Integration of Predictive Routing Information with Dynamic Traffic Signal Control

by

Richard C. Staats

B.S., United States Military Academy,
West Point, New York (1984)

Submitted to the Department of
Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Operations Research

at the

Massachusetts Institute of Technology

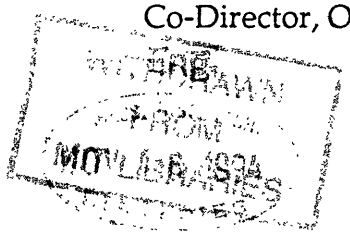
May 1994

Signature of Author_____
Department of Electrical Engineering and Computer Science
May 31, 1994

Certified by_____
John D.C. Little
Institute Professor
Thesis Supervisor

Certified by_____
Arnold I. Barnett
Professor of Operations Research and Management
Thesis Supervisor

Accepted by_____
Thomas L. Magnanti
Co-Director, Operations Research Center



Integration of Predictive Routing Information with Dynamic Traffic Signal Control

by

Richard C. Staats

Submitted to the Department of Electrical Engineering and Computer Science
on May 31, 1994
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Operations Research

Abstract

This thesis explores the integration of predictive routing information available under the Intelligent Vehicle Highway System (IVHS) with dynamic traffic signal control. This exploration was motivated by recent advances in both signal processing and computational technology.

The first portion of the thesis develops the theoretical basis for the Predictive Routing Information Signal Timing INtEgration (PRISTINE) model. PRISTINE explicitly uses the predictive routing information available under IVHS as well as considering the effects of queuing and congestion and compensating for them in setting the traffic signal control plan. The thesis develops a new methodology for using spanning trees to determine the offsets in the network and draws several theoretical results from this premise. The thesis also develops a Queue Effects Model (QEM) that explicitly considers the effects of bulk arrivals on average delay and probability of stopping at an intersection; the Queue Effects Model requires only the average arrival rate and the first two moments of platoon size, all of which can be collected using existing technology. The thesis uses the Queue Effects and spanning tree models as the basis for PRISTINE. Two methods are developed for selecting the splits and cycle time in PRISTINE. One method requires solution of a non-linear program, and the other utilizes a heuristic approach which exploits the structure of the problem to solve the problem in polynomial time. The non-linear program is solved using a gradient descent method utilizing barrier functions, projections and line search techniques to find the constrained optimal solution to the non-continuous, non-linear objective function.

The second portion of the thesis examines the area of evaluating traffic signal control strategies. The thesis develops an optimized third generation control system and a traffic simulation capable of evaluating a generalized traffic signal control strategy. The final section of the thesis compares PRISTINE's performance versus the third generation control system. The two models are compared using actual data collected from in ground sensors in Boston's

Backbay area. PRISTINE offers substantial savings as measured by average delay per vehicle in cases where there were either sudden shifts in traffic flow or dominant directions of flow in the network. Under some conditions, PRISTINE was able to offer 50% savings in average delay and transit time per vehicle over the third generation control model.

Key Words: IVHS, Dynamic Traffic Signal Control, Predictive Routing Information, Spanning Tree, ATMS

Thesis Jointly Supervised by:

John D.C.Little	Institute Professor
Arnold I. Barnett	Professor of Operations Research and Management

Acknowledgments

I would like to begin by thanking the John and Fannie Hertz Foundation which funded my studies at MIT for the past four years. On a more personal note, I would like to thank the Foundation's staff for their help which was always "above and beyond the call of duty." If Dr. Tally had not intervened with the Army in 1990, I would never have been able to come to MIT.

On that note, I would also like to thank the Army for the best thirteen years of my life that began nearly fourteen years ago on the plain at West Point and proceeded through seven assignments on three continents to culminate with the first three years of my doctoral studies at MIT. I would like to sincerely thank my officers and the hundreds of enlisted personnel I had the privilege to command and supervise. "I shall never your likes again."

