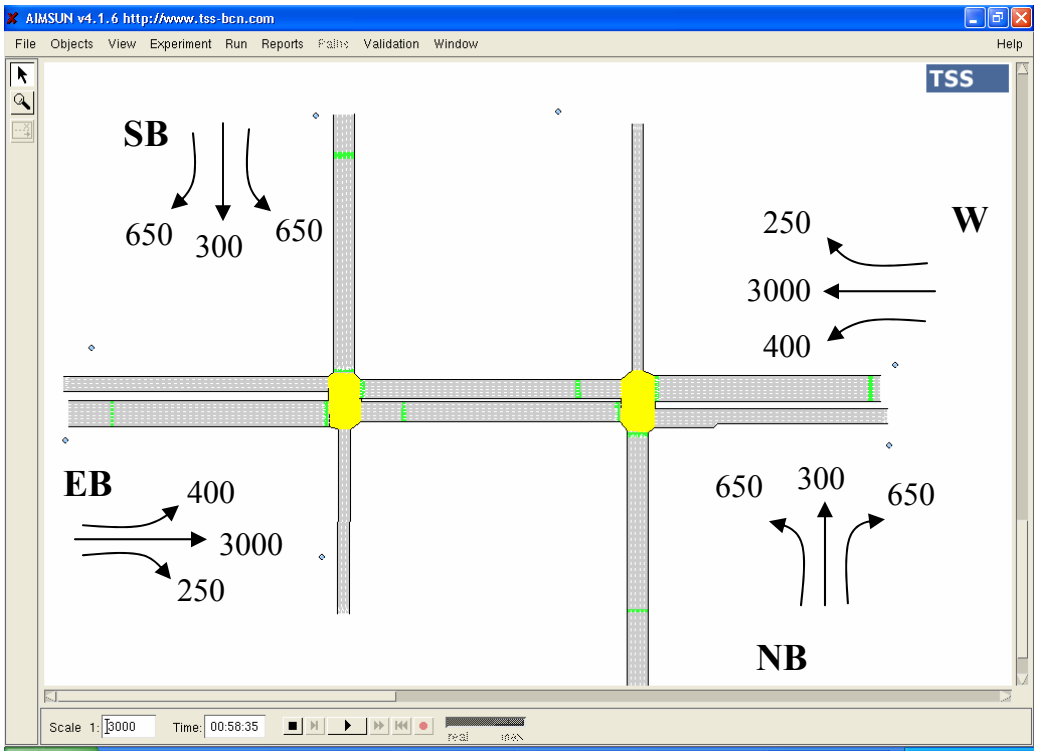


Figure 6.4 O-D Demand Scenario for Internal Link Left Turn Weight

The results in Table 6.7 reveal how the delay of each movement and whole system are affected by increasing the weights on internal links. Each approach shows a clear tendency that its delay is either reduced or increased as the weight of internal links increases. When the internal link weight is much smaller than those for other approaches, i.e., "2 Ramp, 1 Ext and 0.25 Int", the internal left turners are highly delayed with more than 2 minutes per vehicle. By increasing the internal link weight of 0.25 (i.e., "2 Ramp, 1 Ext and 0.5 Int"), the left turn vehicles can be discharged more efficiently with only less than 1 min of delay. This further indicates the importance of having proper weights on both ramps and arterials.

Table 6.7 Delay vs. Internal Link Left Turning Weight

		Demand (Veh/h)	2 Ramp 1 Ext 0.25 Int	2 Ramp 1 Ext 0.5 Int	2 Ramp 1 Ext 0.75 Int	2 Ramp 1 Ext 1 Int	2 Ramp 1 Ext 1.25 Int	2 Ramp 1 Ext 1.5 Int	2 Ramp 1 Ext 1.75 Int	2 Ramp 1 Ext 2 Int
NB	L	650	0:00:25	0:00:22	0:00:26	0:00:23	0:00:58	0:00:54	0:00:52	0:00:54
	TH	300	0:00:16	0:00:16	0:00:18	0:00:19	0:00:17	0:00:16	0:00:17	0:00:18
	R	650	0:00:18	0:00:19	0:00:21	0:00:22	0:00:17	0:00:16	0:00:16	0:00:16
SB	L	650	0:00:20	0:00:21	0:00:25	0:00:24	0:00:34	0:00:39	0:00:35	0:00:39
	TH	300	0:00:27	0:00:17	0:00:17	0:00:18	0:00:16	0:00:15	0:00:16	0:00:18
	R	650	0:00:11	0:00:13	0:00:14	0:00:15	0:00:10	0:00:09	0:00:10	0:00:10
EB	L	400	**0:02:18**	**0:00:39**	**0:00:39**	**0:00:24**	**0:00:35**	**0:00:33**	**0:00:24**	**0:00:22**
	TH-EXT	3000	0:00:30	0:00:27	0:00:32	0:00:34	0:02:27	0:02:59	0:02:45	0:03:01
	TH-INT		0:00:15	0:00:08	0:00:08	0:00:08	0:00:11	0:00:14	0:00:12	0:00:12
	R	250	0:00:10	0:00:00	0:00:00	0:00:00	0:00:13	0:00:17	0:00:15	0:00:16
WB	L	400	**0:03:53**	**0:00:57**	**0:00:41**	**0:00:24**	**0:00:33**	**0:00:32**	**0:00:25**	**0:00:25**
	TH-EXT	3000	0:00:34	0:00:26	0:00:33	0:00:29	0:02:10	0:02:35	0:02:46	0:02:58
	TH-INT		0:00:17	0:00:06	0:00:07	0:00:07	0:00:10	0:00:13	0:00:11	0:00:10
	R	250	0:00:09	0:00:00	0:00:00	0:00:00	0:00:16	0:00:18	0:00:18	0:00:18
System Delay			0:00:34	0:00:28	0:00:31	0:00:30	0:01:28	0:01:42	0:01:40	0:01:46

*: R: Ramp links Ext: Arterial External links Int: Arterails Internal links

Bold number in the table represents the delays for EB and WB Left turning movements

6.4 Delay vs. Various Demand Scenarios

The section investigates the performance of the proposed DP signal control in response to various demand scenarios. A total of twelve types of demand scenarios are developed in this study, including both the balanced demands and the unbalanced demands. As shown in Table 6.8, these scenarios covers various demand situations such as high ramp volume and one high arterial volume, two high ramp volumes, two high arterial volumes and more than two approaches with medium to high volumes. The demand for each movement type is also available in the table. Table 6.8 also presents the delay of each movement type and the whole system for each demand scenario. Both dynamic weights and fixed weights results are provided. In the case of the signal control using fixed weights, the system delay ranges only from 20s to 33s despite the big variations in demand among all scenarios. The fixed weights employed for each approach are also quoted for each scenario. Generally " 2 Ramp, 2 Ext " are used in most cases. However, in some scenarios, when the demand is excessively high, the weights have to be properly adjusted in order that the approaches with high demand get higher weights than other approaches. In such cases, the dynamic weights method shows its flexibility to respond high volume scenarios. The results due to the use of fixed weights and dynamic weights are similar. Both fixed weights and dynamic weights have demonstrated that the proposed signal control based on the DP algorithm is able to accommodate various demand situations.

Table 6.8 Delay vs. Various Demand Scenarios

Movement Type		Demand Scenario								
		1			2			3		
		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)	
		High SB High EB	Fixed 2 Ramp 2 Ext	Dynamic	High SB High WB	Fixed 2 Ramp 2 Ext	Dynamic	High NB High EB	Fixed 2.5 NB 2 EB	Dynamic
NB	L	150	0:00:17	0:00:21	200	0:00:15	0:00:23	**750**	0:00:44	0:01:06
	TH	100	0:00:16	0:00:21	100	0:00:15	0:00:22	200	0:00:17	0:00:32
	R	150	0:00:16	0:00:21	150	0:00:15	0:00:22	450	0:00:18	0:00:40
SB	L	**700**	0:00:44	0:00:50	**700**	0:00:42	0:00:40	200	0:00:12	0:00:28
	TH	**300**	0:00:25	0:00:28	**400**	0:00:25	0:00:33	100	0:00:14	0:00:25
	R	**600**	0:00:20	0:00:30	**600**	0:00:20	0:00:28	50	0:00:08	0:00:24
EB	L	400	0:00:12	0:00:11	400	0:00:27	0:00:26	400	0:00:13	0:00:11
	TH-EXT	**3000**	0:00:11	0:00:14	1800	0:00:12	0:00:11	**3500**	0:00:13	0:00:15
	TH-INT		0:00:05	0:00:04		0:00:03	0:00:02		0:00:07	0:00:04
	R	200	0:00:00	0:00:00	200	0:00:00	0:00:00	200	0:00:00	0:00:00
WB	L	100	0:00:26	0:00:24	200	0:00:17	0:00:25	100	0:00:33	0:00:20
	TH-EXT	1500	0:00:19	0:00:13	**3000**	0:00:15	0:00:16	1500	0:00:24	0:00:16
	TH-INT		0:00:02	0:00:02		0:00:05	0:00:03		0:00:04	0:00:01
	R	50	0:00:00	0:00:00	100	0:00:00	0:00:00	50	0:00:00	0:00:00
System			0:00:20	0:00:21		0:00:21	0:00:21		0:00:21	0:00:23

(Continued in the next page)

* Bold number in the table represents that the corresponding movement has high demand.

(Table 6.8 Continued from the previous page)

Movement Type		Demand Scenario								
		4			5			6		
		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)	
		High NB High SB High EB	Fixed 2.5 Ramp 2 EB	Dynamic	High EB&WB TH and L	Fixed 2.5 Ramp 2 Ext 2 L	Dynamic	High NB High SB	Fixed 2 Ramp 2 Ext	Dynamic
NB	L	**750**	0:00:25	0:00:40	300	0:01:13	0:02:43	**750**	0:00:23	0:00:28
	TH	200	0:00:16	0:00:27	150	0:00:29	0:01:11	**300**	0:00:18	0:00:21
	R	450	0:00:17	0:00:32	100	0:00:21	0:01:03	**750**	0:00:20	0:00:27
SB	L	**700**	0:00:32	0:02:16	300	0:01:44	0:02:55	**750**	0:00:21	0:00:27
	TH	300	0:00:16	0:00:30	200	0:00:31	0:01:28	**300**	0:00:17	0:00:21
	R	350	0:00:11	0:00:21	150	0:00:22	0:01:02	**750**	0:00:13	0:00:20
EB	L	400	0:00:13	0:00:13	**750**	0:00:15	0:00:11	400	0:00:20	0:00:32
	TH-EXT	**4000**	0:00:26	0:00:52	**3000**	0:00:22	0:00:14	2000	0:00:17	0:00:15
	TH-INT		0:00:12	0:00:08		0:00:03	0:00:02		0:00:07	0:00:03
	R	200	0:00:00	0:00:02	**350**	0:00:00	0:00:00	2500	0:00:00	0:00:00
WB	L	100	0:01:12	0:00:29	**600**	0:00:16	0:00:12	400	0:00:19	0:00:38
	TH-EXT	1500	0:00:27	0:00:23	**3500**	0:00:25	0:00:16	2000	0:00:17	0:00:15
	TH-INT		0:00:04	0:00:04		0:00:03	0:00:01		0:00:05	0:00:03
	R	50	0:00:00	0:00:00	200	0:00:00	0:00:00	250	0:00:00	0:00:00
System			0:00:28	0:00:46		0:00:27	0:00:28		0:00:21	0:00:22

(Continued in the next page)

* Bold number in the table represents that the corresponding movement has high demand.

(Table 6.8 Continued from the previous page)

Movement Type		Demand Scenario								
		7			8			9		
		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)	
		High NB High SB Med EB Med WB	Fixed 2 Ramp 2 Ext	Dynamic	High NB High SB Med EB Med WB	Fixed 2 Ramp 2 Ext	Dynamic	High NB High SB High EB High WB	Fixed 2 Ramp 2 Ext	Dynamic
NB	L	**750**	0:00:24	0:00:33	**750**	0:00:27	0:00:32	**750**	0:00:29	0:00:35
	TH	**300**	0:00:17	0:00:26	**300**	0:00:20	0:00:25	**300**	0:00:21	0:00:27
	R	**750**	0:00:23	0:00:31	**750**	0:00:24	0:00:31	**750**	0:00:25	0:00:33
SB	L	**750**	0:00:26	0:00:33	**750**	0:00:26	0:00:30	**750**	0:00:30	0:00:35
	TH	**300**	0:00:18	0:00:26	**300**	0:00:20	0:00:26	**300**	0:00:20	0:00:28
	R	**750**	0:00:15	0:00:23	**750**	0:00:16	0:00:23	**750**	0:00:17	0:00:25
EB	L	400	0:00:27	0:00:20	400	0:00:26	0:00:32	400	0:00:27	0:00:30
	TH-EXT	**2500**	0:00:17	0:00:17	**2700**	0:00:19	0:00:17	**3000**	0:00:22	0:00:19
	TH-INT		0:00:07	0:00:04		0:00:05	0:00:04		0:00:06	0:00:04
	R	250	0:00:00	0:00:00	250	0:00:00	0:00:00	250	0:00:00	0:00:00
WB	L	400	0:00:26	0:00:26	400	0:00:27	0:00:29	400	0:00:32	0:00:36
	TH-EXT	**2500**	0:00:17	0:00:19	**2700**	0:00:19	0:00:16	**3000**	0:00:21	0:00:19
	TH-INT		0:00:06	0:00:03		0:00:05	0:00:03		0:00:05	0:00:03
	R	250	0:00:00	0:00:00	250	0:00:00	0:00:00	250	0:00:00	0:00:00
System			0:00:22	0:00:24		0:00:23	0:00:23		0:00:25	0:00:25

(Continued in the next page)

* Bold number in the table represents that the corresponding movement has high demand.

(Table 6.8 Continued from the previous page)

Movement Type		Demand Scenario								
		10			11			12		
		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)		Demand * (veh/h)	Delay (sec/veh)	
		High NB High SB High EB High WB	Fixed 2 Ramp 2 Ext	Dynamic	Med EB Med WB	Fixed 2 Ramp 2 Ext	Dynamic	High EB Med WB Med NB	Fixed 2 Ramp 2 Ext	Dynamic
NB	L	**750**	0:00:58	0:00:40	350	0:00:26	0:00:52	**550**	0:00:54	0:01:30
	TH	**300**	0:00:22	0:00:30	300	0:00:25	0:00:49	200	0:00:20	0:00:42
	R	**750**	0:00:29	0:00:38	350	0:00:21	0:00:51	300	0:00:21	0:00:44
SB	L	**750**	0:00:56	0:00:40	350	0:00:32	0:00:48	350	0:00:26	0:00:43
	TH	**300**	0:00:23	0:00:29	300	0:00:23	0:00:46	300	0:00:22	0:00:40
	R	**750**	0:00:19	0:00:30	350	0:00:16	0:00:38	250	0:00:14	0:00:33
EB	L	400	0:00:45	0:00:32	400	0:00:12	0:00:23	400	0:00:17	0:00:21
	TH-EXT	**3300**	0:00:27	0:00:18	**2500**	0:00:18	0:00:13	**3000**	0:00:11	0:00:12
	TH-INT		0:00:07	0:00:05		0:00:03	0:00:03		0:00:05	0:00:03
	R	250	0:00:00	0:00:00	250	0:00:00	0:00:00	200	0:00:00	0:00:00
WB	L	400	0:00:52	0:00:36	400	0:00:13	0:00:29	200	0:00:23	0:00:25
	TH-EXT	**3300**	0:00:27	0:00:20	**2500**	0:00:17	0:00:13	**2500**	0:00:16	0:00:14
	TH-INT		0:00:07	0:00:04		0:00:03	0:00:02		0:00:03	0:00:02
	R	250	0:00:00	0:00:00	250	0:00:00	0:00:00	250	0:00:00	0:00:00
System			0:00:33	0:00:27		0:00:20	0:00:24		0:00:19	0:00:23

* Bold number in the table represents that the corresponding movement has high demand.

6.5 Summary

This chapter has firstly discussed how the different types of PMI definition affect system delay under demand scenarios ranged from low to high demand and compared the performance of dynamic weights and fixed weights, then investigated into how the system delay changes as the weights on ramps, arterials and internal links of arterials vary and finally examined the performance of the proposed DP algorithm under various demand scenarios. Major findings from the sensitivity analyses are given as follows,

1. There are four types of PMI defined in this study: PMI 1: Sum of Average Queue Length Per Lane For All Approaches, PMI 2: Sum of Average Delay Per Lane For All Approaches, PMI 3: Sum of Total Delays for All Approaches and PMI 4: Sum of Storage Ratio for All Approaches. The results have shown that PMI 1 and PMI 4 are better optimization criteria than PMI 2 and PMI 3 in terms of minimizing system delay of the simulated interchange under medium to high and high demand scenarios. In the low demand scenarios, all four types of PMI definitions have resulted in very close operational performance.

2. PMI 1 and PMI 4 results are very similar in this study. This is because this particular interchange has the same link length for each approach. This specific geometric layout results in no difference between PMI 1 and PMI 4 as optimization criteria. In the future research, a diamond interchange with left pocket lane or different approach length can be applied to find out which performance measure index is superior.

3. A general rule of choosing the weight of an approach is that a larger weight can usually be applied to the approaches having more demand. But, this would slow the vehicles discharge on other approach(es). Hence, a proper adjustment of the weight on each approach needs to be made to achieve a satisfactory overall performance of the interchange.

4. The study has compared results from using dynamic weights and fixed weights when the demand is unpredictable, varied in each 15-minute period and unbalanced on the interchange. The dynamic weight can significantly reduce system delay than the fixed weight does in various unbalanced demand scenarios, e.g., one approach (i.e., arterial or ramp) with higher demand, two approaches (i.e., two arterials, or one arterial with one ramp) with higher demand. Comparing to fixed weight, the saving on system delay by using dynamic weight is ranged between 36 percent and 49 percent (i.e., 22s, 13s, 38s, 22s and 19s, respectively). The application of dynamic weights enables the DP algorithm more flexible and adaptive to demand fluctuation.

5. The real-time DP algorithm has demonstrated the capability to accommodate various demand situations. These scenarios include high ramp volume together with one high arterial volume, two high ramp volumes, and two high arterial volumes with other approaches having medium to high volume. Both fixed weights and dynamic weights have shown small range of system delay variation, from 20s to 33s. However, fixed weights have to be properly adjusted using the rule developed to achieve good performance of the traffic operation of a diamond interchange. On the other hand, the study has shown the flexibility of using dynamic weights that do not require manually changing the weights in getting a same satisfactory or better performance as fixed weights do under various demand scenarios.

6. The performance of dynamic weights also depends on how the values of dynamic weights are defined. It is needed to calibrate these criteria in order to achieve greatest performance. The more systematic comparison between dynamic weights and fixed weights under different demand scenarios will be conducted in the next chapter.

CHAPTER 7
COMPARISONS OF DIFFERENT OPTIMIZATION METHODS

This chapter presents comparisons on the performance of three different optimization methods, i.e., the real-time DP algorithm, PASSER III and TRANSYT-7F. The comparisons are conducted using the calibrated interchange (see Figure 5.2) based on low demand and varying demand, as described in Section 7.1 and 7.2, respectively. Section 7.2 also discusses several varying demand scenarios, i.e., unbalanced demand with varying demand on ramps, varying demand on arterials, and varying demand on both ramps and arterials. The performance measures (i.e., system delay and movement delay) from three different optimization algorithms are compared with respect to each scenario. Finally a summary of the comparisons is provided.