In particular, I would like to thank Colonel Giordano and Colonel Kolb who were more than mentors to me at the Academy and proceeded to guide and advise through all my assignments in the service. I would like to thank Colonel Bill Hand who was not only my battalion commander at Fort Riley, he was my best friend too. I will always be appreciative to Colonel Bill MacKinlay, my first battalion commander in Korea, who accepted me sight unseen and gave me the chance to command the largest divisional supply and service company in the US Army. I will remember Colonel Bob Bishop not only for his superb guidance as my battalion commander, but also his ability to sift through complex issues and find the important points buried within. I would like to thank General Paul Vanderploog, because without his influence on my career I would never have left "the trenches" and had the opportunity to contribute to the logistics community at a theater level.

I take this opportunity to thank my co-advisors Institute Professor Little and Professor Barnett for taking me in as their Ph. D. student when things seemed darkest in the Summer of 1993. I would like to thank Professor Sussman not only for being a member of my thesis committee but also for being "the hardest working, non-signing member of a thesis committee in the history of the Institute."

I thank my children (Kathy, Beth and Ben) for enduring the last four years where school requirements and physical separations of various types and

durations took me away from them for long periods. You have been an inspiration to me throughout the process.

I take this chance to thank the members of New House III would helped me retain my sanity over the past four years. Between: gas explosions, bicycle accidents, tumors, giving up my career in the service, and personal strife there have been many opportunities for them to come forward and offer their aid and support, and on each occasion they have done so willingly and with glad hearts. In particular I would like to thank the following: Eric Zylstra, Jon Roorda, Phillip Hume, Todd Foley, Joe Wyzorek, Douglas Soo, Costa, Koichi, Ana and Cati, Sumit, Herb, Jorge Medina, Deanna Hiltner, Mark Brandreth, Humair, Lalit, Toby, John Tewksbury and Jeff Dulik. I could not have made it through the last four years without you. I would like to thank those members of Mitgaard who helped me as well. In particular, I would like to thank: dan, Anca, Laura, Alia, Sharon, Jan and Vanessa for their terrific support!

I would like to thank some special friends who have supported me for many years. Dr. Mary Hillstrom has been a firm friend for ten years and the best penpal someone could ask for. Kathy Dunn has been a friend for eighteen years and continues to be one of my closest to this day. Jessica Maybar, a fellow Ph.D. candidate, has been a great source of emotional strength during the past year.

I would like to thank my friends and colleagues at the 94th Army Reserve Command who offered me support and friendship this last year at MIT. I would especially like to thank LTC Kelly, MAJ Barclay and MAJ Towson who have gone well out of their way to help and guide me.

I would like to thank the ORC staff and fellow students who helped and befriended me. In particular, Paulette Mosely and Laura Rose have been tremendous sources of strength and help. Christian Voigtlaender, Rodrigo Rubio, Dave Markowitz, Stefanos, Milt, Barry Kostiner and Alexander Mueller have all gone beyond the pale in offering their friendship and support.

Last, I would like to thank LMI and especially Earl Wingrove, Ron Frola and General Bill Tuttle for keeping a position open for me upon my departure from MIT.

Dedication

I dedicate this thesis to my three wonderful children: Katherine Laurinda (7), Elizabeth Sandra (5) and Richard Benjamin (5) who waited patiently while Daddy played with traffic lights for the last four years. You have been my beacon stars. With all my love and hope!

Contents

List of Tables

List of Figures

Chapter One

Introduction

SECTION 1.1 Overview

Traffic signal control is an area that affects each of us every day, ranging from the morning commute to the economic aspects of a society where virtually 100% of all products are transported on the roadways of the nation at some point between production and consumption. Although traffic signal control and traffic signal research have been active for over thirty years, only recent advances in information technology and micro-processor design and cost effectiveness have made the dream of dynamic control of traffic signals feasible. It is not just the availability of microprocessors that makes dynamic control more plausible but also the new capabilities in data collection, information processing, data base management and the advances in telecommunications which have changed and will continue to change the landscape in which Traffic Signal Control (TSC) finds itself. The Intelligent Vehicle Highway System (IVHS) was conceptualized to integrate these and other factors into a comprehensive plan for alleviating congestion, improving throughput, increasing safety, and numerous other goals for the US traffic systems in the 21st Century (IVHS, 92).

A key portion of IVHS is the Predictive Routing Information (PRI) which will be available to traffic signal control systems. The predictive routing information presents both new opportunities and new challenges. Current traffic signal control strategies are not explicitly designed to utilize this information. There are four basic approaches which have been traditionally used in the area of

TSC; they are: maximizing bandwidth through off-line computation, minimizing average delays or stops using off-line computing, semi-dynamic control which uses off-line computing to derive candidate settings and then fits the setting closest to real time observations, and fully dynamic control.

This thesis accepts the Master Clock Concept (MCC) as the basis for successful traffic signal control. The MCC is the idea that all of the traffic signal control in an area work from the same *master clock*, or stated another way, all of the traffic signals have the same total cycle times. A cycle time is the total amount of red, green and amber time at an intersection in a particular direction.

We make three primary contributions in this thesis. First, we develop a *platform* which allows us to evaluate the benefits of predictive routing information. Second, we explore new strategies for setting traffic signals which can be applied even without the additional information available under IVHS, and third, we examine a particular case, the surge situation, where the use of predictive routing information offers its greatest benefits.

The goal of the thesis is to take the Predictive Routing Information (PRI) available under IVHS and use it in a dynamic TSC system. By dynamic TSC we mean a system that is capable of responding to traffic conditions and PRI information in *real time.* To accomplish this objective, we developed several models. We describe and analyze a new framework for setting off-sets in a TSC scheme by explicitly considering a tree structure in the traffic network under scrutiny. We explore three explicit methods of selecting the offset tree in chapter

four. We explore a means for evaluating TSC plans in terms of common measures of effectiveness, and in particular, the document will develop a Queue Effects Model (QEM) for determining average wait per vehicle at a particular intersection and probability of delay at a particular intersection for a randomly incident vehicle. We demonstrate how to use the average wait in concert with parameters set by the traffic manager to determine the overall cycle time and splits for the network.

These techniques collectively make up the Predictive Routing Information Signal Timing INtEgration (PRISTINE) model, traffic signal timing system which explicitly uses predictive routing information (see Annex C for an example of predictive routing information). The thesis explores two methods to implement PRISTINE. One method is the Split setting Heuristic (SH) technique which exploits elements of the problems structure, and the second method for implementing PRISTINE utilizes a Non-Linear Program (NLP) to solve for the splits and cycle time of the network. Several subtleties in the structure of the problem allow us to solve this NLP extremely quickly.

The ultimate TSC setting scheme should be efficient enough that it can be implemented in real time. There are many good, robust routines which are commercially available which perform TSC operations, but none of the existing packages are designed to utilize the PRI explicitly. We want to evaluate PRISTINE against a system which uses existing traffic data. To accomplish this, we develop a traffic simulation model and an optimized version of a Third

Generation Control (3GC) system. PRISTINE is compared to 3GC using real traffic data collected from Boston's Backbay area.

The thesis is divided into several major sections. The first is an overview of the thesis and a description of how IVHS supports the concept of predictive routing information. The second section is a literature review of relevant articles which sets the historical niche for the current document and examines the traditional approaches to TSC in the literature. The third section introduces the mathematical definitions and derivations which are common to later chapters, and it specifically defines the common measures of effectiveness (MOE's) for traffic signal control. The fourth chapter develops three heuristics which explicitly utilize predictive routing information to set the off-sets and introduces the concept of using a tree structure to define these off-sets. The fifth chapter explores methods for evaluating performance of a TSC and a network in terms of our MOE's. In particular, chapter five develops a Queue Effects Model (QEM) that predicts the effects of congestion on the average wait per vehicle and average stops per vehicle. Chapter six describes several representative, traditional methods for determining cycle time and off-sets, and then goes on to develop PRISTINE, a TSC method which explicitly integrates the predictive routing information available under IVHS and takes into account congestion effects. Chapter seven evaluates PRISTINE using simulation techniques. Chapter seven begins by developing the Lin-Sarkar-Staats Simulation (LS3) and uses LS3 to compare the two techniques for implementing PRISTINE, SH and NLP. The NLP fares better in the majority of the cases tested and is adopted as the standard for PRISTINE. Additionally, the NLP converges to an optimal solution extremely quickly, typically in five to seven hundred iterations. Chapter