7.1 Overview

For each demand scenario, the optimal signal plans from PASSER III and TRANSYT-7F are obtained before the AIMSUN simulation. These optimal fixed signal plans are then input to the AIMSUN interchange network. The signal plans of the DP algorithm are not pre-determined while they are generated in each 10 seconds based on real-time demand situations. The simulated diamond interchange is then used to compare the performance of three different optimal signal plans under the same scenarios. The simulation time is 1 hour.

7.2 Low Demand

The performance of three optimization tools is compared in a low demand scenario as shown in Figure 7.1.

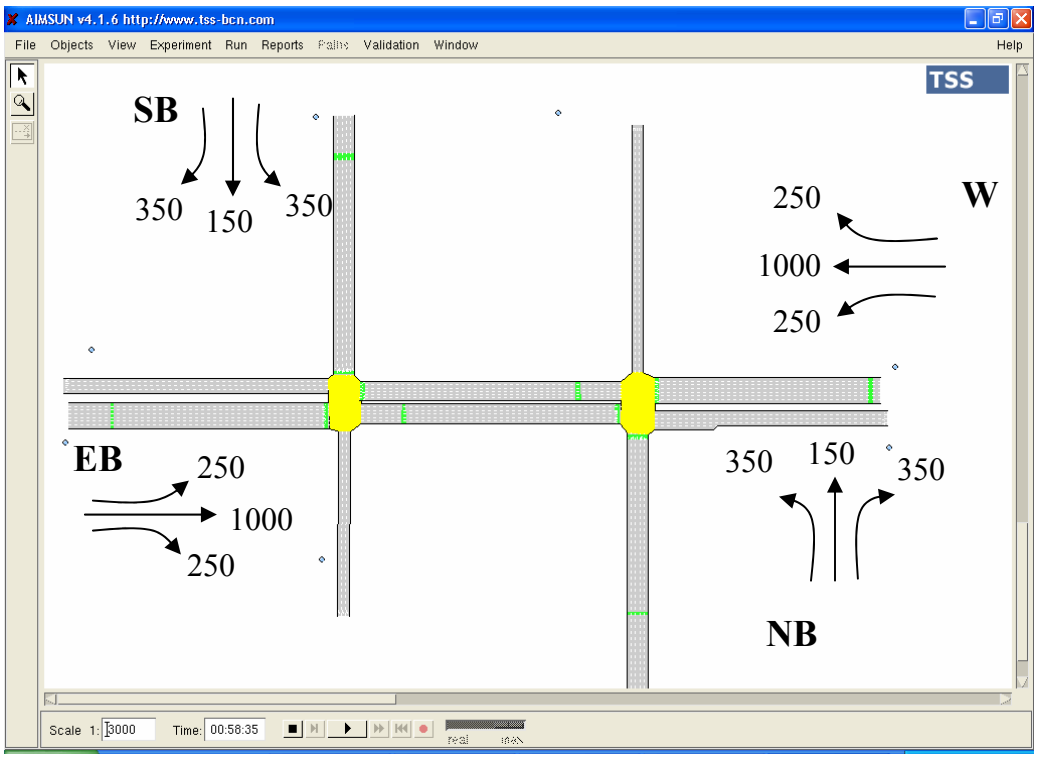


Figure 7.1 Low Demand

Based on this low demand, the off-line signal optimization tools of PASSER III and TRANSYT-7F generate their optimal signal plans. Table 7.1 and 7.2 presents the signal plan from PASSER III and TRANSYT-7F, respectively. Optimal (best) signal plan from PASSER III is a four-phase plan with overlaps. TRANSYT-7F does not optimize signal phase sequence, therefore the signal phase sequence obtained from PASSER III is used in TRANSYT-7F.

Table 7.1 PASSER III Optimal Signal Plan

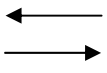
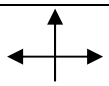
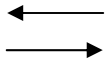
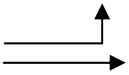
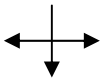
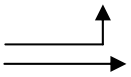
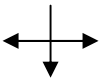
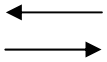
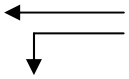
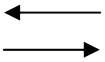
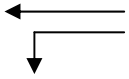
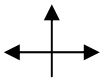
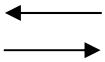
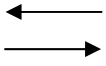
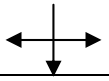
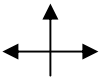
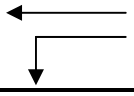
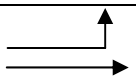
Four-Phase Plan with Overlap C = 60s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Overlap with Phase 1		19		19
Phase 2		6		6
Phase 3		4		4
Overlap with Phase 4		19		19
Phase 5		8		8
Phase 6		4		4

Table7.2 TRANSYT-7F Optimal Signal Plan

C=65s, Offset=0s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Phase 1		27		27
Phase 2		22		22
Phase 3		16		16

These optimized pre-timed signal plans from TRANSYT-7F and PASSER III are implemented in the interchange for AIMSUN simulation and the results are compared to those from the DP algorithm that is dynamically linked to the interchange during the

simulation. AIMSUN simulation also employs the low demand (Figure 7.1) defined for the off-line optimization. The simulation results are given in Table 7.4. The results have shown that the delays of the DP algorithm using fixed weights and of PASSER III are slightly smaller than those of the DP algorithm using dynamic weights and of TRANSYT-7F. The difference in the system delay is around 6 or 7 seconds. Nevertheless, all of them have shown the similar performance.

Table 7.3 A Comparison of System Delays (sec/veh) for the Low Demand Scenario

#	Demand Scenario	Movement Type		Real-Time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext	Dynamic Weight		
1	Low Demand	NB	L	0:00:20	0:00:40	0:00:16	0:00:18
			TH	0:00:20	0:00:33	0:00:16	0:00:20
			R	0:00:21	0:00:45	0:00:15	0:00:19
		SB	L	0:00:19	0:00:36	0:00:15	0:00:18
			TH	0:00:18	0:00:30	0:00:15	0:00:20
			R	0:00:15	0:00:33	0:00:12	0:00:15
		EB	L	0:00:10	0:00:13	0:00:13	0:00:27
			TH-EXT	0:00:11	0:00:09	0:00:14	0:00:14
			TH-INT	0:00:03	0:00:02	0:00:02	0:00:12
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:09	0:00:17	0:00:11	0:00:27
			TH-EXT	0:00:12	0:00:10	0:00:13	0:00:15
			TH-INT	0:00:03	0:00:01	0:00:02	0:00:11
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:15**	**0:00:21**	**0:00:14**	**0:00:22**

7.3 Varying Demand

7.3.1 Introduction

A varying demand means that simulation (or field) demand is deviated from the one used for the off-line PASSER III or TRANSYT-7F optimization, and also varies in each 15-minute period for one hour. Some varying demand scenarios are developed in this study to examine the capability of each optimization tool in handling demand fluctuations. The following sections analyze three cases of varying demand scenarios: Case A: Unbalanced Demand with Varying Demand on Ramps, Case B: Varying Demand on Arterials, and Case C: Varying Demand on Ramps and Arterials.

7.3.2 Case A: Unbalanced Demand with Varying Demand on Ramps

Unbalanced demand describes that demands are asymmetrically distributed on a diamond interchange. Figure 7.2 shows the unbalanced demand employed:

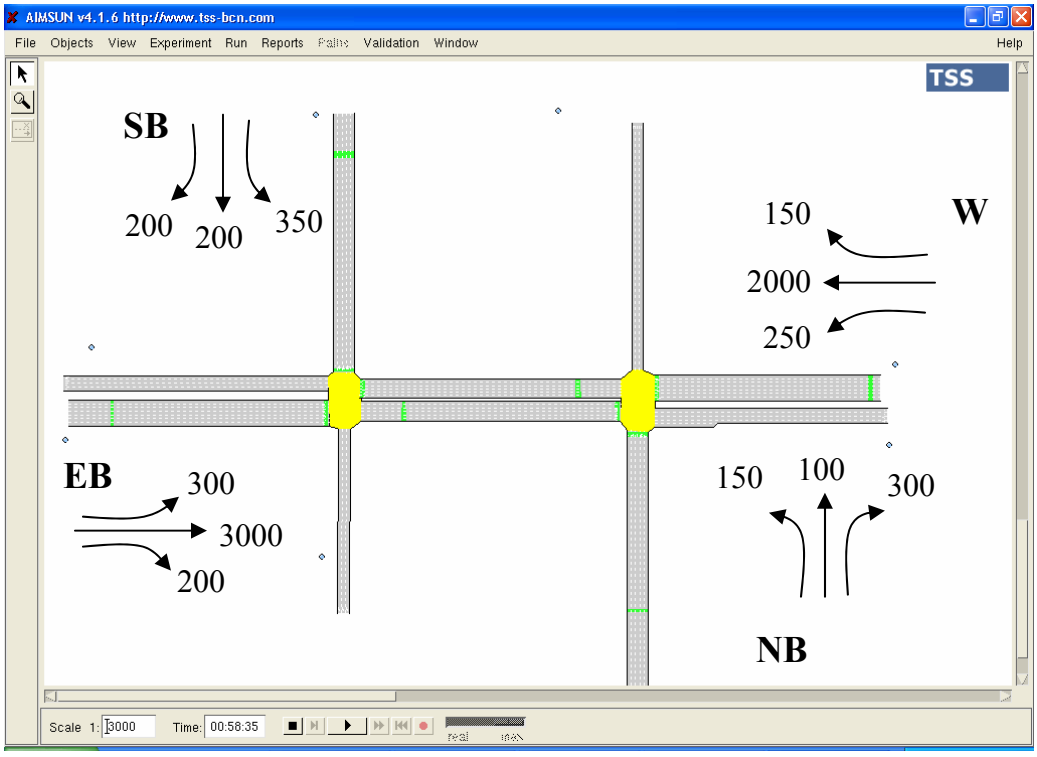


Figure 7.2 Unbalanced Demand

Given the O-D volume in Figure 7.2, PASSER III has produced an optimal signal plan presented in Table 7.4. Table 7.5 shows the optimal signal plan obtained from TRANSYT-7F.

Table 7.4 PASSER III Optimal Signal Plan

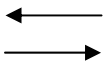
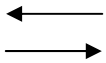
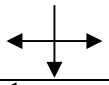
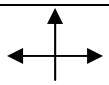
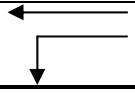
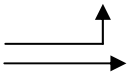
C = 60s, offset = 15s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Phase 1		36		15
Phase 2		13		35
Phase 3		11		10

Table7.5 TRANSYT-7F Optimal Signal Plan

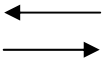
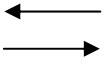
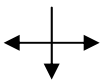
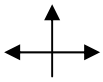
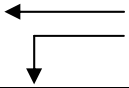
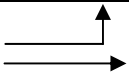
C = 75s, Offset = 12s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Phase 1		43		41
Phase 2		16		15
Phase 3		16		19

Table 7.6 provides four demand scenarios used in AIMSUN simulation. The demand varies in each 15 minutes and "total" denotes hourly volume that is a sum of these 15 minutes volumes. The demand used in the optimization is also provided in the table as a reference to compare with the simulation demand. In each scenario, simulation demand varies from the optimization demand with the ramp movement volumes increased by 6 percent to 67 percent (10 to 150 veh/h).

Table 7.6 O-D Demand of Each Scenario for Case A: Unbalanced Demand with Varying Demand on Ramps

#	Scenario	Movement Type			Demand* (veh/h) 0-15m	15-30m	30-45m	45-60m	Total (1h)	Optimization Demand (1h)
1	Scenario 1	Ramp	NB	L	**100**	**60**	**50**	**40**	**250**	150
				TH	**40**	**50**	**40**	**30**	**160**	100
				R	100	60	50	40	250	300
			SB	L	**200**	**150**	**100**	**50**	**500**	350
				TH	80	50	40	30	200	200
				R	**50**	**100**	**50**	**50**	**250**	200
		Arterial	EB	L	50	50	50	50	200	300
				TH	1000	800	700	500	3000	3000
				R	65	60	65	60	250	200
			WB	L	50	50	50	50	200	250
				TH	800	600	400	400	**2200**	2000
				R	60	65	65	60	250	150
2	Scenario 2	Ramp	NB	L	**100**	**60**	**50**	**40**	**250**	150
				TH	**30**	**50**	**40**	**30**	**150**	100
				R	150	60	50	40	300	300
			SB	L	**200**	**100**	**100**	**50**	**450**	350
				TH	80	50	40	30	200	200
				R	**50**	**100**	**50**	**50**	**250**	200
		Arterial	EB	L	50	50	50	50	200	300
				TH	900	900	700	500	3000	3000
				R	75	85	65	75	**300**	200
			WB	L	50	100	50	50	250	250
				TH	800	600	400	400	**2200**	2000
				R	60	65	65	60	**250**	150

* Bold number in the table represents that the corresponding movement has high demand.

(Continued in the next page)

(Table 7.6 Continued from the previous page)

#	Scenario	Movement Type			Demand * (veh/h) 0-15m	15-30m	30-45m	45-60m	Total (1h)	Optimization Demand (1h)
3	Scenario 3	Ramp	NB	L	50	50	30	30	**160**	150
				TH	30	50	40	30	**150**	100
				R	150	60	50	40	300	300
			SB	L	200	100	50	50	**400**	350
				TH	50	50	40	40	180	200
				R	50	50	50	50	200	200
		Arterial	EB	L	60	40	50	50	200	300
				TH	800	800	750	650	3000	3000
				R	50	50	50	50	200	200
			WB	L	50	100	50	50	250	250
				TH	800	600	400	400	2200	2000
				R	60	65	65	60	250	150
4	Scenario 4	Ramp	NB	L	50	50	30	30	**160**	150
				TH	30	50	40	30	**150**	100
				R	100	100	50	50	**300**	300
			SB	L	180	100	60	40	**380**	350
				TH	80	50	50	20	200	200
				R	50	50	50	50	200	200
		Arterial	EB	L	60	40	50	50	200	300
				TH	800	800	750	650	3000	3000
				R	50	50	50	50	200	200
			WB	L	50	100	50	50	250	250
				TH	800	600	400	400	**2200**	2000
				R	60	65	65	60	250	150

* Bold number in the table represents that the corresponding movement has high demand.

Table 7.7 presents the delay resulted from the DP algorithm using fixed weights, the DP algorithm using the dynamic weights, PASSER III and TRANSYT-7F for unbalanced demand scenarios, respectively. The results have shown that each method exhibits nearly the same system delay among all demand scenarios. But, both PASSER III and TRANSYT-7F have shown much higher ramp delays (e.g., 1 minute) than the DP algorithm in some scenarios.

Table 7.7 A Comparison of Delay (sec/veh) for Case A: Unbalanced Demand with Varying Demand on Ramps

#	Demand Scenario	Movement Type		Real-Time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 1 Ext 1 Int	Dynamic Weight		
1	Scenario 1	NB	L	0:00:23	0:00:50	0:00:59	0:00:30
			TH	0:00:20	0:00:53	0:00:29	0:00:30
			R	0:00:19	0:00:55	0:00:49	0:00:35
		SB	L	0:00:59	0:01:41	0:03:41	0:01:39
			TH	0:00:21	0:00:53	0:00:27	0:00:30
			R	0:00:16	0:00:50	0:00:20	0:00:24
		EB	L	0:00:11	0:00:23	0:00:17	0:00:41
			TH-EXT	0:00:30	0:00:13	0:00:10	0:00:13
			TH-INT	0:00:07	0:00:03	0:00:02	0:00:04
			R	0:00:01	0:00:00	0:00:00	0:00:00
		WB	L	0:00:28	0:00:33	0:00:17	0:00:21
			TH-EXT	0:00:16	0:00:10	0:00:09	0:00:13
			TH-INT	0:00:03	0:00:02	0:00:01	0:00:10
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		0:00:26	0:00:24	0:00:26	0:00:24

(Continued in the next page)

(Table 7.7 Continued from the previous page)

#	Demand Scenario	Movement Type		Real-Time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 1 Ext 1 Int	Dynamic Weight		
2	Scenario 2	NB	L	0:00:21	0:00:50	0:00:58	0:00:32
			TH	0:00:23	0:00:55	0:00:30	0:00:32
			R	0:00:21	0:00:56	0:03:03	0:00:45
		SB	L	0:00:50	0:01:36	0:02:07	0:01:13
			TH	0:00:25	0:00:52	0:00:24	0:00:29
			R	0:00:20	0:01:00	0:00:19	0:00:25
		EB	L	0:00:11	0:00:32	0:00:15	0:00:40
			TH-EXT	0:00:24	0:00:10	0:00:10	0:00:13
			TH-INT	0:00:05	0:00:04	0:00:02	0:00:03
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:24	0:00:38	0:00:19	0:00:21
			TH-EXT	0:00:17	0:00:08	0:00:09	0:00:13
			TH-INT	0:00:03	0:00:03	0:00:01	0:00:10
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		0:00:23	0:00:23	0:00:27	0:00:23

(Continued in the next page)

(Table 7.7 Continued from the previous page)

#	Demand Scenario	Movement Type		Real-Time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 1 Ext 1 Int	Dynamic Weight		
3	Scenario 3	NB	L	0:00:21	0:00:45	0:00:24	0:00:29
			TH	0:00:26	0:00:58	0:00:30	0:00:28
			R	0:00:27	0:01:05	0:00:54	0:00:45
		SB	L	0:00:45	0:01:42	**0:02:28**	0:01:29
			TH	0:00:31	0:00:55	0:00:23	0:00:29
			R	0:00:23	0:00:57	0:00:18	0:00:25
		EB	L	0:00:10	0:00:25	0:00:16	0:00:40
			TH-EXT	0:00:18	0:00:10	0:00:09	0:00:13
			TH-INT	0:00:04	0:00:02	0:00:02	0:00:03
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:13	0:00:28	0:00:17	0:00:21
			TH-EXT	0:00:16	0:00:09	0:00:09	0:00:13
			TH-INT	0:00:03	0:00:02	0:00:01	0:00:10
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		0:00:20	0:00:23	0:00:26	0:00:23

(Continued in the next page)

(Table 7.7 Continued from the previous page)

#	Demand Scenario	Movement Type		Real-Time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 1 Ext 1 Int	Dynamic Weight		
4	Scenario 4	NB	L	0:00:24	0:00:48	0:00:25	0:00:28
			TH	0:00:26	0:00:52	0:00:29	0:00:33
			R	0:00:35	0:01:21	0:00:57	0:00:34
		SB	L	0:00:44	0:02:22	**0:01:31**	0:00:51
			TH	0:00:25	0:01:04	0:00:25	0:00:31
			R	0:00:20	0:00:59	0:00:18	0:00:25
		EB	L	0:00:11	0:00:22	0:00:16	0:00:40
			TH-EXT	0:00:14	0:00:11	0:00:09	0:00:13
			TH-INT	0:00:04	0:00:03	0:00:02	0:00:03
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:13	0:00:38	0:00:17	0:00:21
			TH-EXT	0:00:14	0:00:10	0:00:09	0:00:13
			TH-INT	0:00:03	0:00:02	0:00:01	0:00:10
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		0:00:16	0:00:26	0:00:17	0:00:21

7.3.3 Case B: Varying Demand on Arterials

Given O-D volume as shown in Figure 7.3, the PASSER III optimal (best) signal plan among all possible alternatives is a four-phase plan with overlap as shown in Table 7.8. The PASSER III optimal three-phase signal plan is also presented in Table 7.9. As the decision network used for the DP optimization does not include a phase option that accommodates both ramp and internal link movement, i.e., a four-phase plan is not included, thus the optimal three-phase signal plan from PASSER III (Table 7.9) is also used for comparisons. The TRANSYT-7F optimal plan is provided in Table 7.10.