eight develops an optimized third generation traffic signal control model, 3GC. PRISTINE is compared to 3GC, and we find that predictive routing information can be very useful in decreasing both the average stops per vehicle and the average delay per vehicle under non-saturated conditions. In the final portion of chapter eight we examine a surge situation; this is where a sudden increase in traffic flow exceeds the capacity of the street segments for the current traffic setting. Since PRISTINE uses predictive routing information, PRISTINE is able to anticipate the increase in the traffic flow and modify the traffic signal plan to accommodate the *surge*.. The concept of predictive routing information is central to IVHS, and we will describe IVHS and how predictive routing information fits into the IVHS vision in the next section.

SECTION 1.2 IVHS, TSC and Predictive Routing Information

The framework for vehicle routing and control for the US in the early portion of the 21st Century will be IVHS. The Strategic Plan for Intelligent Vehicle Highway Systems (IVHS) in the United States (IVHS, 1992) states "surface transportation is at a crossroads." Past solutions will not solve all the complex issues motorists will face in the future. Congestion, accidents, frustration, pollution and plethora of other issues face the traffic system designers and controllers of the future. But, just as there are additional quandaries, so there will be additional sources of help. A significant portion of that help will come in the form of additional information which will be available to the traffic signal control systems of the future. Specifically, through interactions between drivers and data bases and vehicles and central control

units, information on routings will become available. Existing TSC systems are not designed to utilize the additional information under such a configuration. The paragraphs that follow are brief descriptions of IVHS as a whole and its component sub-systems, but they are not designed to be a whole or partial survey of IVHS. These paragraphs focus on the place that predictive traffic information plays in IVHS.

The strategic plan for IVHS in the United States (IVHS, 1992) describes five fundamental components: the Advanced Traffic Management System (ATMS), the Advanced Traveler Information System (ATIS), the Advanced Vehicle Control System (AVCS), Commercial Vehicle Operations (CVO) and Advanced Public Transportation Systems (APTS). The following sections briefly describe these areas.

Intelligent Vehicle Highway System (IVHS)