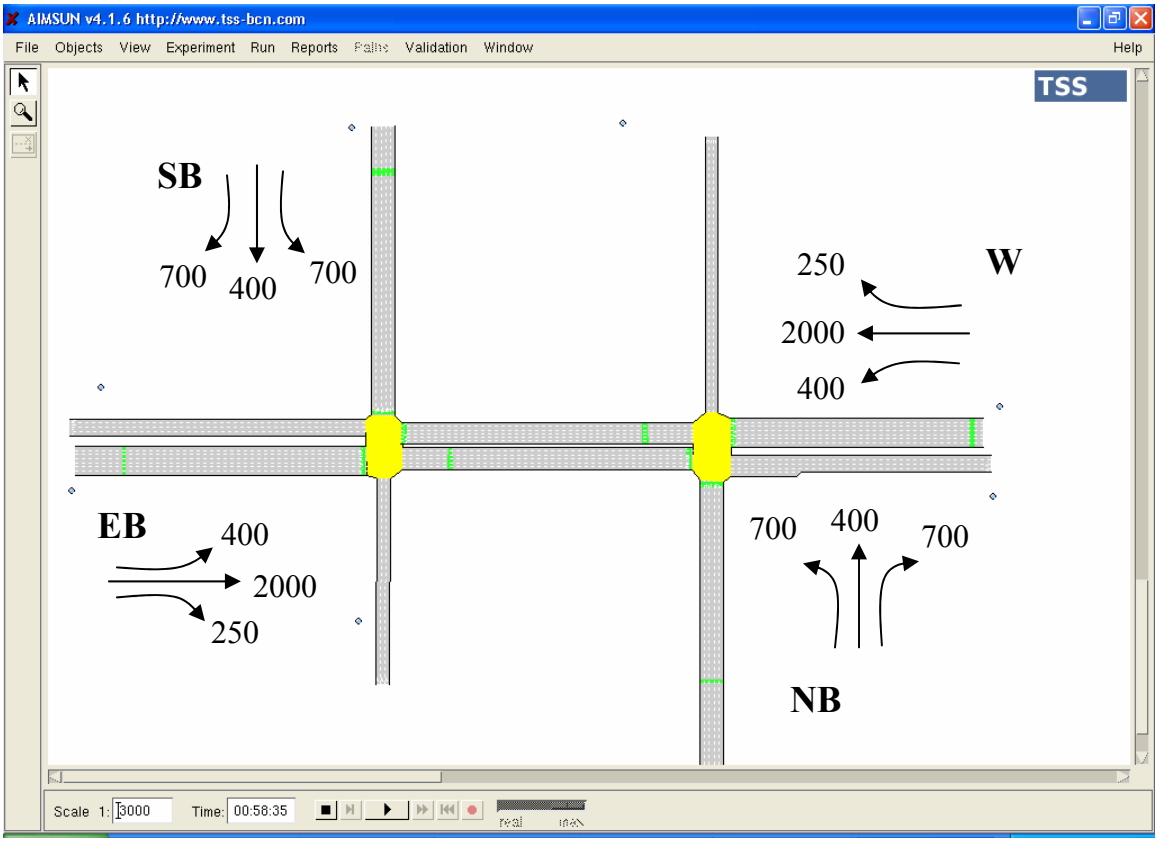


Figure 7.3 O-D Demand for Case B: Varying Demand on Arterials

Table 7.8 PASSER III Optimal (Best) Signal Plan

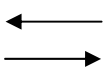
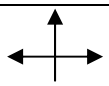
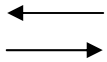
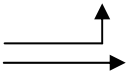
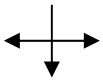
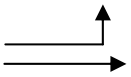
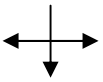
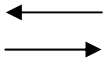
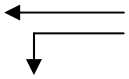
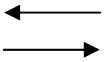
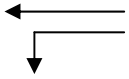
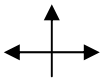
Four-Phase Plan with Overlap C = 60s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Overlap with Phase 1		17		17
Phase 2		7		7
Phase 3		6		6
Overlap with Phase 4		15		15
Phase 5		10		10
Phase 6		5		5

Table 7.9 PASSER III Optimal Three-Phase Signal plan

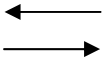
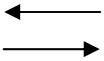
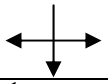
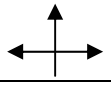
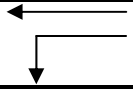
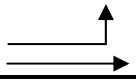
C = 60s, offset = 38s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Phase 1		24		22
Phase 2		14		25
Phase 3		22		13

Table 7.10 TRANSYT-7F Optimal Signal Plan

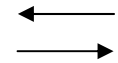
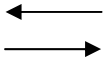
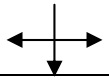
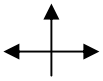
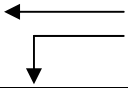
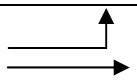
C=75s, Offset=38s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Phase 1		30		29
Phase 2		28		28
Phase 3		17		18

Table 7.11 presents the O-D demand scenarios designed for Case B. All scenarios are balanced demands. Arterials volume increase from 2000 veh/h to 3300 veh/h. The study intends to explore how the interchange performance will be affected by substantially increasing volume on certain approaches, although some simulation demands are much different from the optimization demand.

Table 7.11 O-D Demand of Each Scenario for Case B: Varying Demand on Arterials

#	Scenario	Movement Type			Demand* (veh/h) 0-15m	15-30m	30-45m	45-60m	Total (1h)	Optimization Demand (1h)
1	Arterial TH 2000 v/h	Ramp	NB or SB	L	150	200	210	190	750	700
				TH	80	75	100	45	300	400
				R	150	200	210	190	750	700
		Arterial	EB or WB	L	100	150	50	100	400	400
				TH	**700**	**600**	**400**	**300**	**2000**	2000
				R	65	60	65	60	250	250
2	Arterial TH 2300 v/h	Ramp	NB or SB	L	150	200	210	190	750	700
				TH	80	75	100	45	300	400
				R	150	200	210	190	750	700
		Arterial	EB or WB	L	100	150	50	100	400	400
				TH	**900**	**600**	**400**	**400**	**2300**	2000
				R	65	60	65	60	250	250
3	Arterial TH 2500 v/h	Ramp	NB or SB	L	150	200	210	190	750	700
				TH	80	75	100	45	300	400
				R	150	200	210	190	750	700
		Arterial	EB or WB	L	100	150	50	100	400	400
				TH	**900**	**800**	**400**	**400**	**2500**	2000
				R	65	60	65	60	250	250
4	Arterial TH 2700 v/h	Ramp	NB or SB	L	150	200	210	190	750	700
				TH	80	75	100	45	300	400
				R	150	200	210	190	750	700
		Arterial	EB or WB	L	100	150	50	100	400	400
				TH	**900**	**750**	**600**	**450**	**2700**	2000
				R	65	60	65	60	250	250

* Bold number in the table shows the arterial movement volumes

(Continued in the next page)

(Table 7.11 Continued from the previous page)

5	Arterial TH 3000 v/h	Ramp	NB or SB	L	150	200	210	190	**750**	700
				TH	80	75	100	45	**300**	400
				R	150	200	210	190	**750**	700
		Arterial	EB or WB	L	100	150	50	100	**400**	400
				TH	**900**	**750**	**800**	**550**	**3000**	2000
				R	65	60	65	60	**250**	250
6	Arterial TH 3300 v/h	Ramp	NB or SB	L	150	200	210	190	**750**	700
				TH	80	75	100	45	**300**	400
				R	150	200	210	190	**750**	700
		Arterial	EB or WB	L	100	150	50	100	**400**	400
				TH	**900**	**750**	**800**	**850**	**3300**	2000
				R	65	60	65	60	**250**	250

* Bold number in the table shows the arterial movement volumes

Table 7.12 presents the comparisons of the system delay among different optimization methods in terms of varying arterial demand scenarios. In the case of the DP algorithm, the increase of the demand does not increase the delay as much as in the case of off-line systems. When the demand is close to the optimization demand (i.e., arterial 2000 veh/h), each system has the similar results. However, when the demand increases, the delays of PASSER III and TRANSYT-7F are much higher than that of the DP algorithm. This further demonstrates that the signal plans in the on-line system are adjusted accordingly to respond to real-time demand fluctuations. With varying demands, the off-line systems would cause longer delays either on the arterial or the ramp than the on-line system.

The results in Table 7.12 have also shown that the DP algorithm using dynamic weights can achieve better performance than the one using fixed weights in dealing with increased arterial demand situations. With the increase of arterial demand from 2000 veh/h to 3300 veh/h (increase by 65 percent), the system delay resulted from fixed weights increases from 21s to 43s while the delay from dynamic weights keeps almost the same with the range between 23s and 29s. Therefore, compared to fixed weights, using dynamic weights can reduce the system delay up to 32.5 percent in handling increased arterial demands.

The delay of each movement type for each scenario is addressed in Table 7.13 in details.

Table 7.12 A Comparison of System Delay (sec/veh) for Varying Arterial Demand Scenarios

#	Scenario	Real-time DP		PASSER III		TRANSYT-7F
		Fixed Weight	Dynamic Weight	Optimal Signal Plan	Optimal Three – Phase Signal Plan	
1	Arterial TH 2000 v/h	0:00:21	0:00:26	0:00:19	0:00:19	0:00:21
2	Arterial TH 2300 v/h	0:00:26	0:00:26	0:00:38	0:00:34	0:00:33
3	Arterial TH 2500 v/h	0:00:32	0:00:29	0:00:51	0:01:06	0:01:07
4	Arterial TH 2700 v/h	0:00:34	0:00:23	0:01:01	0:01:26	0:01:26
5	Arterial TH 3000 v/h	0:00:40	0:00:24	0:01:16	0:01:21	0:01:22
6	Arterial TH 3300 v/h	0:00:43	0:00:24	0:01:21	0:01:20	0:01:32

Table 7.13 A Comparison of Delay (sec/veh) of Each Movement for Varying Arterial Demand Scenarios

#	Scenario	Movement Type		Real-time DP		PASSER III		TRANSYT-7F
				Fixed Weight 2 Ramp 1.5 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	Optimal Three – Phase Signal Plan	
1	Arterial TH 2000 v/h	NB	L	0:00:17	0:00:21	0:00:19	0:00:17	0:00:21
			TH	0:00:13	0:00:19	0:00:17	0:00:17	0:00:21
			R	0:00:16	0:00:20	0:00:18	0:00:18	0:00:21
		SB	L	0:00:15	0:00:21	0:00:19	0:00:17	0:00:21
			TH	0:00:12	0:00:17	0:00:18	0:00:17	0:00:19
			R	0:00:09	0:00:15	0:00:14	0:00:13	0:00:16
		EB	L	0:00:25	0:00:28	0:00:16	0:00:16	0:00:10
			TH-EXT	0:00:25	0:00:22	0:00:18	0:00:18	0:00:20
			TH-INT	0:00:10	0:00:10	0:00:04	0:00:06	0:00:03
			R	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:28	0:00:25	0:00:13	0:00:09	0:00:10
			TH-EXT	0:00:22	0:00:22	0:00:17	0:00:17	0:00:21
			TH-INT	0:00:09	0:00:09	0:00:05	0:00:04	0:00:02
			R	0:00:00	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:21**	**0:00:26**	**0:00:19**	**0:00:19**	**0:00:21**

(Continued in the next page)

(Table 7.13: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III		TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	Optimal Three – Phase Signal Plan	
2	Arterial TH 2300 v/h	NB	L	0:00:29	0:00:27	0:00:19	0:00:17	0:00:22
			TH	0:00:18	0:00:22	0:00:17	0:00:17	0:00:21
			R	0:00:21	0:00:25	0:00:18	0:00:18	0:00:21
		SB	L	0:00:22	0:00:24	0:00:27	0:00:18	0:00:22
			TH	0:00:18	0:00:20	0:00:18	0:00:17	0:00:20
			R	0:00:13	0:00:18	0:00:14	0:00:13	0:00:16
		EB	L	0:00:42	0:00:30	0:00:34	0:00:45	0:00:15
			TH-EXT	0:00:20	0:00:22	0:01:25	0:00:49	0:00:40
			TH-INT	0:00:09	0:00:07	0:00:06	0:00:08	0:00:03
			R	0:00:00	0:00:00	0:00:01	0:00:01	0:00:01
		WB	L	0:01:01	0:00:31	0:00:15	0:00:27	0:00:17
			TH-EXT	0:00:21	0:00:22	0:00:22	0:00:35	0:00:50
			TH-INT	0:00:07	0:00:06	0:00:05	0:00:07	0:00:03
			R	0:00:00	0:00:00	0:00:00	0:00:00	0:00:01
		System		**0:00:26**	**0:00:26**	**0:00:38**	**0:00:34**	**0:00:33**

(Continued in the next page)

(Table 7.13: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III		TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	Optimal Three – Phase Signal Plan	
3	Arterial TH 2500 v/h	NB	L	0:00:36	0:00:29	0:00:19	0:00:18	0:00:25
			TH	0:00:20	0:00:23	0:00:17	0:00:17	0:00:22
			R	0:00:22	0:00:27	0:00:18	0:00:18	0:00:22
		SB	L	0:00:27	0:00:29	0:00:31	0:00:18	0:00:26
			TH	0:00:18	0:00:21	0:00:20	0:00:17	0:00:20
			R	0:00:14	0:00:19	0:00:14	0:00:13	0:00:16
		EB	L	0:01:06	0:00:25	0:00:35	0:01:27	0:00:24
			TH-EXT	0:00:23	0:00:30	0:02:12	0:01:58	0:02:00
			TH-INT	0:00:10	0:00:07	0:00:07	0:00:10	0:00:05
			R	0:00:00	0:00:00	0:00:08	0:00:07	0:00:06
		WB	L	0:01:50	0:00:26	0:00:16	0:02:14	0:00:27
			TH-EXT	0:00:24	0:00:24	0:00:22	0:01:12	0:01:40
			TH-INT	0:00:07	0:00:05	0:00:04	0:00:08	0:00:04
			R	0:00:00	0:00:00	0:00:00	0:00:05	0:00:07
		System		**0:00:32**	**0:00:29**	**0:00:51**	**0:01:06**	**0:01:07**

(Continued in the next page)

(Table 7.13 Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III		TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	Optimal Three – Phase Signal Plan	
4	Arterial TH 2700 v/h	NB	L	0:00:37	0:00:25	0:00:20	0:01:00	0:01:00
			TH	0:00:22	0:00:24	0:00:18	0:00:31	0:00:31
			R	0:00:25	0:00:26	0:00:18	0:00:19	0:00:19
		SB	L	0:00:27	0:00:25	0:00:41	0:00:28	0:00:28
			TH	0:00:20	0:00:22	0:00:20	0:00:27	0:00:27
			R	0:00:15	0:00:20	0:00:14	0:00:14	0:00:14
		EB	L	0:02:03	0:00:23	0:00:38	0:03:08	0:03:08
			TH-EXT	0:00:24	0:00:19	0:02:32	0:02:26	0:02:26
			TH-INT	0:00:09	0:00:05	0:00:07	0:00:15	0:00:15
			R	0:00:00	0:00:00	0:00:10	0:00:11	0:00:11
		WB	L	0:02:05	0:00:23	0:00:23	0:02:41	0:02:41
			TH-EXT	0:00:23	0:00:19	0:00:36	0:01:45	0:01:45
			TH-INT	0:00:07	0:00:04	0:00:05	0:00:11	0:00:11
			R	0:00:00	0:00:00	0:00:01	0:00:09	0:00:09
		System		**0:00:34**	**0:00:23**	**0:01:01**	**0:01:26**	**0:01:26**

(Continued in the next page)

(Table 7.13 Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III		TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	Optimal Three – Phase Signal Plan	
5	Arterial TH 3000 v/h	NB	L	0:01:02	0:00:29	0:00:28	0:00:18	0:00:26
			TH	0:00:30	0:00:26	0:00:18	0:00:46	0:00:23
			R	0:00:32	0:00:29	0:00:19	0:00:21	0:00:22
		SB	L	0:00:44	0:00:28	0:00:54	0:00:18	0:00:24
			TH	0:00:26	0:00:24	0:00:20	0:00:38	0:00:21
			R	0:00:22	0:00:21	0:00:14	0:00:14	0:00:16
		EB	L	0:02:30	0:00:27	0:00:38	0:01:15	0:00:30
			TH-EXT	0:00:25	0:00:18	0:02:37	0:02:12	0:02:25
			TH-INT	0:00:09	0:00:05	0:00:08	0:00:14	0:00:06
			R	0:00:00	0:00:00	0:00:13	0:00:11	0:00:10
		WB	L	0:02:18	0:00:28	0:00:37	0:01:00	0:00:29
			TH-EXT	0:00:24	0:00:18	0:01:22	0:01:45	0:02:10
			TH-INT	0:00:08	0:00:04	0:00:06	0:00:13	0:00:05
			R	0:00:00	0:00:00	0:00:05	0:00:10	0:00:10
		System		**0:00:40**	**0:00:24**	**0:01:16**	**0:01:21**	**0:01:22**

(Continued in the next page)

(Table 7.13 Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III		TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	Optimal Three – Phase Signal Plan	
6	Arterial TH 3300 v/h	NB	L	0:01:09	0:00:30	0:00:29	0:00:18	0:00:25
			TH	0:00:30	0:00:27	0:00:19	0:00:45	0:00:23
			R	0:00:33	0:00:30	0:00:19	0:00:21	0:00:22
		SB	L	0:00:54	0:00:29	0:00:54	0:00:18	0:00:28
			TH	0:00:27	0:00:25	0:00:20	0:00:39	0:00:22
			R	0:00:23	0:00:23	0:00:14	0:00:14	0:00:16
		EB	L	0:02:58	0:00:29	0:00:37	0:01:15	0:00:30
			TH-EXT	0:00:27	0:00:17	0:02:36	0:02:08	0:02:24
			TH-INT	0:00:09	0:00:05	0:00:08	0:00:14	0:00:05
			R	0:00:00	0:00:00	0:00:13	0:00:11	0:00:11
		WB	L	0:02:46	0:00:34	0:00:35	0:01:00	0:00:30
			TH-EXT	0:00:26	0:00:18	0:01:44	0:01:43	0:02:12
			TH-INT	0:00:08	0:00:04	0:00:06	0:00:12	0:00:05
			R	0:00:00	0:00:00	0:00:09	0:00:10	0:00:09
		System		**0:00:43**	**0:00:24**	**0:01:21**	**0:01:20**	**0:01:32**

7.3.4 Case C: Varying Demand on Ramps and Arterials

Given O-D volume in Figure 7.4, PASSER III has optimal signal plan in Table 7.14 and TRANSYT-7F has optimal three-phase signal plan in Table 7.15.

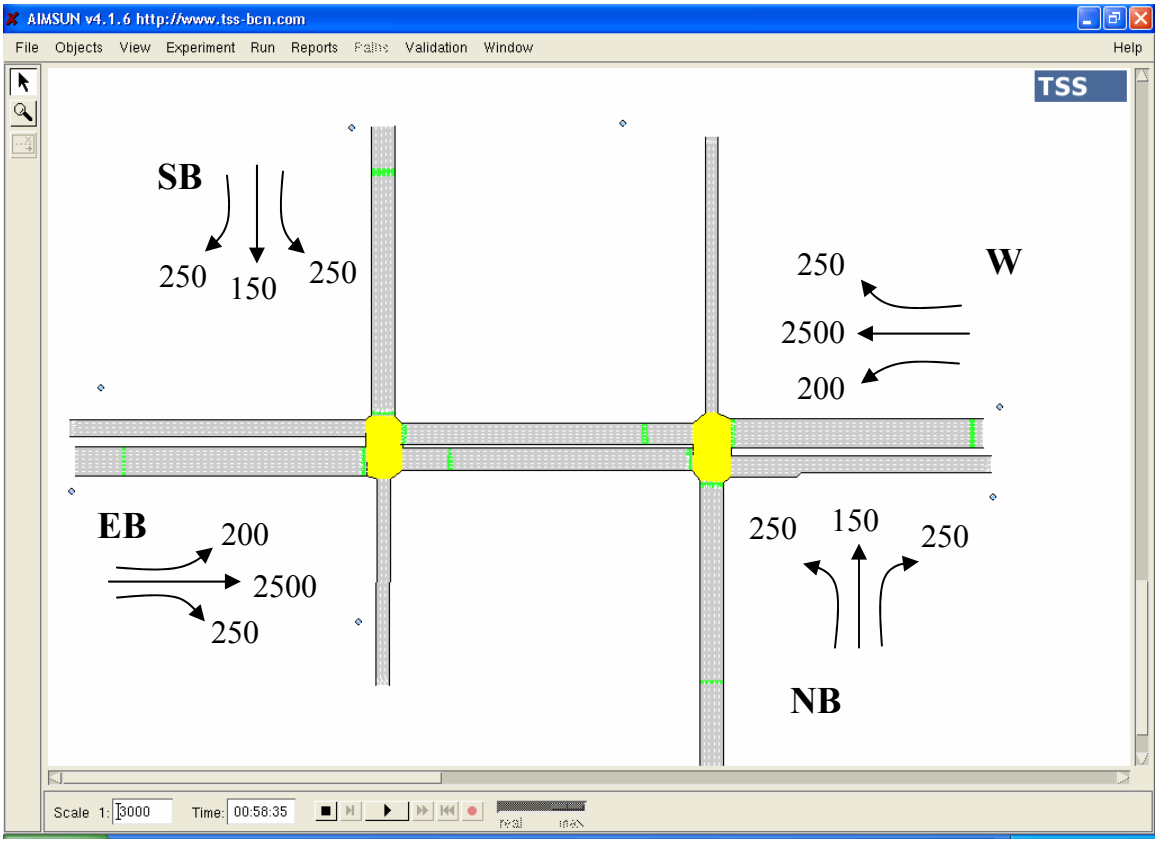


Figure 7.4 O-D Demand for Case C: Varying Demand on Ramps and Arterials

Table 7.14 PASSER III Optimal Signal Plan

C = 60s, Offset = 19s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Phase 1		38		11
Phase 2		11		37
Phase 3		11		12

Table 7.15 TRANSYT-7F Optimal Signal Plan

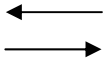
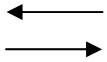
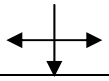
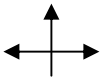
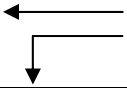
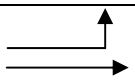
C=65s, Offset= 63s				
	Movements	Phase Duration (s)	Movements	Phase Duration (s)
Phase 1		36s		36s
Phase 2		14s		14s
Phase 3		15s		15s

Table 7.16 presents the O-D demand on each 15 minutes for Case C. There are seven demand scenarios designed in this study based on varying demand situations on arterials and ramps. These scenarios are named as “similar to optimization demand”, “higher NB and SB”, etc by comparing hourly simulation to the optimization demand. For all scenarios, the ramp volumes increase by 20 percent - 100 percent and the arterial volumes increase by 12 percent - 20 percent from the off-line demand.

Table 7.16 O-D Demand of Each Scenario for Case C: Varying Demand on Arterials and Ramps

#	Scenario	Movement Type			Demand (veh/h)					Optimization Demand (1h)
					0-15m	15-30m	30-45m	45-60m	Total (1h)	
1	Similar to Optimization Demand	Ramp	NB or SB	L	100	60	50	40	250	250
				TH	40	50	40	30	160	150
				R	100	60	50	40	250	250
		Arterial	EB or WB	L	50	50	50	50	200	200
				TH	900	800	400	400	2500	2500
				R	65	60	65	60	250	250
2	Higher NB and SB	Ramp	NB or SB	L	250	100	100	50	500	250
				TH	80	70	75	75	300	150
				R	250	100	100	50	500	250
		Arterial	EB or WB	L	50	50	50	50	200	200
				TH	900	800	400	400	2500	2500
				R	65	60	65	60	250	250
3	Higher on Each Approach	Ramp	NB or SB	L	150	100	100	50	400	250
				TH	50	70	65	65	250	150
				R	150	100	100	50	400	250
		Arterial	EB or WB	L	100	70	80	50	300	200
				TH	900	900	600	600	3000	2500
				R	65	60	65	60	250	250
4	Higher on Each Approach	Ramp	NB or SB	L	200	100	100	50	450	250
				TH	80	70	75	75	300	150
				R	200	100	100	50	450	250
		Arterial	EB or WB	L	50	50	50	50	200	200
				TH	900	900	600	600	3000	2500
				R	65	60	65	60	250	250

(Continued in the next page)

(Table 7.16: Continued from the previous page)

5	Higher on NB and SB	Ramp	NB or SB	L	200	100	50	50	400	250
				TH	50	70	65	65	250	150
				R	200	100	50	50	400	250
		Arterial	EB or WB	L	50	50	50	50	200	200
				TH	900	800	400	400	2500	2500
				R	65	60	65	60	250	250
6	Higher on Each Approach	Ramp	NB or SB	L	200	100	100	100	500	250
				TH	50	50	50	50	200	150
				R	200	100	50	50	400	250
		Arterial	EB or WB	L	65	60	65	60	250	200
				TH	800	900	550	550	2800	2500
				R	100	100	50	50	300	250
7	Higher on Each Approach	Ramp	NB or SB	L	200	200	100	100	500	250
				TH	50	50	50	50	200	150
				R	200	100	50	50	400	250
		Arterial	EB or WB	L	65	60	65	60	250	200
				TH	800	900	550	550	2800	2500
				R	100	100	50	50	300	250

Table 7.17 shows the delay of each optimization tool for the comparison case C where the demands on arterials and ramps vary. When the simulation demand is similar to the one used for off-line optimization, all optimization methods have produced similar results except the one using dynamic weights. When the ramp demands increase by 0 percent - 100 percent (i.e., 0 veh/h to 500 veh/h) and arterials increase by 12 percent - 20 percent (300-500 veh/h) from the off-line demand, the delay on ramp left turns, in the case of PASSER III, will be substantially increased, from 25 s/veh to several mins/veh (over-saturated). The results from TRANSYT-7F are similar to the one from PASSER III, but with lower delay on ramps in the same scenario.

On the other hand, the interchange under the control of the DP algorithm experiences the similar delay for all scenarios. This shows that the DP algorithm, using either fixed weights or dynamic weights, has the capability to adjust the signal plan to accommodate varying demand scenarios. From the results, it can be observed that dynamic weights would not cause higher delay when the demand increases considerably. However, in the low to medium demand scenarios, dynamic weights may also not be able to reduce the delay significantly. One possible reason might be the more lost time due to the frequent change of the signal. With dynamic weights, operations remain under-saturated for higher demands than with fixed weights.

Table 7.17 A Comparison of Delay (sec/veh) for Case C: Varying Demand on Arterials and Ramps

#	Scenario	Movement Type		Real-time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	
1	Similar to Optimization Demand	NB	L	0:00:40	0:01:29	0:00:25	0:00:25
			TH	0:00:33	0:01:10	0:00:25	0:00:27
			R	0:00:31	0:01:27	0:00:35	0:00:28
		SB	L	0:00:40	0:01:19	0:00:29	0:00:25
			TH	0:00:33	0:01:08	0:00:24	0:00:26
			R	0:00:25	0:01:05	0:00:21	0:00:20
		EB	L	0:00:16	0:00:43	0:00:17	0:00:22
			TH-EXT	0:00:12	0:00:09	0:00:08	0:00:12
			TH-INT	0:00:05	0:00:03	0:00:02	0:00:08
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:17	0:00:44	0:00:17	0:00:26
			TH-EXT	0:00:11	0:00:08	0:00:09	0:00:12
			TH-INT	0:00:04	0:00:02	0:00:01	0:00:05
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:16**	**0:00:25**	**0:00:13**	0:00:18

(Continued in the next page)

(Table 7.17: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	
2	Higher NB and SB	NB	L	0:00:41	0:00:43	0:05:00	0:02:49
			TH	0:00:21	0:00:40	0:02:57	0:00:56
			R	0:00:21	0:00:38	0:03:34	0:02:08
		SB	L	0:00:31	0:00:41	0:06:15	0:02:26
			TH	0:00:20	0:00:38	0:04:19	0:00:49
			R	0:00:15	0:00:29	0:04:11	0:01:49
		EB	L	0:00:25	0:01:00	0:00:18	0:00:22
			TH-EXT	0:00:20	0:00:12	0:00:08	0:00:12
			TH-INT	0:00:08	0:00:06	0:00:02	0:00:08
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:22	0:00:53	0:00:18	0:00:26
			TH-EXT	0:00:22	0:00:13	0:00:09	0:00:12
			TH-INT	0:00:08	0:00:05	0:00:01	0:00:05
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:25**	**0:00:26**	**0:01:33**	0:00:49

(Continued in the next page)

(Table 7.17: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	
3	Higher on Each Approach	NB	L	0:00:42	0:01:00	0:01:47	0:00:33
			TH	0:00:28	0:00:55	0:00:29	0:00:27
			R	0:00:31	0:00:56	0:00:48	0:00:34
		SB	L	0:00:47	0:00:49	0:03:20	0:00:28
			TH	0:00:27	0:00:51	0:00:30	0:00:26
			R	0:00:21	0:00:43	0:00:48	0:00:23
		EB	L	0:00:36	0:00:30	0:00:23	0:00:23
			TH-EXT	0:00:20	0:00:13	0:00:09	0:00:12
			TH-INT	0:00:06	0:00:05	0:00:02	0:00:09
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:43	0:00:36	0:00:22	0:00:26
			TH-EXT	0:00:21	0:00:13	0:00:09	0:00:12
			TH-INT	0:00:06	0:00:04	0:00:01	0:00:05
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:25**	**0:00:25**	**0:00:26**	0:00:21

(Continued in the next page)

(Table 7.17: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	
4	Higher on Each Approach	NB	L	0:00:45	0:00:52	**0:03:44**	0:01:52
			TH	0:00:27	0:00:51	0:01:15	0:00:28
			R	0:00:26	0:00:45	0:01:56	0:00:58
		SB	L	0:00:53	0:00:48	**0:05:30**	0:01:13
			TH	0:00:26	0:00:44	**0:02:06**	0:00:27
			R	0:00:20	0:00:36	0:02:25	0:00:46
		EB	L	0:00:27	0:00:53	0:00:19	0:00:22
			TH-EXT	0:00:16	0:00:11	0:00:09	0:00:12
			TH-INT	0:00:08	0:00:05	0:00:02	0:00:09
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:29	0:01:05	0:00:18	0:00:27
			TH-EXT	0:00:18	0:00:12	0:00:09	0:00:12
			TH-INT	0:00:07	0:00:04	0:00:01	0:00:05
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:23**	**0:00:25**	**0:00:53**	0:00:29

(Continued in the next page)

(Table 7.17: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	
5	Higher on NB and SB	NB	L	0:00:36	0:00:51	**0:03:53**	0:01:54
			TH	0:00:27	0:00:56	**0:01:08**	0:00:29
			R	0:00:27	0:00:50	0:02:24	0:01:03
		SB	L	0:00:35	0:00:44	**0:04:41**	0:01:11
			TH	0:00:26	0:00:50	**0:02:06**	0:00:28
			R	0:00:17	0:00:35	**0:02:58**	0:00:52
		EB	L	0:00:22	0:00:24	0:00:17	0:00:22
			TH-EXT	0:00:16	0:00:14	0:00:08	0:00:12
			TH-INT	0:00:08	0:00:05	0:00:02	0:00:08
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:20	0:00:22	0:00:17	0:00:26
			TH-EXT	0:00:17	0:00:16	0:00:09	0:00:12
			TH-INT	0:00:07	0:00:05	0:00:01	0:00:05
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:21**	**0:00:27**	**0:00:56**	**0:00:29**

(Continued in the next page)

(Table 7.17: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	
6	Higher on Each Approach	NB	L	0:00:40	0:01:06	0:03:23	0:01:17
			TH	0:00:22	0:00:46	0:00:37	0:00:28
			R	0:00:25	0:00:49	0:02:23	0:01:00
		SB	L	0:00:37	0:01:01	0:05:12	0:01:00
			TH	0:00:20	0:00:41	0:01:57	0:00:28
			R	0:00:17	0:00:35	0:02:29	0:00:50
		EB	L	0:00:23	0:00:45	0:00:20	0:00:22
			TH-EXT	0:00:17	0:00:11	0:00:08	0:00:12
			TH-INT	0:00:07	0:00:05	0:00:02	0:00:08
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:23	0:00:51	0:00:18	0:00:26
			TH-EXT	0:00:17	0:00:11	0:00:09	0:00:12
			TH-INT	0:00:06	0:00:04	0:00:01	0:00:05
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:21**	**0:00:25**	**0:00:51**	**0:00:27**

(Continued in the next page)

(Table 7.17: Continued from the previous page)

#	Scenario	Movement Type		Real-time DP		PASSER III	TRANSYT-7F
				Fixed Weight 2 Ramp 2 Ext 1 Int	Dynamic Weight	Optimal Signal Plan	
7	Higher on Each Approach	NB	L	0:00:27	0:00:55	0:03:31	0:01:17
			TH	0:00:18	0:00:35	0:00:36	0:00:28
			R	0:00:17	0:00:35	0:02:23	0:01:00
		SB	L	0:00:52	0:01:13	0:06:34	0:02:19
			TH	0:00:17	0:00:35	0:01:58	0:00:30
			R	0:00:12	0:00:27	0:02:03	0:00:51
		EB	L	0:00:16	0:00:30	0:00:21	0:00:22
			TH-EXT	0:00:26	0:00:13	0:00:08	0:00:12
			TH-INT	0:00:08	0:00:05	0:00:02	0:00:08
			R	0:00:00	0:00:00	0:00:00	0:00:00
		WB	L	0:00:18	0:00:37	0:00:18	0:00:26
			TH-EXT	0:00:26	0:00:13	0:00:09	0:00:12
			TH-INT	0:00:07	0:00:04	0:00:01	0:00:05
			R	0:00:00	0:00:00	0:00:00	0:00:00
		System		**0:00:28**	**0:00:24**	**0:00:55**	**0:00:31**

7.4 Summary

The proposed DP algorithm is a real-time signal control that can adjust the signal plan in response to demand fluctuation while the optimal signal plans from PASSER III and TRANSYT-7F are off-line determined and are not adaptive to the field traffic information. As the field demand can vary from time to time, which is different from the one used in off-line optimization, the optimal signal plans by PASSER III and TRANSYT-7F cannot accommodate large variations in real demand scenarios.

This study has simulated the signal plans from all three tools in AIMSUN by applying various demand scenarios, i.e., low demand and varying demand, to determine their performances and capabilities in different situations. The following conclusions have been drawn from the above comparisons:

1. When the simulation demand is similar to the one in optimization and the demand does not vary considerably in each 15 minutes, all three methods exhibit nearly identical performance in terms of the system delay and each movement delay.

2. When ramp demands during the simulation increase by 6 percent to 67 percent (10 to 150 veh/h) based on the off-line optimization, PASSER III and TRANSYT-7F have shown much higher ramp delays (e.g., 1 minute) than the DP algorithm. Similarly, in the case that both arterial demands and ramp demands increase during the simulation, i.e., ramps increase by 0 percent - 100 percent (0-250 veh/h), arterials increase by 12 percent - 20 percent (300-500 veh/h), both PASSER III and TRANSYT-7F cause longer delays either on the arterial or the ramp than the DP on-line system. In particular, PASSER III could lead to higher delays up to several minutes than other optimization tools. The study has shown that PASSER III results are more sensitive to demand fluctuation. However, this finding needs to be further tested using more simulation data.

3. When arterial demand substantially increases, i.e., increase by 65 percent from 2000 veh/h to 3300 veh/h, the PASSER III and TRANSYT-7F signals exhibit much larger delays than that of the DP algorithm. Both PASSER III and TRANSYT-7F signals increases the system delay up to 1 minute or more. The interchange under the control of the proposed real-time algorithm does not experience delay as much as off-line systems do. The fixed weighted algorithm increases delay 22s while the dynamic weighted one maintains delay between 23s and 29s.

4. Compared to the one using fixed weights, the signal by dynamic weights behaves better in terms of handling increased demand situations. Taking case B as an example, when the arterial demand increases by 60 percent (from 2000 veh/h to 3300 veh/h), using fixed weights result in the delay from 21s to 43s while using dynamic weights has the delay varied from 23s to 29s. The dynamic weighted algorithm can reduce the system delay up to 32.5 percent.

5. Using dynamic weights does not significantly reduce the delay in a relatively low demand, compared to fixed weights. However, it will also not considerably increase the delay when the demands increase. With dynamic weights, operations remain under-saturated for higher demands than with fixed weights.

CHAPTER 8
SUMMARY, CONCLUSIONS AND FUTURE RESEARCH

8.1 A Summary of the Thesis Research

A critical and extensive literature review of the research in the field of the traffic signal control and optimization has been conducted. It has been understood that PASSER III is the only existing signal optimization model for diamond interchanges. It optimizes the pre-timed signal plan based on off-line demand and cannot adapt itself to fluctuating demand situations. Moreover, its optimal signal plan is chosen among a restricted number of alternatives and it is not a globally optimum solution. Therefore, there is a need to develop an adaptive and optimal signal control of diamond interchanges by considering real time traffic flow conditions. OPAC and RHODES are two promising adaptive systems that are based on dynamic programming and operate using the rolling horizon concept. Both systems have some limitations. OPAC does not actually use a dynamic programming solution procedure rather than a restricted optimal sequential constrained search (OSCO). Consequently, the OPAC optimization methodology should be classified as a trial-and-error enumeration that cannot guarantee a globally optimum solution. RHODES requires a fixed sequence of phases and a longer projection time. The two formulations cannot be applied to optimize phase sequence under phase constraints that are involved in the control for diamond interchanges. Moreover, both systems take into account a future horizon greater than 20 seconds. Thus, the arrival pattern used for entire optimization horizon may not be reliable since it is not possible or practical to place detectors far enough upstream to provide the desired amount of information.

This research develops a methodology and the corresponding implementation algorithm to provide optimal signal control of diamond interchanges in response to real-time traffic fluctuations. The problem is formulated to find a phase sequencing decision with phase duration that makes a pre-specified performance measure minimized over a finite horizon that rolls forward. Dynamic programming (DP) is applied as the optimization method in this research because it offers great computational savings over

exhaustive enumerations in decision-making processes. The performance measure can be, for example, total delay, queue lengths, number of stops, or any combination of these. The decision variables are the sequence of signal phases and their duration. A horizon of 10 seconds is divided into an integer number of intervals, each having 2.5 seconds. The algorithm finds the optimal signal switches by minimizing the selected performance index over each horizon, and the optimization process proceds one horizon after another. This DP problem is solved by forward value iterations in the algorithm. As the DP optimization operates on advanced vehicle information over each horizon, a dynamic model of future vehicular detections, arrivals, and departures is developed at the microscopic level based on loop detectors set back a certain distance from the stop-line. This model accounts for the discrete-event dynamics of the signal switches, vehicle arrival and discharging. With this model the traffic flows at the stop-line can be precisely predicted for a horizon when the DP optimization operates on.

The DP algorithm is coded in C++ language and dynamically linked to the AIMSUN, a stochastic micro-simulation package, which is used for evaluation and simulation of the developed methodology. In this study AIMSUN simulates a signalized diamond interchange instrumented with loop detectors that can provides vehicle counts and speeds to the DP algorithm. Based on this, the algorithm calculates the optimal phase sequence and the duration of each phase, and passes them back to AIMSUN, which subsequently controls the interchange in real time. The communication between the AIMSUN and the DP algorithm is achieved through a number of user-defined extension functions. To enable the algorithm to implement practical scenarios, a so-called majority rolling technique was developed. This technique can be stated as follows: over a single horizon of 10 seconds the signal phase being chosen most frequently is implemented for 7.5 seconds followed by a yellow-and-all-red clearance time of 2.5 second; and if two consecutive horizons hold same majority signal then no clearance time is inserted between them.

A sensitivity analysis using simulation results is conducted to study the characteristics of the proposed DP algorithm including how the delay of an interchange is

affected by the different definitions of performance measure index (PMI) and the weight associated with each approach as well as the performance of the algorithm under various demand scenarios. The results have shown that queue length and storage ratio defined performance measures are the best ones in minimizing system delays. A general rule of choosing the weight of an approach is that a larger weight is applied for approaches having more demand. The study has also demonstrated the benefits of using dynamic weights without manually requiring the changing of weights. The study has also demonstrated the benefits of using dynamic weights without manually requiring the changing of weights. Dynamic weights can reduce system delay by 36 percent – 49 percent than fixed weights when the demand varies unpredictably every 15 minutes and is unbalanced. With dynamic weights, operations remain under-saturated for higher demands than with fixed weights. Moreover, the real-time DP algorithm has revealed the capability to accommodate various demand situations.

The proposed real-time adaptive algorithm has also been compared to two off-line optimization packages: PASSER III and TRANSYT-7F, with respect to a calibrated diamond interchange. The optimized pre-timed signal plans from TRANSYT-7F and PASSER III are implemented in AIMSUN and the results are compared to those from the DP algorithm. The comparison study covers various demand scenarios including low, medium, high, and time varying demand. The results have exhibited that the proposed real-time DP signal algorithm is superior to PASSER III and TRANSYT-7F in handling demand fluctuations for medium to high flow scenarios when the field demand is increased from the one used in off-line optimization. In the low demand scenario, the real-time DP algorithm performs nearly identically to PASSER III and TRANSYT-7F.

8.2 Contributions and Conclusions

The major unique contributions this research has made are summarized as follows:

1. The adaptive signal optimization problem has been formulated in a decision network with phase sequence and duration as decision variables. The forward recursive DP

solution procedure has been developed in the context of real-time optimization. A number of performance measures have been defined and used as the objective functions for the signal optimization based on the detected vehicle information.

2. The developed real-time adaptive signal algorithm can optimize both phase sequence and phase duration by minimizing a user pre-specified performance measure over a finite horizon that rolls forward.

3. Traffic dynamics of the vehicles detected at the detectors, arrived and then stopped or discharged at the stop-lines at each movement, have been modeled and an interactive projection scheme has been developed at the microscopic level.

4. The proposed algorithm that incorporates the DP procedure, the projection of the detected vehicles, and the arrival-discharge dynamics has been implemented in C++. It has also dynamically linked to AIMSUN, which has made possible the real-time simulation of diamond interchanges using the optimized signals under different demand scenarios.

5. Three types of sensitivity analysis were conducted with respect to how the delay of the chosen interchange is affected by the optimized signals using different types of performance measure index (PMI), applying the weight on each approach, and varying demand scenarios in the DP algorithm, respectively. Important findings are:
   a. In medium-to-high and high demand scenarios, PMI 1: "Sum of Average Queue Length Per Lane For All Approaches" and PMI 4: "Sum of Storage Ratio for All Approaches" can result in a signal with which the interchange operation can produce a lower system delay than it does with the signal due to the use of other user-defined performance measures. However, in the low and low-to-medium demand situations, there is no big difference in the results by using different PMIs.
   b. A larger weight can usually be applied to the approaches having higher demand, but, this would slow the vehicles' discharge on other approach(es). Hence, a

proper adjustment of the weight on each approach needs to be made to achieve a satisfactory overall performance of the interchange.

c. The real-time DP algorithm has demonstrated the capability to accommodate various demand situations that include high ramp volume together with one high arterial volume, two high ramp volumes, and two high arterial volumes with other approaches having medium to high volume. Both the signals obtained from the use of fixed and dynamic weights have produced small range of system delay variation, from 20 seconds to 33 seconds, under the above scenarios.

6. Various comparisons of three optimization methods, i.e., the developed real-time DP algorithm, PASSER III and TRANSYT-7F have been made. Important findings are:
   a. When the field demand is similar to the demand for off-line optimization and it varies little in each 15 minutes period, all three methods exhibit nearly identical performance in terms of the system delay and each movement delay.
   b. When the simulation demand varies in each 15 minutes, and is higher than the demand for off-line optimization (e.g., ramp demands increase 6 percent to 67 percent), PASSER III and TRANSYT-7F signals produce much higher ramp delays (e.g., 1 minute) than the DP algorithm signal does. In the case that ramp demands increase 0 percent – 100 percent, arterials increase 12 percent – 20 percent from the off-line demand, both the PASSER III and TRANSYT-7F signals cause much longer delays either on the arterial or the ramp than the DP on-line system, in particular, the PASSER III signal could generate ramp delays up to several minutes.

7. The comparisons of the DP algorithm using dynamic weight and fixed weight were performed under many demand scenarios including varying unbalanced and balanced demands. Major findings are:
   a. When the demand varies unpredictably every 15 minutes and is unbalanced, using dynamic weights reduces the system delay by 36 percent – 49 percent, compared to using fixed weights.

b. The algorithm using dynamic weights can perform better than the one using fixed weights in handling increased demand situations on one or more approaches. An example in the study has shown that when arterial demand increases by 60 percent (from 2000 veh/h to 3300 veh/h), dynamic weights can reduce the system delay up to 32.5 percent, compared to using fixed weights.
c. In the low to medium demand scenarios, dynamic weights may not be able to significantly reduce delay compared to fixed weights, but it will also not considerably increase delay when the demands increase. With dynamic weights, operations remain under-saturated for higher demands than with fixed weights.
d. With dynamic weights, users do not need to manually adjusting the weights.
e. The performance of dynamic weights also depends on how the values of dynamic weights are defined. It is needed to calibrate these criteria in order to achieve greatest performance.

To conclude, the proposed signal optimization based on DP is adaptive and responsive to real-time traffic fluctuations at microscopic level, and its methodology and algorithm implementation have been fully developed. The signals by the developed algorithm behave nearly identical to the signals by the popular PASSER III and TRANSYT-7F models in the low-to-medium demand scenarios. Importantly, the former perform better over the latter in medium-to-high demand scenarios, particularly when the dynamic weights method is used. The dynamic weighted DP algorithm is appropriate to be applied in special events or incidents when high demands are unexpected and varying.

8.3 Future Research

Upon the completion of the study, the following has been identified and recommended for future research. They are:

1. The current decision network formulated for the signal optimization of a diamond interchange is based on three-phase ring structure. It needs to be extended to include every possible combination of phases for diamond interchange traffic operations. In particular, the phase moving vehicles on a ramp together with an

internal link should be incorporated. The absence of this phase is acknowledged in the present algorithm by assigning weights on ramp approaches in order to get ramp vehicles being discharged efficiently.

2. The developed algorithm requires a detector set upstream of the stop-line on each approach of a diamond interchange. When it is not possible or practical to place detectors far enough from the intersection to provide the reliable vehicular information, for example at tight interchange configurations, the use of predicted traffic data are suggested, and the traffic flows at the internal link are to be obtained from detected traffic information at external bounds.

3. The developed interactive model for projecting the detected vehicles onto the stop-line is critical to the proposed signal optimization method. Some of the assumptions, such as average vehicle speed of 30-35mph between the detector and stop-line, should be revisited in order to refine the algorithm performance.

4. The methodology developed in this study considers a diamond interchange of two signalized intersections. It is suggested to study the feasibility of the developed algorithm to be used for adaptive signal control of an urban arterial and corridor that involves multiple signals.

BIBLIOGRAPHY

Bang, K.L., "Optimal control of Isolated Traffic Signals", *Traffic Engineer and Control*, Vol.17, No.7, 1976, pp-288-292.

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

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

Charles, P., "Time for a Change", *Traffic Technology International*, June/July, 2001, pp. 68-71

Dunne, M.C., and Potts, R.B., "Algorithm for Traffic Control", *Operation Research*, Vol. 12, 1964, pp.870-881

Elefteriadou, L., J.D. Leonard II, G. List, H. Lieu, M. Thomas, R. Giguere, G. Johnson, R. Brewish, "Beyond the Highway Capacity Manual – Framework for Selecting Simulation Models in Traffic Operational Analyses", *Transportation Research Record (1687),* Transportation Research Board, Washington, D.C., 1997, pp.96-106

Elefteriadou, L, Fang C.F., Mason, J.M., et al, "Capacity and Quality of Service of Interchange Ramp Terminals", Interim Report, NCHRP 3-60, February, 2002

Fambro, D.B., N.A. Chaudhary, J.A. Bonneson, C.J. Messer, and L.L. Arabie, "PASSER III –90 Users Manual and Application Guide", Texas Transportation Institute, College Station, TX, March 1991, 104 pages

Federal Highway Administration, "Traffic Software Integrated System 97 User's Guide", 1997

Gartner, N. H., "Microscopic Analysis of Traffic Flow Patterns for Minimizing Delay on Signal-Controller Links", HRB, Highway Research Record 445, 1973, pp. 12-23

Gartner N. H., "OPAC: A Demand-Responsive Strategy for Traffic Signal Control", *Transportation Research Record (906)*, Transportation Research Board, Washington, DC, 1983, pp. 75-81

Gartner, N.H. and Pooran, F.J., "Implementation and Field Testing of the OPAC Adaptive Control Strategy in RT-TRACS", paper presented at *the 81st Annual TRB Meeting*, Washington DC, January 2002

Hastings, N.A.J., "Dynamic Programming with Management Applications", Butter Worths, London, 1973, 173pp

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

Head, K.L., "Event-Based Short-Term Traffic Flow Prediction Model", *Transportation Research Record (1510),* Transportation Research Board, Washington, D.C., 1995, pp.45-52

Henry, J.J., Farges, J.L., and Tuffa, J., "The PRODYN real time traffic algorithm", *4th IFAC-IFIP-IFORS Conference on "Control in Transportation Systems*", Baden-Baden. 1983, pp.305-310

Hicks, B. and Carter, M., "What Have We Learned About ITS? Arterial Management", at http://www.itsdocs.fhwa.dot.gov/jpodocs/repts_te/@9z01!.PDF, 2001

Hillier F.S., and Lieberman, G. J., “Introduction to Operation Research”, Chapter 11 Dynamic Programming, 7th edition, McGraw Hill, New York, NY, 10020, 2001

Hunt, P.B., et al., "The SCOOT on-line-traffic-signal optimization technique" *Traffic Engineering and Control*, Vol.23, No.4, 1982, pp.190-192.

Hansen, B.G., Martin, P.T. and Perrin, H.J., “SCOOT Real-Time Adaptive Control in a CORSIM Simulation Environment”, *Transportation Research Record (1727),* Transportation Research Board, Washington, D.C., 2000, pp.27-31

ITE (Institute of Transportation Engineering), “Traffic Control Systems Handbook”, ITE publication No. LP-123, 1985

Koonce, P., Urbanki II, T., and Bullock, D., “Evaluation of Diamond Interchange Signal Controller Settings by Using Hardware-in-the-Loop Simulation”, *Transportation Research Record (1683),* Transportation Research Board, Washington, D.C., 1999, pp.59-66

Kronborg, P., Davidson, F., “MOVA and LHOVRA: Traffic signal control for isolated intersections”, *Traffic Engineering and Control*, 1993, pp.195-200.

Krueger, G.D., "Development and Analysis of A Flexible Signal Phasing Strategy for Diamond Interchange Control", MSc Thesis, Texas A&M University, 1995, 125pp

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

Lin, F.B, and Vijayayumar, S., “Adaptive signal control at isolated intersections”, *ASCE Journal of Transportation Engineering*, Vol. 114, No. 5, September 1988, pp.555-573

Luk, J.Y.K., "Two Traffic Responsive Area Traffic Control Methods: SCATS and SCOOT", *Traffic Engineering and Control*, Vol. 30. 1984, pp.14-20

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

Messer, C. J., and J.A. Bonneson, "NCHRP Project 3-47 Final Report: Capacity Analysis of Interchange Ramp Terminal", *NCHRP Web Document 12*, TRB, National Research Council, Washington, D.C., April 1997, 335 pp. (Available at http://www.nap.edu/readingroom)

Michalopoulos, P.G., and Stephanopoulos, G., "An Algorithm for Real-time Control of Critical Intersections", *Traffic Engineering & Control*, 1979, pp. 9-15

Mirchandani, P. and Head, L., "A Real-time Traffic Signal Control System: Architecture, Algorithms and Analysis", *Transportation Research,* Part C, 9 (2001) pp. 415-432

Miller, A. J., "Settings for Fixed-Cycle Traffic Signals", Operational Research Quarterly, Vol. 14, No. 4, 1963(a), pp.373-386

Miller, A.J., "A Computer Control Ssytem for Traffic Networks", 1963(b), pp. 200-220.

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

Pearce, V., "Real-time Remedy", *Traffic Technology International*, Dec 2000/Jan2001, pp. 61-63.

PTV, “VISSIM 3.5 User Manual”, Germany, 2000

Robertson, D. I., and Brethertn R. D., “Optimum Control of an Intersection for Any Known Sequence of Vehicle Arrival”, *Traffic Control and Transportation Systems*, Proceedings of the 2nd IFAC/IFIP/IFORS Symposium, Monte Carlo, edited by AFCET, 1974, pp. 57-69

Roess, R.P., McShane, W.R., and Prassas, E.S., “Traffic Engineering”, 2nd Edition, Pretice Hall, New Jersey, 1998

Roess, R.P., “Draft Materials for Analysis of Interchange Ramp Terminals for the HCM 2000”, *A3A10TRB Committee Correspondence*, 1999, 48 pages

Sen, U., and Head, K.L., “Controlled Optimization of Phases at an Intersection”, Transportation Science, vol. 31, No.1, February, 1997, pp5-17

TSS-Transportation Simulation System, “AIMSUN User Manual, Version 4.1”, Barcelona, Spain, 2002

TSS-Transportation Simulation System, “GETRAM Extensions User Manual, Version 4.1”, Barcelona, Spain, 2002

Webster F.V., “Traffic Signal Settings”, *Road Research Technical Paper,* No. 39, Road Research Laboratory, H.M.S.O., England, 1958

Venglar S. and Urbanik, T., “Real-time Optimization at Diamond Interchange”, *Proceedings, Real-Time Technology and Applications Symposium*, Chicago, Illinois, May 1995

Vincent, R.A., and Young, C. P., "Self-Optimizing Traffic Signal Control using Microprocessors – the TRRL "MOVA" Strategy for Isolated Intersections", *Traffic Engineering and Contro*l, Vol. 27, July/August 1986, pp. 385-387

Winsten, C.B., " Studies in the Economics of Transportation" by M. Beckmann, C.B. Mcguire and C.B. Winsten, Yale University Press, 1956, particularly pp. 40-42

APPENDIX
C++ CODE OF THE DP ALGORITHM

The appendix presents the C++ code (*.cpp) of the real-time DP algorithm. The *.cpp file can be loaded in Microsoft Visual C++ 6.0 to generate a dynamic link library (DLL) that builds the communication between AIMSUN simulation and the algorithm dynamically. The entire code consists of six routines defined by GETRAM extension module. They are GetExtLoad, GetExtInit, GetExtManage, GetExtPostManage, GetExtFinish, and GetExtUnLoad. Of these modules, GetExtManage is the major function. It is called in every simulation stop at the beginning of the cycle and used to request and update detector measures. The DP optimization and the model for vehicles projection, arrival and discharge are also embedded in this routine.

```
#include "GetExt.h"
#include "GetExt_common.h"
#include "AKIProxie.h"
#include "CIProxie.h"
#include <stdio.h>

float DPprojection(int j, int i, float qinitial[8], float nvd [28][8], float ahdw[8], float asvd[8],float ff0,float ff1,float qfinal[8]);

const float BrakingD=51.16;
const float DetectorD=656.0;
const float DT=2.5;
const float Deceleration=13.12;
const float MeanSpeed=45.0;
const unsigned short int nbdet=7;
const float TimeAdvance=16.0;
const float TimeStart=300.0;

char astring[128];
int isimustep,idp,idp1,idprolling;
int phase_optimal[4];
int pre_phase,igreen,majoritySignal;
int ihorizon;
float ff0,ff1,nvd[28][8],svd[28][8],hdw[28][8],asvd[8],ahdw[8],anvd[8],qinitial[8],qfinal[8], w[8];
int cphase1,cphase2,rphase1, rphase2,minphase;

float intervalEst, nextTimeStatis; // the initial values are set in GetExtInit

int GetExtLoad()
{
        AKIPrintString ("Getram is being initialised by AIMSUN ..");
        return 0;
}

int GetExtInit()
{
        int nbj,id,nbp,cbp;

        nbj=ECIGetNumberJunctions();
        for (int i=0; i<nbj; i++)
        {
        id=ECIGetJunctionId(i);
        nbp=ECIGetNumberPhases(id);
        cbp=ECIGetCurrentPhase(id);
        }

        isimustep=-1;  //initialising the Detection step. -1 is for the first step of 0.
        idprolling=0; // initial integer number of DP rolling done
        idp1=0;         // initial DP interval counts (ie, between 1 and 4)
        idp=0;
        ihorizon=0;
        intervalEst=60;                 // used for section statistics info collection
        nextTimeStatis=60;

        return 0;
}
```

```
int GetExtPostManage(float time, float timeSta, float timTrans, float acicle)
{
		return 0;
}
int GetExtFinish ()
{
		float System_Delay_Ave, System_Delay_Dev, System_Stoptime_Ave, System_Stoptime_Dev, System_Nstop;
		struct StructAkiEstadSystem Info_System;

		AKIPrintString("Finishing the AIMSUN and GetRam calculation ... ");
		AKIPrintString("Collecting System Statistics Information over the Whole Simulation ... ");
		Info_System=AKIEstGetGlobalStatisticsSystem(NULL);			// Statis System Info is collected over the Whole simulation period

		System_Delay_Ave=Info_System.DTa;	// Delay time - average
		System_Delay_Dev=Info_System.DTd;	// Delay time - deviation
		System_Stoptime_Ave=Info_System.STa;
		System_Stoptime_Dev=Info_System.STd;
		System_Nstop=Info_System.NumStops;

		return 0;
}
int GetExtUnLoad ()

{
		AKIPrintString ("Unload Getram Extension program ... ");
		return 0;
}

int GetExtManage(float time, float timeSta, float timTrans, float acicle)
{
  float qqinitial[8][5], qqtemp[8][5];
  float f[5][5],imin[5][5];
  float minf;
  int j1,phase1,phase2,phase4,phase12,phase5,phase6,phase8,phase56;
  int iopt, isvd, jnode;
		int nvsection;
 struct InfVeh InfVeh_Section;
 struct StaticInfVeh InfVeh_Split;

		if (time>=284.5-0.0001)
		{ iopt=0;
    if (idprolling==0)
{
				isimustep=isimustep+1;
				nvd[isimustep][0]=AKIDetGetCounterCycle("DD1",NULL)/2.0;
				svd[isimustep][0]=1.4667*AKIDetGetSpeedCycle("DD1",NULL);

				nvd[isimustep][1]=AKIDetGetCounterCycle("DD2",NULL)/6.0;
				svd[isimustep][1]=1.4667*AKIDetGetSpeedCycle("DD2",NULL);

				nvd[isimustep][2]=AKIDetGetCounterCycle("DD3",NULL)/4.0;
				svd[isimustep][2]=1.4667*AKIDetGetSpeedCycle("DD3",NULL);

				nvd[isimustep][3]=AKIDetGetCounterCycle("DD4",NULL)/4.0;
				svd[isimustep][3]=1.4667*AKIDetGetSpeedCycle("DD4",NULL);

				nvd[isimustep][4]=AKIDetGetCounterCycle("DD5",NULL)/2.0;
				svd[isimustep][4]=1.4667*AKIDetGetSpeedCycle("DD5",NULL);

				nvd[isimustep][5]=AKIDetGetCounterCycle("DD6",NULL)/6.0;
				svd[isimustep][5]=1.4667*AKIDetGetSpeedCycle("DD6",NULL);

				nvd[isimustep][6]=AKIDetGetCounterCycle("DD7",NULL)/4.0;
				svd[isimustep][6]=1.4667*AKIDetGetSpeedCycle("DD7",NULL);

				nvd[isimustep][7]=AKIDetGetCounterCycle("DD8",NULL)/4.0;
				svd[isimustep][7]=1.4667*AKIDetGetSpeedCycle("DD8",NULL);

		}

    if (idprolling>=1)  // if idprolling is not "0"
      {
		if (isimustep==0)
		{
				for (int i=0;i<13;i++)
				for (int j=0;j<8;j++)
				{
				nvd[i][j]=nvd[i+15][j];
				 svd[i][j]=svd[i+15][j];
				hdw[i][j]=hdw[i+15][j];
				}
		isimustep=12;
		}
		if (isimustep>=12)
		{
		isimustep=isimustep+1;
		nvd[isimustep][0]=AKIDetGetCounterCycle("DD1",NULL)/2.0;
		svd[isimustep][0]=1.4667*AKIDetGetSpeedCycle("DD1",NULL);
		nvd[isimustep][1]=AKIDetGetCounterCycle("DD2",NULL)/6.0;
		svd[isimustep][1]=1.4667*AKIDetGetSpeedCycle("DD2",NULL);
		nvd[isimustep][2]=AKIDetGetCounterCycle("DD3",NULL)/4.0;
		svd[isimustep][2]=1.4667*AKIDetGetSpeedCycle("DD3",NULL);
		nvd[isimustep][3]=AKIDetGetCounterCycle("DD4",NULL)/4.0;
```

```
                svd[isimustep][3]=1.4667*AKIDetGetSpeedCycle("DD4",NULL);
                nvd[isimustep][4]=AKIDetGetCounterCycle("DD5",NULL)/2.0;
                svd[isimustep][4]=1.4667*AKIDetGetSpeedCycle("DD5",NULL);
                nvd[isimustep][5]=AKIDetGetCounterCycle("DD6",NULL)/6.0;
                svd[isimustep][5]=1.4667*AKIDetGetSpeedCycle("DD6",NULL);
                nvd[isimustep][6]=AKIDetGetCounterCycle("DD7",NULL)/4.0;
                svd[isimustep][6]=1.4667*AKIDetGetSpeedCycle("DD7",NULL);
                nvd[isimustep][7]=AKIDetGetCounterCycle("DD8",NULL)/4.0;
                svd[isimustep][7]=1.4667*AKIDetGetSpeedCycle("DD8",NULL);
                }
        }
  }

// &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&& Block One   Initial Queue Estimation Begins &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
        if (isimustep==27)
    {
                isimustep=0;
                iopt=1;
                idprolling=idprolling+1; // one DP horizon is over, and a new horizon starts

                        for (int j=0;j<=7;j++)
                        {
                        asvd[j]=0;
                                anvd[j]=0;
                                isvd=0;
                                w[j]=1.0;

                        for (int i=0;i<=27;i++)
                        {
                                        if (svd[i][j]>0.01)
                                        {
                                                asvd[j]=asvd[j]+svd[i][j];
                                                isvd=isvd+1;
                                        }
                                        anvd[j]=anvd[j]+nvd[i][j];
                                }
                        asvd[j]=asvd[j]/isvd;
                                if (asvd[j]<0.1) asvd[j]=MeanSpeed;
                                ahdw[j]=1.5;
                                anvd[j]=anvd[j]/28;
                        }
                        phase1=0;
                        phase2=0;
                        phase4=0;
                        phase12=0;
                        phase5=0;
                        phase6=0;
                        phase8=0;
                        phase56=0;

                        if (ECIGetCurrentPhase(1)==1) phase2=1;
                        if (ECIGetCurrentPhase(1)==3) phase4=1;
                        if (ECIGetCurrentPhase(1)==5) phase1=1;
                        if (phase1==1||phase2==1) phase12=1;

                        if (ECIGetCurrentPhase(2)==1) phase6=1;
                        if (ECIGetCurrentPhase(2)==3) phase8=1;
                        if (ECIGetCurrentPhase(2)==5) phase5=1;
                        if (phase5==1||phase6==1) phase56=1;

sprintf(astring,"\t\t  Phas2= percentd Phase4= %d Phase1=%d && Phase6=%d Phase8= %d Phase5=%d", phase2, phase4, phase1, phase6, phase8, phase5);
AKIPrintString(astring);

if ((phase2==1)&&(phase6==1)) pre_phase=26;
if ((phase2==1)&&(phase5==1)) pre_phase=25;
if ((phase1==1)&&(phase6==1)) pre_phase=16;
if ((phase1==1)&&(phase5==1)) pre_phase=15;
if ((phase4==1)&&(phase8==1)) pre_phase=48;

                        nvsection=AKIVehStateGetNbVehiclesSection(4);
                        qinitial[0]=0;
                        qinitial[2]=0;

                        if (nvsection>0)
                        {
                                for (int invs=0; invs<nvsection; invs++)
                                {
                                InfVeh_Section=AKIVehStateGetVehicleInfSection(4,invs);
                                InfVeh_Split=AKIVehGetVehicleStaticInfSection(4,invs);
                                if (InfVeh_Section.CurrentSpeed<3.3)
                                        {
                                                if (InfVeh_Split.centroidDest==4) qinitial[0]=qinitial[0]+1;
                                                if (InfVeh_Split.centroidDest==1) qinitial[2]=qinitial[2]+1;
                                        }
                                }
                                qinitial[0]=qinitial[0]/2.0-phase1*(2/ahdw[0]);
                                qinitial[0]=qinitial[0]/(800/23);
                                if (qinitial[0]>0.1)
                                        {       w[0]=1.0;
                                        }
                                if (qinitial[0]>0.2)
                                        {       w[0]=2.0;
                                        }
```

```
                        if (qinitial[0]>0.3)
                                {       w[0]=3.0;
                                }
                        if (qinitial[0]>0.4)
                                {       w[0]=4.0;
                                }
                        if (qinitial[0]>0.5)
                                {       w[0]=5.0;
                                }
                        if (qinitial[0]>0.6)
                                {       w[0]=6.0;
                                }
                        if (qinitial[0]>0.7)
                                {       w[0]=7.0;
                                }
                        if (qinitial[0]>0.8)
                                {       w[0]=8.0;
                                }
                        if (qinitial[0]>0.9)
                                {       w[0]=9.0;
                                }
                        if (qinitial[0]<0) qinitial[0]=0;

                        qinitial[2]=qinitial[2]/4.0-phase12*(2/ahdw[1]);
                        qinitial[2]=qinitial[2]/(800/23);

                        if (qinitial[2]>0.1)
                                {       w[2]=1.0;
                                }
                        if (qinitial[2]>0.2)
                                {       w[2]=2.0;
                                }
                        if (qinitial[2]>0.3)
                                {       w[2]=3.0;
                                }
                        if (qinitial[2]>0.4)
                                {       w[2]=4.0;
                                }
                        if (qinitial[2]>0.5)
                                {       w[2]=5.0;
                                }
                        if (qinitial[2]>0.6)
                                {       w[2]=6.0;
                                }
                        if (qinitial[2]>0.7)
                                {       w[2]=7.0;
                                }
                        if (qinitial[2]>0.8)
                                {       w[2]=8.0;
                                }
                        if (qinitial[2]>0.9)
                                {       w[2]=9.0;
                                }
                        if (qinitial[2]<0) qinitial[2]=0;
                }
// Lane Group 2 +DP phase 2
                nvsection=AKIVehStateGetNbVehiclesSection(1);
                qinitial[1]=0;
                if (nvsection>0)
                {
                        for (int invs=0; invs<nvsection; invs++)
                        {
                        InfVeh_Section=AKIVehStateGetVehicleInfSection(1,invs);
                        if (InfVeh_Section.CurrentSpeed<3.3) qinitial[1]=qinitial[1]+1;

                        }
                        qinitial[1]=qinitial[1]/6.0-phase2*(2/ahdw[1]);
                        qinitial[1]=qinitial[1]/(800/23);

                        if (qinitial[1]>0.1)
                                {       w[1]=1.0;
                                }
                        if (qinitial[1]>0.2)
                                {       w[1]=2.0;
                                }
                        if (qinitial[1]>0.3)
                                {       w[1]=3.0;
                                }
                        if (qinitial[1]>0.4)
                                {       w[1]=4.0;
                                }
                        if (qinitial[1]>0.5)
                                {       w[1]=5.0;
                                }
                        if (qinitial[1]>0.6)
                                {       w[1]=6.0;
                                }
                        if (qinitial[1]>0.7)
                                {       w[1]=7.0;
                                }
                        if (qinitial[1]>0.8)
                                {       w[1]=8.0;
                                }
```

```
                        if (qinitial[1]>0.9)
                                {               w[1]=9.0;
                                }

                        if (qinitial[1]<0) qinitial[1]=0;
                }

// Lane Group 4 +DP Phase 4
                nvsection=AKIVehStateGetNbVehiclesSection(2);                   // Section ID=4  for DD4-DS4
                qinitial[3]=0;
                if (nvsection>0)
                {
                        for (int invs=0; invs<nvsection; invs++)
                        {
                        InfVeh_Section=AKIVehStateGetVehicleInfSection(2,invs);
                        if (InfVeh_Section.CurrentSpeed<3.3) qinitial[3]=qinitial[3]+1;
                        }
                        qinitial[3]=qinitial[3]/4.0-phase4*(2/ahdw[3]);
                        qinitial[3]=qinitial[3]/(800/23);

                        if (qinitial[3]>0.1)
                                {               w[3]=4.0;
                                }
                        if (qinitial[3]>0.2)
                                {               w[3]=5.0;
                                }
                        if (qinitial[3]>0.3)
                                {               w[3]=6.0;
                                }
                        if (qinitial[3]>0.4)
                                {               w[3]=7.0;
                                }
                        if (qinitial[3]>0.5)
                                {               w[3]=8.0;
                                }
                        if (qinitial[3]>0.6)
                                {               w[3]=9.0;
                                }
                        if (qinitial[3]>0.7)
                                {               w[3]=10.0;
                                }
                        if (qinitial[3]>0.8)
                                {               w[3]=11.0;
                                }
                        if (qinitial[3]>0.9)
                                {               w[3]=12.0;
                                }
                        if (qinitial[3]<0) qinitial[3]=0;
                }
                nvsection=AKIVehStateGetNbVehiclesSection(5);
                qinitial[4]=0;
                qinitial[6]=0;
                if (nvsection>0)
                {
                        for (int invs=0; invs<nvsection; invs++)
                        {
                        InfVeh_Section=AKIVehStateGetVehicleInfSection(5,invs);
                        InfVeh_Split=AKIVehGetVehicleStaticInfSection(5,invs);
                        if (InfVeh_Section.CurrentSpeed<3.3)
                                {
                                if (InfVeh_Split.centroidDest==8) qinitial[4]=qinitial[4]+1;
                                if (InfVeh_Split.centroidDest==7) qinitial[6]=qinitial[6]+1;
                                }
                        }
                        qinitial[4]=qinitial[4]/2.0-phase5*(2/ahdw[4]);
                        qinitial[4]=qinitial[4]/(800/23);

                        if (qinitial[4]>0.1)
                                {               w[4]=1.0;
                                }
                        if (qinitial[4]>0.2)
                                {               w[4]=2.0;
                                }
                        if (qinitial[4]>0.3)
                                {               w[4]=3.0;
                                }
                        if (qinitial[4]>0.4)
                                {               w[4]=4.0;
                                }
                        if (qinitial[4]>0.5)
                                {               w[4]=5.0;
                                }
                        if (qinitial[4]>0.6)
                                {               w[4]=6.0;
                                }
                        if (qinitial[4]>0.7)
                                {               w[4]=7.0;
                                }
                        if (qinitial[4]>0.8)
                                {               w[4]=8.0;
                                }
                        if (qinitial[4]>0.9)
                                {               w[4]=9.0;
```

```
                                        }
                                if (qinitial[4]<0) qinitial[4]=0;

                                qinitial[6]=qinitial[6]/4.0-phase56*(2/ahdw[6]);
                                qinitial[6]=qinitial[6]/(800/23);

                                if (qinitial[6]>0.1)
                                        {               w[6]=1.0;
                                        }
                                if (qinitial[6]>0.2)
                                        {               w[6]=2.0;
                                        }
                                if (qinitial[6]>0.3)
                                        {               w[6]=3.0;
                                        }
                                if (qinitial[6]>0.4)
                                        {               w[6]=4.0;
                                        }
                                if (qinitial[6]>0.5)
                                        {               w[6]=5.0;
                                        }
                                if (qinitial[6]>0.6)
                                        {               w[6]=6.0;
                                        }
                                if (qinitial[6]>0.7)
                                        {               w[6]=7.0;
                                        }
                                if (qinitial[6]>0.8)
                                        {               w[6]=8.0;
                                        }
                                if (qinitial[6]>0.9)
                                        {               w[6]=9.0;
                                        }
                                if (qinitial[6]<0) qinitial[6]=0;
                        }
// Lane Group 6 +DP Phase 6
                        nvsection=AKIVehStateGetNbVehiclesSection(9);                   // Section ID=6  for DD6-DS6
                        qinitial[5]=0;
                        if (nvsection>0)
                        {
                                for (int invs=0; invs<nvsection; invs++)
                                {
                                InfVeh_Section=AKIVehStateGetVehicleInfSection(9,invs);
                                if (InfVeh_Section.CurrentSpeed<3.3) qinitial[5]=qinitial[5]+1;
                                }
                                qinitial[5]=qinitial[5]/6.0-phase6*(2/ahdw[5]);
                                qinitial[5]=qinitial[5]/(800/23);
                                if (qinitial[5]>0.1)
                                        {               w[5]=1.0;
                                        }
                                if (qinitial[5]>0.2)
                                        {               w[5]=2.0;
                                        }
                                if (qinitial[5]>0.3)
                                        {               w[5]=3.0;
                                        }
                                if (qinitial[5]>0.4)
                                        {               w[5]=4.0;
                                        }
                                if (qinitial[5]>0.5)
                                        {               w[5]=5.0;
                                        }
                                if (qinitial[5]>0.6)
                                        {               w[5]=6.0;
                                        }
                                if (qinitial[5]>0.7)
                                        {               w[5]=7.0;
                                        }
                                if (qinitial[5]>0.8)
                                        {               w[5]=8.0;
                                        }
                                if (qinitial[5]>0.9)
                                        {               w[5]=9.0;
                                        }
                                if (qinitial[5]<0) qinitial[5]=0;
                        }
// Lane Group 8 +DP Phase 8
                        nvsection=AKIVehStateGetNbVehiclesSection(11);                  // Section ID=8  for DD8-DS8
                        qinitial[7]=0;
                        if (nvsection>0)
                        {
                                for (int invs=0; invs<nvsection; invs++)
                                {
                                InfVeh_Section=AKIVehStateGetVehicleInfSection(11,invs);
                                if (InfVeh_Section.CurrentSpeed<3.3) qinitial[7]=qinitial[7]+1;
                                }
                                qinitial[7]=qinitial[7]/4.0-phase8*(2/ahdw[7]);
                                qinitial[7]=qinitial[7]/(800/23);

                                if (qinitial[7]>0.1)
                                        {               w[7]=4.0;
                                        }
                                if (qinitial[7]>0.2)
```

```
                                                {               w[7]=5.0;
                                                }
                                        if (qinitial[7]>0.3)
                                                {               w[7]=6.0;
                                                }
                                        if (qinitial[7]>0.4)
                                                {               w[7]=7.0;
                                                }
                                        if (qinitial[7]>0.5)
                                                {               w[7]=8.0;
                                                }
                                        if (qinitial[7]>0.6)
                                                {               w[7]=9.0;
                                                }
                                        if (qinitial[7]>0.7)
                                                {               w[7]=10.0;
                                                }
                                        if (qinitial[7]>0.8)
                                                {               w[7]=11.0;
                                                }
                                        if (qinitial[7]>0.9)
                                                {               w[7]=12.0;
                                                }
                                        if (qinitial[7]<0) qinitial[7]=0;
                        }
}
// ********Block Two - Vehicles Projection and DP Value Iteration - Forward Stage *************
if (iopt==1)
{ // === DP Optimization Starts ===
iopt=0;
if (pre_phase==26)
{
sprintf(astring,"\t\t  Decision Network 1 - Phase 26");
AKIPrintString(astring);
                for (int i=0;i<=4;i++)
                for (int j=0;j<=4;j++)
  {
  f[j][i]=0;
  imin[j][i]=0;
   }
for (int ii=0;ii<=3;ii++)
                {
                if (ii==0)
                {
                        jnode=0;
                        for (int j=0;j<=3;j++)
                        {
                                ff0=f[j][ii];
                                ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);

                                f[j][ii+1]=ff1;
                                imin[j][ii+1]=jnode;
                        for (int k=0;k<=7;k++)
                                {
                                        qqinitial[k][j]=qfinal[k];
                                }
                        }
                }

                if (ii==1)
                {
                        for (int jj=0;jj<=3;jj++)
                        {
                        for (int k=0;k<=7;k++)
                        {
                        qinitial[k]=qqinitial[k][jj];
                        }
                        if (jj==0)  /
                        {
                                for (int j=0;j<=3;j++)
                                {
                                ff0=f[jj][ii];
                                ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        f[j][ii+1]=ff1;
                        imin[j][ii+1]=jj;

                                for (int k=0;k<=7;k++)
                                                {
                                                qqtemp[k][j]=qfinal[k];
                                                }
                        }
                        }


  if (jj==1)
                {

                  ff0=f[jj][ii];

                  j1=1;
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
    if (ff1<f[1][ii+1])
                        {
```

```
                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jj;
    for (int k=0;k<=7;k++)
                                {
                                        qqtemp[k][j1]=qfinal[k];
                                }
                                }

                                ff0=f[jj][ii];
                                j1=3;

                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
   if (ff1<f[3][ii+1])
   {
                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jj;
    for (int k=0;k<=7;k++)
                                {
                                        qqtemp[k][j1]=qfinal[k];
                                }
   }
}

if (jj==2)
{
                                j1=2;
                                ff0=f[jj][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[2][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jj;
                for (int k=0;k<=7;k++)
                                        {
                                                qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=3;
                  ff0=f[jj][ii];
 ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
   if (ff1<f[3][ii+1])
        {
                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jj;
                for (int k=0;k<=7;k++)
                                        {
                                                qqtemp[k][j1]=qfinal[k];
                                        }
    }
}

if (jj==3)
{
                                j1=3;
                                ff0=f[jj][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[3][ii+1])
                                {
                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jj;
                for (int k=0;k<=7;k++)
                                        {
                                                qqtemp[k][j1]=qfinal[k];
                                        }

                }


                                j1=4;
                                ff0=f[jj][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j1][ii+1]=ff1;
                for (int k=0;k<=7;k++)
                                {
                                        qqtemp[k][j1]=qfinal[k];
                                }
   }

                }


                for (int j=0;j<=4;j++)

                {
                                for (int k=0;k<=7;k++)
                                {
                                qqinitial[k][j]=qqtemp[k][j];
                                }
                }


}
```

```
if (ii>1)
{
		for (int jj=0;jj<=4;jj++)
		{

		for (int k=0;k<=7;k++)
		{
				qinitial[k]=qqinitial[k][jj];
		}

  if (jj==0)
  {
		  for (int j=0;j<=3;j++)
				{
					ff0=f[jj][ii];
					ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
					f[j][ii+1]=ff1;
					imin[j][ii+1]=jj;

		for (int k=0;k<=7;k++)
					{
						qqtemp[k][j]=qfinal[k];
					}
    }
  }

  if (jj==1)
		{
				j1=1;
				ff0=f[jj][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				if (ff1<f[1][ii+1])
				{
				f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jj;
      for (int k=0;k<=7;k++)
					{
						qqtemp[k][j1]=qfinal[k];
					}

				}
				j1=3;
		ff0=f[jj][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				if (ff1<f[3][ii+1])
				{
				f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jj;
		for (int k=0;k<=7;k++)
					{
						qqtemp[k][j1]=qfinal[k];
					}
    }
  }

 if (jj==2)
  {
				j1=2;
		ff0=f[jj][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				if (ff1<f[2][ii+1])
				{
		f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jj;
		for (int k=0;k<=7;k++)
					{
						qqtemp[k][j1]=qfinal[k];
					}
				}

				j1=3;
		ff0=f[jj][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				if (ff1<f[3][ii+1])

				{
		f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jj;
		for (int k=0;k<=7;k++)
					{
						qqtemp[k][j1]=qfinal[k];
					}
    }
  }

  if (jj==3)
  {
				j1=3;
		ff0=f[jj][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				if (ff1<f[3][ii+1])
				{
```

```
                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jj;
                for (int k=0;k<=7;k++)
                                        {
                                                qqtemp[k][j1]=qfinal[k];
                                        }
                        }

                        j1=4;
                ff0=f[jj][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jj;
                for (int k=0;k<=7;k++)
                                        {
                                                qqtemp[k][j1]=qfinal[k];
                                        }

  }

  if (jj==4)
  {
                        j1=0;
                ff0=f[jj][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        if (ff1<f[0][ii+1])
                        {
                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jj;
                for (int k=0;k<=7;k++)
                                        {
                                                qqtemp[k][j1]=qfinal[k];
                                        }
                        }

                        j1=4;
                ff0=f[jj][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        if (ff1<f[4][ii+1])
                        f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jj;
                for (int k=0;k<=7;k++)
                                        {
                                                qqtemp[k][j1]=qfinal[k];
                                        }
    }
   }
  }


  for (int j=0;j<=4;j++)

                {
                        for (int k=0;k<=7;k++)
                        {
                        qqinitial[k][j]=qqtemp[k][j];
                        }
                }
 }

if (pre_phase==25)
{
sprintf(astring,"\t\t  Decision Network 2 - Phase 25");
AKIPrintString(astring);

                for (int i=0;i<=4;i++)
                for (int j=0;j<=4;j++)
  {
  f[j][i]=0;
  imin[j][i]=0;
  }

                for (int ii=0;ii<=3;ii++)
                {

                if (ii==0)
                {
                        jnode=1;
                                        ff0=f[jnode][ii];
                                        j1=1;
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                        {
                                        qqinitial[k][j1]=qfinal[k];
                                        }

                                        j1=3;
                                        ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
```

```
                        for (int k1=0;k1<=7;k1++)
                        {
                        qqinitial[k1][j1]=qfinal[k1];
                        }
}

if (ii==1)
{

        jnode=1;
                        for (int k=0;k<=7;k++)
                        {
                        qinitial[k]=qqinitial[k][jnode];
                        }

                        ff0=f[jnode][ii];
                        j1=1;
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
        f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
                        for (int k1=0;k1<=7;k1++)
                        {
                        qqtemp[k1][j1]=qfinal[k1];
                        }

                        ff0=f[jnode][ii];
                        j1=3;
        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
        f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
                        for (int k2=0;k2<=7;k2++)
                        {
                        qqtemp[k2][j1]=qfinal[k2];
                        }

        jnode=3;
                        for (int k3=0;k3<=7;k3++)
                        {
                        qinitial[k3]=qqinitial[k3][jnode];
                        }

                        j1=3;
                        ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        if (ff1<f[j1][ii+1])
                        {
                        f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
                        for (int k=0;k<=7;k++)
                                {
                                qqtemp[k][j1]=qfinal[k];
                                }
                        }

                        j1=4;           /
                        ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
        for (int k8=0;k8<=7;k8++)
                        {
                        qqtemp[k8][j1]=qfinal[k8];
                        }


                        for (int j=0;j<=4;j++)
                        {
                                for (int k=0;k<=7;k++)
                                {
                                qqinitial[k][j]=qqtemp[k][j];
                                }
                        }

}


if (ii==2)
{ /
        jnode=1;
                        for (int k=0;k<=7;k++)
                        {
                        qinitial[k]=qqinitial[k][jnode];
                        }

                        ff0=f[jnode][ii];
                        j1=1;
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
        f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
                        for (int k1=0;k1<=7;k1++)
                        {
                        qqtemp[k1][j1]=qfinal[k1];
                        }
```

```
                    ff0=f[jnode][ii];
                    j1=3;
            ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
            f[j1][ii+1]=ff1;
                    imin[j1][ii+1]=jnode;
                    for (int k3=0;k3<=7;k3++)
                    {
                    qqtemp[k3][j1]=qfinal[k3];
                    }

            jnode=3;
                    for (int k4=0;k4<=7;k4++)
                    {
                    qinitial[k4]=qqinitial[k4][jnode];
                    }

                    j1=3;
                    ff0=f[jnode][ii];
                    ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                    if (ff1<f[j1][ii+1])
                    {
                    f[j1][ii+1]=ff1;
                            imin[j1][ii+1]=jnode;
                    for (int k=0;k<=7;k++)
                            {
                            qqtemp[k][j1]=qfinal[k];
                            }
                    }

                    j1=4;
                    ff0=f[jnode][ii];
                    ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                    f[j1][ii+1]=ff1;
                    imin[j1][ii+1]=jnode;
            for (int k9=0;k9<=7;k9++)
                    {
                    qqtemp[k9][j1]=qfinal[k9];
                    }

            jnode=4;
                    for (int k5=0;k5<=7;k5++)
                    {
                    qinitial[k5]=qqinitial[k5][jnode];
                    }

                    j1=0;
            ff0=f[jnode][ii];
                    ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
            f[j1][ii+1]=ff1;
                    imin[j1][ii+1]=jnode;
            for (int k6=0;k6<=7;k6++)
                    {
                            qqtemp[k6][j1]=qfinal[k6];
                    }


                    j1=4;
            ff0=f[jnode][ii];
                    ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                    if (ff1<f[j1][ii+1])
                    {
                            f[j1][ii+1]=ff1;
                            imin[j1][ii+1]=jnode;
                    for (int k=0;k<=7;k++)
                            {
                            qqtemp[k][j1]=qfinal[k];
                            }
                    }

                    for (int j=0;j<=4;j++)
                    {
                            for (int k=0;k<=7;k++)
                            {
                            qqinitial[k][j]=qqtemp[k][j];
                            }
                    }
}


if (ii==3)
{

            jnode=0;
                    for (int k=0;k<=7;k++)
                    {
                    qinitial[k]=qqinitial[k][jnode];
                    }
                    for (int j=0;j<=3;j++)
                    {
                            ff0=f[jnode][ii];
                            ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                            f[j][ii+1]=ff1;
```

```
                                        imin[j][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j]=qfinal[k];
                                        }
                                }

                        jnode=1;
                                j1=1;
                                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=3;
                        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                        f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
        for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                        jnode=3;

                                j1=3;
                        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=4;
                        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
                        for (int k1=0;k1<=7;k1++)
                                        {
                                        qqtemp[k1][j1]=qfinal[k1];
                                        }


                        jnode=4;

                                j1=0;
                        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=4;
                        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                        f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k1=0;k1<=7;k1++)
                                        {
                                        qqtemp[k1][j1]=qfinal[k1];
                                        }
                                }

                        }
if (pre_phase==16)
{
sprintf(astring,"\t\t  Decision Network 3 - Phase 16");
AKIPrintString(astring);
```

```
		for (int i=0;i<=4;i++)
		for (int j=0;j<=4;j++)
{
f[j][i]=0;
imin[j][i]=0;
}

		for (int ii=0;ii<=3;ii++)
		{
		if (ii==0)
		{
				jnode=2;
					ff0=f[jnode][ii];
					j1=2;
					ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jnode;
					for (int k=0;k<=7;k++)
					{
					qqinitial[k][j1]=qfinal[k];
					}

					j1=3;
					ff0=f[jnode][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
		f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jnode;
					for (int k1=0;k1<=7;k1++)
					{
					qqinitial[k1][j1]=qfinal[k1];
					}
		}

		if (ii==1)
		{
				jnode=2;
					for (int k=0;k<=7;k++)
					{
					qinitial[k]=qqinitial[k][jnode];
					}

					ff0=f[jnode][ii];
					j1=2;
					ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jnode;
					for (int k1=0;k1<=7;k1++)
					{
					qqtemp[k1][j1]=qfinal[k1];
					}

					ff0=f[jnode][ii];
					j1=3;
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jnode;
					for (int k2=0;k2<=7;k2++)
					{
					qqtemp[k2][j1]=qfinal[k2];
					}

				jnode=3;
					for (int k3=0;k3<=7;k3++)
					{
					qinitial[k3]=qqinitial[k3][jnode];
					}

					j1=3;
					ff0=f[jnode][ii];
					ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
					if (ff1<f[j1][ii+1])
					{
					f[j1][ii+1]=ff1;
						imin[j1][ii+1]=jnode;
					for (int k=0;k<=7;k++)
						{
						qqtemp[k][j1]=qfinal[k];
						}
					}

					j1=4;
					ff0=f[jnode][ii];
					ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
					f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jnode;
				for (int k8=0;k8<=7;k8++)
					{
					qqtemp[k8][j1]=qfinal[k8];
					}


					for (int j=0;j<=4;j++)
```

```
                {
                        for (int k=0;k<=7;k++)
                        {
                        qqinitial[k][j]=qqtemp[k][j];
                        }
                }

}

if (ii==2)
{
        jnode=2;
                for (int k=0;k<=7;k++)
                {
                qinitial[k]=qqinitial[k][jnode];
                }

                ff0=f[jnode][ii];
                j1=2;
                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
        f[j1][ii+1]=ff1;
                imin[j1][ii+1]=jnode;
                for (int k1=0;k1<=7;k1++)
                {
                qqtemp[k1][j1]=qfinal[k1];
                }

                ff0=f[jnode][ii];
                j1=3;
        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
        f[j1][ii+1]=ff1;
                imin[j1][ii+1]=jnode;
                for (int k3=0;k3<=7;k3++)
                {
                qqtemp[k3][j1]=qfinal[k3];
                }

        jnode=3;
                for (int k4=0;k4<=7;k4++)
                {
                qinitial[k4]=qqinitial[k4][jnode];
                }

                j1=3;
                ff0=f[jnode][ii];
                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                if (ff1<f[j1][ii+1])
                {
                f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
                for (int k=0;k<=7;k++)
                        {
                        qqtemp[k][j1]=qfinal[k];
                        }
                }

                j1=4;
                ff0=f[jnode][ii];
                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                f[j1][ii+1]=ff1;
                imin[j1][ii+1]=jnode;
        for (int k9=0;k9<=7;k9++)
                {
                qqtemp[k9][j1]=qfinal[k9];
                }

        jnode=4;
                for (int k5=0;k5<=7;k5++)
                {
                qinitial[k5]=qqinitial[k5][jnode];
                }

                j1=0;
        ff0=f[jnode][ii];
                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
        f[j1][ii+1]=ff1;
                imin[j1][ii+1]=jnode;
        for (int k6=0;k6<=7;k6++)
                {
                        qqtemp[k6][j1]=qfinal[k6];
                }


                j1=4;
        ff0=f[jnode][ii];
                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                if (ff1<f[j1][ii+1])
                {
                        f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
                for (int k=0;k<=7;k++)
                        {
```

```
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                for (int j=0;j<=4;j++)
                                {
                                        for (int k=0;k<=7;k++)
                                        {
                                        qqinitial[k][j]=qqtemp[k][j];
                                        }
                                }
}


if (ii==3)
{

        jnode=0;
                                for (int k=0;k<=7;k++)
                                {
                                qinitial[k]=qqinitial[k][jnode];
                                }
                                for (int j=0;j<=3;j++)
                                {
                                        ff0=f[jnode][ii];
                                        ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                        f[j][ii+1]=ff1;
                                        imin[j][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j]=qfinal[k];
                                        }
                                }

        jnode=2;
                                j1=2;
                                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=3;
        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
        f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

        jnode=3;

                                j1=3;
        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=4;
        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
        for (int k1=0;k1<=7;k1++)
                                        {
                                        qqtemp[k1][j1]=qfinal[k1];
                                        }


        jnode=4;

                                j1=0;
        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
```

```
                                        {
                                        f[j1][ii+1]=ff1;
                                                imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                                {
                                                qqtemp[k][j1]=qfinal[k];
                                                }
                                        }

                                        j1=4;
                                ff0=f[jnode][ii];
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                        if (ff1<f[j1][ii+1])
                                        {
                                                f[j1][ii+1]=ff1;
                                                imin[j1][ii+1]=jnode;
                                        for (int k1=0;k1<=7;k1++)
                                                {
                                                qqtemp[k1][j1]=qfinal[k1];
                                                }
                                        }

                                }

if (pre_phase==15)

{
sprintf(astring,"\t\t  Decision Network 4 - Phase 15");
AKIPrintString(astring);

                for (int i=0;i<=4;i++)
                for (int j=0;j<=4;j++)
  {
  f[j][i]=0;
  imin[j][i]=0;
  }

                for (int ii=0;ii<=3;ii++)
                {

                if (ii==0)
                {
                                jnode=3;
                                        ff0=f[jnode][ii];
                                        j1=3;
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                        {
                                        qqinitial[k][j1]=qfinal[k];
                                        }

                                        j1=4;
                                        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k1=0;k1<=7;k1++)
                                        {
                                        qqinitial[k1][j1]=qfinal[k1];
                                        }
                }

                if (ii==1)
                {

                                jnode=3;
                                        for (int k=0;k<=7;k++)
                                        {
                                        qinitial[k]=qqinitial[k][jnode];
                                        }

                                        ff0=f[jnode][ii];
                                        j1=3;
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k1=0;k1<=7;k1++)
                                        {
                                        qqtemp[k1][j1]=qfinal[k1];
                                        }

                                        ff0=f[jnode][ii];
                                        j1=4;
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k2=0;k2<=7;k2++)
                                        {
                                        qqtemp[k2][j1]=qfinal[k2];
                                        }
```

```
		jnode=4;
				for (int k3=0;k3<=7;k3++)
				{
				qinitial[k3]=qqinitial[k3][jnode];
				}

				j1=4;
				ff0=f[jnode][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				if (ff1<f[j1][ii+1])
				{
				f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jnode;
				for (int k=0;k<=7;k++)
					{
					qqtemp[k][j1]=qfinal[k];
					}
				}

				j1=0;
				ff0=f[jnode][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				f[j1][ii+1]=ff1;
				imin[j1][ii+1]=jnode;
		for (int k8=0;k8<=7;k8++)
				{
				qqtemp[k8][j1]=qfinal[k8];
				}


				for (int j=0;j<=4;j++)
				{
					for (int k=0;k<=7;k++)
					{
					qqinitial[k][j]=qqtemp[k][j];
					}
				}

}


if (ii==2)
{
		jnode=0;
				for (int k=0;k<=7;k++)
				{
				qinitial[k]=qqinitial[k][jnode];
				}
				for (int j=0;j<=3;j++)
				{
					ff0=f[jnode][ii];
					ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
					f[j][ii+1]=ff1;
					imin[j][ii+1]=jnode;
				for (int k=0;k<=7;k++)
					{
					qqtemp[k][j]=qfinal[k];
					}
				}

		jnode=3;
				for (int k4=0;k4<=7;k4++)
				{
				qinitial[k4]=qqinitial[k4][jnode];
				}

				j1=3;
				ff0=f[jnode][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				if (ff1<f[j1][ii+1])
				{
				f[j1][ii+1]=ff1;
					imin[j1][ii+1]=jnode;
				for (int k=0;k<=7;k++)
					{
					qqtemp[k][j1]=qfinal[k];
					}
				}

				j1=4;
				ff0=f[jnode][ii];
				ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
				f[j1][ii+1]=ff1;
				imin[j1][ii+1]=jnode;
		for (int k9=0;k9<=7;k9++)
				{
				qqtemp[k9][j1]=qfinal[k9];
				}

		jnode=4;
				for (int k5=0;k5<=7;k5++)
				{
				qinitial[k5]=qqinitial[k5][jnode];
```

```
                        }

                        j1=0;
                ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
                for (int k6=0;k6<=7;k6++)
                        {
                                qqtemp[k6][j1]=qfinal[k6];
                        }


                        j1=4;
                ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        if (ff1<f[j1][ii+1])
                        {
                                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
                        for (int k=0;k<=7;k++)
                                {
                                qqtemp[k][j1]=qfinal[k];
                                }
                        }

                        for (int j9=0;j9<=4;j9++)
                        for (int k=0;k<=7;k++)
                                {
                                qqinitial[k][j9]=qqtemp[k][j9];
                                }
                }


if (ii==3)
{

                jnode=0;
                        for (int k=0;k<=7;k++)
                        {
                        qinitial[k]=qqinitial[k][jnode];
                        }
                        for (int j=0;j<=3;j++)
                        {
                                ff0=f[jnode][ii];
                                ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j][ii+1]=ff1;
                                imin[j][ii+1]=jnode;
                        for (int k=0;k<=7;k++)
                                {
                                qqtemp[k][j]=qfinal[k];
                                }
                        }

                jnode=1;
                        j1=1;
                        ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        if (ff1<f[j1][ii+1])
                        {
                        f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                {
                                qqtemp[k][j1]=qfinal[k];
                                }
                        }

                        j1=3;
                ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        if (ff1<f[j1][ii+1])
                        {
                f[j1][ii+1]=ff1;
                        imin[j1][ii+1]=jnode;
for (int k=0;k<=7;k++)
                                {
                                qqtemp[k][j1]=qfinal[k];
                                }
                        }

                jnode=2;
                        j1=2;
                        ff0=f[jnode][ii];
                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                        if (ff1<f[j1][ii+1])
                        {
                        f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                {
                                qqtemp[k][j1]=qfinal[k];
                                }
```

```
                                        }

                                        j1=3;
                                ff0=f[jnode][ii];
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                        if (ff1<f[j1][ii+1])
                                        {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                        for (int k=0;k<=7;k++)
                                                {
                                                qqtemp[k][j1]=qfinal[k];
                                                }
                                        }

                                jnode=3;

                                        j1=3;
                                ff0=f[jnode][ii];
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                        if (ff1<f[j1][ii+1])
                                        {
                                        f[j1][ii+1]=ff1;
                                                imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                                {
                                                qqtemp[k][j1]=qfinal[k];
                                                }
                                        }

                                        j1=4;
                                ff0=f[jnode][ii];
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                        f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k1=0;k1<=7;k1++)
                                                {
                                                qqtemp[k1][j1]=qfinal[k1];
                                                }


                                jnode=4;

                                        j1=0;
                                ff0=f[jnode][ii];
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                        if (ff1<f[j1][ii+1])
                                        {
                                        f[j1][ii+1]=ff1;
                                                imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                                {
                                                qqtemp[k][j1]=qfinal[k];
                                                }
                                        }

                                        j1=4;
                                ff0=f[jnode][ii];
                                        ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                        if (ff1<f[j1][ii+1])
                                        {
                                                f[j1][ii+1]=ff1;
                                                imin[j1][ii+1]=jnode;
                                        for (int k1=0;k1<=7;k1++)
                                                {
                                                qqtemp[k1][j1]=qfinal[k1];
                                                }
                                        }

                                }
                        }
}

if (pre_phase==48)
{

sprintf(astring,"\t\t  Decision Network 5 - Phase 48");
AKIPrintString(astring);

                for (int i=0;i<=4;i++)
                for (int j=0;j<=4;j++)
  {
  f[j][i]=0;
  imin[j][i]=0;
  }

                for (int ii=0;ii<=3;ii++)
                {

                if (ii==0)
                {
                                jnode=4;
                                        ff0=f[jnode][ii];
```

```
                    j1=0;
                    ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
          f[j1][ii+1]=ff1;
                    imin[j1][ii+1]=jnode;
                    for (int k=0;k<=7;k++)
                    {
                    qqinitial[k][j1]=qfinal[k];
                    }

                    j1=4;
                    ff0=f[jnode][ii];
          ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
f[j1][ii+1]=ff1;
                    imin[j1][ii+1]=jnode;
                    for (int k1=0;k1<=7;k1++)
                    {
                    qqinitial[k1][j1]=qfinal[k1];
                    }
}

          if (ii==1)
{
          jnode=0;
                    for (int k=0;k<=7;k++)
                    {
                    qinitial[k]=qqinitial[k][jnode];
                    }
                    for (int j=0;j<=3;j++)
                    {
                              ff0=f[jnode][ii];
                              ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                              f[j][ii+1]=ff1;
                              imin[j][ii+1]=jnode;
                    for (int k=0;k<=7;k++)
                              {
                              qqtemp[k][j]=qfinal[k];
                              }
                    }

          jnode=4;
                    for (int k5=0;k5<=7;k5++)
                    {
                    qinitial[k5]=qqinitial[k5][jnode];
                    }

                    j1=4;
          ff0=f[jnode][ii];
                    ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
          f[j1][ii+1]=ff1;
                    imin[j1][ii+1]=jnode;
          for (int k6=0;k6<=7;k6++)
                    {
                              qqtemp[k6][j1]=qfinal[k6];
                    }


                    j1=0;
          ff0=f[jnode][ii];
                    ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                    if (ff1<f[j1][ii+1])
                    {
                              f[j1][ii+1]=ff1;
                              imin[j1][ii+1]=jnode;
                    for (int k=0;k<=7;k++)
                              {
                              qqtemp[k][j1]=qfinal[k];
                              }
                    }

                    for (int j8=0;j8<=4;j8++) /
                    for (int k=0;k<=7;k++)
                              {
                              qqinitial[k][j8]=qqtemp[k][j8];
                              }
          }

if (ii>1)
{

          jnode=0;
                    for (int k=0;k<=7;k++)
                    {
                    qinitial[k]=qqinitial[k][jnode];
                    }
                    for (int j=0;j<=3;j++)
                    {
                              ff0=f[jnode][ii];
                              ff1=DPprojection(j,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                              f[j][ii+1]=ff1;
                              imin[j][ii+1]=jnode;
                    for (int k=0;k<=7;k++)
                              {
                              qqtemp[k][j]=qfinal[k];
```

```
                                        }
                                }

                jnode=1;
                                j1=1;
                                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=3;
                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                jnode=2;
                                j1=2;
                                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                        for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=3;
                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                jnode=3;

                                j1=3;
                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                        {
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=4;
                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                f[j1][ii+1]=ff1;
                                imin[j1][ii+1]=jnode;
                for (int k1=0;k1<=7;k1++)
                                        {
                                        qqtemp[k1][j1]=qfinal[k1];
                                        }

                jnode=4;

                                j1=0;
                ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k=0;k<=7;k++)
                                        {
```

```
                                        qqtemp[k][j1]=qfinal[k];
                                        }
                                }

                                j1=4;
                        ff0=f[jnode][ii];
                                ff1=DPprojection(j1,ii,qinitial,nvd,ahdw,asvd,ff0,ff1,qfinal);
                                if (ff1<f[j1][ii+1])
                                {
                                        f[j1][ii+1]=ff1;
                                        imin[j1][ii+1]=jnode;
                                for (int k1=0;k1<=7;k1++)
                                        {
                                        qqtemp[k1][j1]=qfinal[k1];
                                        }
                                }

                                for (int j7=0;j7<=4;j7++)
                                for (int k=0;k<=7;k++)
                                {
                                qqinitial[k][j7]=qqtemp[k][j7];
                                }
                        }
                }
}


// **********************Block Three - Optimal DP Signal Declaration - Backward Stage ***

minf=1000000000;
for (int j=0;j<=4;j++)
                {
                        if (f[j][4]<minf)
                        {
                        minf=f[j][4];
                        minphase=j;

                        }
  }

    phase_optimal[3]=minphase;                                                                  // for layer 3 to 4
  phase_optimal[2]=imin[minphase][4];                           // for layer 2 to 3
  phase_optimal[1]=imin[phase_optimal[2]][3];   // for layer 1 to 2
  phase_optimal[0]=imin[phase_optimal[1]][2];   // for layer 0 to 1

                majoritySignal=phase_optimal[0];                // initial value
                if ((phase_optimal[1]==phase_optimal[0])||(phase_optimal[1]==phase_optimal[2]) || (phase_optimal[1]==phase_optimal[3]))
                {
                  majoritySignal=phase_optimal[1];
                }
                else if ((phase_optimal[2]==phase_optimal[0])||(phase_optimal[2]==phase_optimal[3]))
                {
                  majoritySignal=phase_optimal[2];
                }
                else if (phase_optimal[3]==phase_optimal[0])
                {
                  majoritySignal=phase_optimal[3];
                }
sprintf(astring,"\t\t Optimal Signal at Layer 0 to 1= %d", phase_optimal[0]);
AKIPrintString(astring);
sprintf(astring,"\t\t Optimal Signal at Layer 1 to 2= %d", phase_optimal[1]);
AKIPrintString(astring);
sprintf(astring,"\t\t Optimal Signal at Layer 2 to 3= %d", phase_optimal[2]);
AKIPrintString(astring);
sprintf(astring,"\t\t Optimal Signal at Layer 3 to 4= %d", phase_optimal[3]);
AKIPrintString(astring);
sprintf(astring,"\t\t Majority Signal Phase= %d", majoritySignal);
AKIPrintString(astring);
}
// ************************Block Four - Optimal DP Signal Implementation *******
if (idprolling>=1)
{
                if ((time>=300-0.0001)&&(time<=300+0.0001))
                {
                ECIDisableEvents(1);            // disable the fixed control of junction 1
                ECIDisableEvents(2);            // disable the fixed control of junction 2
                }
if (time>=300+ihorizon*7.5-0.0001) // ihorizon for Majority Signal Rolling, idprolling is for Arrival Detection rolling
  {
if (idp==0)

  {
                 idp=1;
                 idp1=0;
                   if (pre_phase==26)
                                        {
                                        rphase1=2;
                                        rphase2=2;
                                        }

                                        if (pre_phase==25)
                                        {
                                        rphase1=2;
```

```
                                        rphase2=6;
                                        }

                                        if (pre_phase==16)
                                        {
                                        rphase1=6;
                                        rphase2=2;
                                        }

                                        if (pre_phase==15)
                                        {
                                        rphase1=6;
                                        rphase2=6;
                                        }

                                        if (pre_phase==48)
                                        {
                                        rphase1=4;
                                        rphase2=4;
                                        }

                igreen=0;
                                if (majoritySignal==0)
                                {
                                cphase1=1;
                                cphase2=1;
                                if (pre_phase==26) igreen=1;
                                pre_phase=26;
                                }

                                if (majoritySignal==1)
                                {
                                cphase1=1;
                                cphase2=5;
                                if (pre_phase==25) igreen=1;
                                pre_phase=25;
                                }
                                if (majoritySignal==2)
                                {
                                cphase1=5;
                                cphase2=1;
                                if (pre_phase==16) igreen=1;
                                pre_phase=16;
                                }

                                if (majoritySignal==3)
                                {
                                cphase1=5;
                                cphase2=5;
                                if (pre_phase==15)igreen=1;
                                pre_phase=15;
                                }

                                if (majoritySignal==4)
                                {
                                cphase1=3;
                                cphase2=3;
                                if (pre_phase==48)igreen=1;
                                pre_phase=48;
                                }
        }

                 if (igreen==1)
                {
                                 if (idp1<15)
                                 {
                                                ECIChangeDirectPhase(1,cphase1,timeSta,time,acicle);
                                ECIChangeDirectPhase(2,cphase2,timeSta,time,acicle);
                                idp1=idp1+1;
                                }
                }

                if (igreen==0)
                {

                                if (idp1<5)
                                {

                                ECIChangeDirectPhase(1,2,timeSta,time,acicle);
                                ECIChangeDirectPhase(2,2,timeSta,time,acicle);
                                idp1=idp1+1;
                                }
                                if ((idp1>=5) && (idp1<15))
                                {

                                ECIChangeDirectPhase(1,cphase1,timeSta,time,acicle);
                                ECIChangeDirectPhase(2,cphase2,timeSta,time,acicle);
                                idp1=idp1+1;
                                }
                }

}
                if (idp1==15)
```

```
   {
                idp=0;
                idp1=0;
                ihorizon=ihorizon+1;
   }
                if ((time>=7200-0.0001)&&(time<=7200+0.0001))

                {
                ECIEnableEvents(1);
                ECIEnableEvents(2);

                }
}
return 0;
}

// ------------------------------USER SUBROUTINE 1 ---------------------------------------------
// *********************************************************************************************
// The subroutine for the vehicle projection, arrival-discharge at the stopline
// DP value calcultaion over a single DP Interval
// *********************************************************************************************

float DPprojection(int j, int i, float qinitial[8], float nvd [28][8], float ahdw[8], float asvd[8],float ff0, float ff1,float qfinal[8])
{
float T, T1, T2;
float qprojected, qdischarged, dqueue, reward;
int Jahead, Jadditional;
float dqlength[8];
float ff2, ff3;
int phase[8];
int sign_dq;

if (j==0)
   {
   phase[0]=0;
   phase[1]=1;
   phase[2]=1;
   phase[3]=0;
   phase[4]=0;
   phase[5]=1;
   phase[6]=1;
   phase[7]=0;
                }

if (j==1)
                {
   phase[0]=0;
   phase[1]=1;
   phase[2]=1;
   phase[3]=0;
   phase[4]=1;
   phase[5]=0;
   phase[6]=1;
   phase[7]=0;
   }

if (j==2)
                {
phase[0]=1;
phase[1]=0;
phase[2]=1;
phase[3]=0;
phase[4]=0;
phase[5]=1;
phase[6]=1;
                phase[7]=0;
                }

if (j==3)
                {
                phase[0]=1;
                phase[1]=0;
                phase[2]=1;
                phase[3]=0;
                phase[4]=1;
                phase[5]=0;
                phase[6]=1;
                phase[7]=0;
                }

if (j==4)
                {// phase 48, i.e, j=4
                phase[0]=0;
                phase[1]=0;
                phase[2]=0;
                phase[3]=1;
                phase[4]=0;
                phase[5]=0;
                phase[6]=0;
                phase[7]=1;
                }
for (int k=0; k<=7; k++)
{
```

```
T1=asvd[k]/Deceleration;
T=DetectorD/asvd[k];

                        if ((qinitial[k]<1)&&(phase[k]==1))          // no queue and signal green
                                                {
                                                T=DetectorD/asvd[k];
                                                }

                        if ((qinitial[k]<1)&&(phase[k]==0)) // no queue and signal red
                                                {
                                                T2=(DetectorD-BrakingD)/asvd[k];          // T2 is traveling time without braking
                                                T=T1+T2;
                                                }

                        if (qinitial[k]>=1)                                                 // initial queue present
                                                {
                                                T2=(DetectorD-(16.4+(qinitial[k]-1)*23)-BrakingD)/asvd[k];
                                                T=T1+T2;
                                                }

                        if (T>=16) T=16;

        Jahead=static_cast<int>((16-(T-i*2.5))/0.5);

        if (Jahead>23) Jahead=23;
        if (Jahead<0) Jahead=0;

qprojected=nvd[Jahead][k]+nvd[Jahead+1][k]+nvd[Jahead+2][k]+nvd[Jahead+3][k]+nvd[Jahead+4][k];

qdischarged=0;

if (phase[k]==1)
                        {
                        qdischarged=DT/ahdw[k];

                        }
Jadditional=0;
        dqueue=qprojected-qdischarged;  // net added queue
        sign_dq=1;
        if (dqueue<0) sign_dq=-1;
        dqueue=dqueue*sign_dq;
        Jadditional=static_cast<int> (dqueue);

                        if (Jahead+sign_dq*Jadditional<=27)
                        {
                                for (int jj=1; jj<=Jadditional; jj++)
                                {
                                qprojected=qprojected+sign_dq*nvd[Jahead+4+sign_dq*jj][k];
                                }
                        }

                        qfinal[k]=qinitial[k]+qprojected-qdischarged;
                        if (qfinal[k]<0) qfinal[k]=0;

                        dqlength[k]=2.5*qinitial[k]+0.5*2.5*(qprojected-qdischarged); // delay avergae per lane
                        if (dqlength[k]<0) dqlength[k]=0;

                        if (qfinal[k]<0)
                        {
                                qfinal[k]=0;

                        }

                        if (qfinal[k]>10)

                        {
                                qfinal[k]=10;

                        }

}
        reward=0;

//&&&&&&&&&&&&&&&&&&&&&&&& Performance Measures To Be Minimized &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
//
//      Given:
//                      1. qinitial[k] - the queue in the beginning of a DP interval
//                      2. qfinal[k] - the queue in the end of a DP interval
//                      3. dqlength[k] - the queue changes in a DP interval
//                      4. DP interval - 2.5 seconds
//
//      Performance Measure Options
//
//      (1) Total Average Queue Length Per Lane (ie, the Back of Queue Distance ), Unit=Queue


        ff1=ff0+w[0]*qfinal[0]+w[1]*qfinal[1]+w[2]*qfinal[2]+w[3]*qfinal[3]+w[4]*qfinal[4]+w[5]*qfinal[5]+w[6]*qfinal[6]+w[7]*qfinal[7];


// &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&Performance Measure Ends &&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&&
        return ff1;
```

VITA

EDUCATION

Ph.D. Transportation Engineering, Pennsylvania State University
M.Eng. Project and Construction Management, University of British Columbia, Canada,
M.Phil. Civil Engineering, City University of Hong Kong, Hong Kong
B.Eng. Civil Engineering, ChengDu University of Science and Technology, China

WORK EXPERIENCE

Graduate Research Assistant, Pennsylvania Transportation Institute, Pennsylvania State University

Carried out NCHRP 3-60 project–Capacity and LOS Analysis of Interchange Ramp Terminals
Conducted my thesis on optimal and adaptive signal control in ITS.
Managed the AUTOSCOPE (Real-time Traffic Data Collection) System /ATLAS laboratory at PTI

Teaching Assistant, Department of Civil and Environmental Engineering, Pennsylvania State University

Taught the laboratory section of a senior-level undergraduate course (CE 423 – Traffic Operations)

Engineer, JuHua Urban Transport Planning and Design Institute, ZheJiang province, China

AWARDS AND HONORS

Alumni Association Dissertation Award, Graduate School, Pennsylvania State University
Thesis Fellowship, Mid-Atlantic Universities Transportation Center (MAUTC)
Carmen E. Turner Fellowship, Women's Transportation Seminar (WTS), Philadelphia
Graduate Scholar Fellowship, College of Engineering, Pennsylvania State University
WEP Research Travel Grant Fund, Women in Engineering Program, Pennsylvania State University
International Student Scholarship, University of British Columbia
Research Scholarship, City University of Hong Kong

PROFESSIONAL SERVICE

Manager of Advanced Traffic Laboratory for Automated Systems (ATLAS), Center for Traffic Operational Analysis, Pennsylvania Transportation Institute, Pennsylvania State University
Webmaster of the Center for Traffic Operational Analysis website: http://cta.pti.psu.edu/index.html
Secretary, Institute of Transportation Engineering Student Chapter, Pennsylvania State University, 01/2002- 08/2003

PROFESSIONAL MEMBERSHIPS

Member, Institute of Transportation Engineering, American Society of Civil Engineering
Friend, Highway Capacity and Quality of Service Committee and Traffic Signal System Committee

OTHER EDUCATIONAL ACTIVITIES

AIMSUN (a traffic simulation software) Training course, TSS and KLD Associates, Inc., Washington DC, Jan. 2003
Poster presentation at the annual PSU Graduate Students Exhibition, and College of Engineering/PTI Showcase