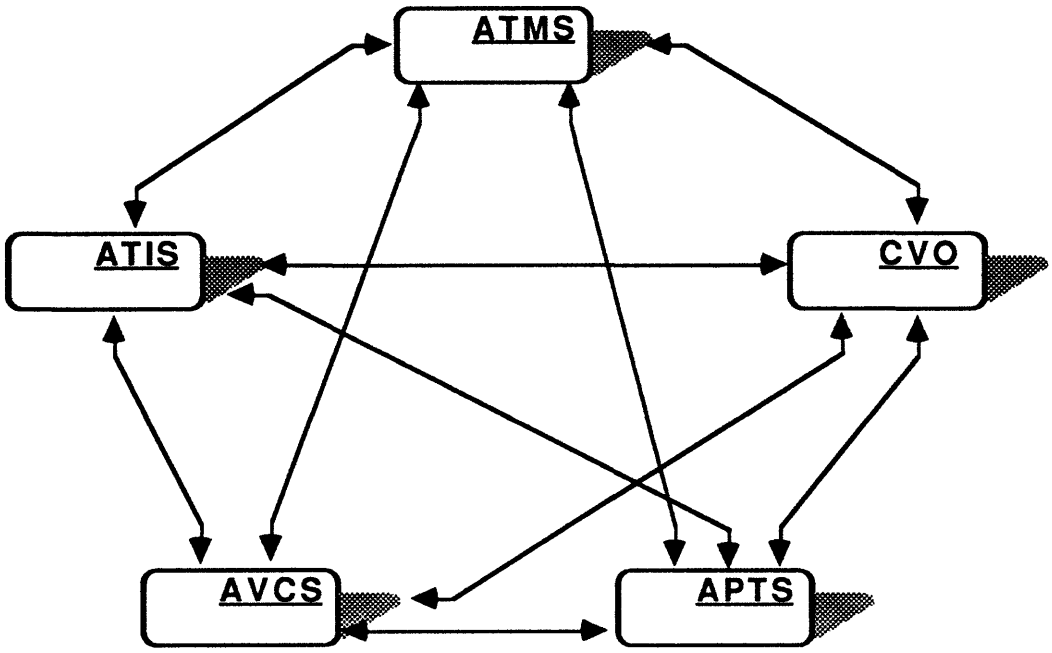


Diagram showing the structure of IVHS; notice how all parts are interconnected.

Figure 1.1

Advanced Traffic Management System (ATMS) is the employment of new, innovative technologies and their integration with traffic management systems to bring increased order and efficiency to the movement of surface vehicles. The strategic plan succinctly states "ATMS represents the 'smart highway' with which the 'smart vehicle' will communicate." (IVHS, 1992) ATMS will rely upon the collection of real-time traffic data, react to that data, change routing information and traffic signal control devices to alleviate congestion and promote safety. All other elements of IVHS will depend on ATMS. Of particular note to this thesis is that ATMS includes *predictive control algorithms* which are products that control traffic based on predicted traffic flows and congestion. Additionally,

ATMS also umbrellas the concept of using vehicles *as probes,* or in other words, the surface transportation systems of the future must account for the additional information gathered through and by the motorists and their vehicles.

The Advanced Traffic Information System (ATIS), quite simply, is the umwelt for the flow of data from all available sources and its journey until the extracted and analyzed final product, traffic and routing information, is passed to the motorist (IVHS, 1992). ATIS includes numerous features some of which are: electronic navigation systems, data communication from and to traffic control centers, route planning and guidance systems, vehicle identification systems, motorist warning systems, detailed maps, directories, etc., and dynamic route guidance. Naturally, ATIS will interface with ATMS. In effect, a motorist planning a route interfaces with a Goliath data base and is able to accurately predict future congestion and anticipated usage of various streets and highway segments. One of the primary goals of the "coordination stage of ATIS" which is projected to be deployed in the range 2000 - 2004 AD is to include "route guidance systems that interact cooperatively with a traffic management center, providing the center with information, as well as receiving information." Thus, the vision of ATIS would include a host of raw data collection systems which would in turn give this data to huge data bases, allowing sophisticated analysis paradigms to accurately and effectively turn this data into invaluable motorist information. The information would be disseminated to motorists in a variety of media, ranging from on-board vehicle information systems to variable signs and radio transmissions.

The strategic plan describes the Advanced Vehicle Control System (AVCS) best when it states "AVCS combines sensors, computers, and control systems in vehicles and in infrastructure to warn and assist drivers or to intervene in the driving task." (IVHS, 1992) AVCS would help warn drivers of potential hazards or invaluable safety and control information by enhancing motorists' perceptions; the on-board display of road signs and speed limits would be an example this implementation. AVCS includes an entire range of options in the area of vehicle control, all the way from driving assistance (e.g. automated steering, acceleration/deceleration) to fully automated control of the vehicle. AVCS seems the most "long term" of all the aspects of IVHS, but in fact, if has been on-going for over 30 years. Consider for a moment all for the enhancements already available in today's vehicles such as power steering, cruise control, anti-lock brakes, etc. Ultimately, perhaps as soon as twenty years from the present, AVCS would comprise of on-board warning systems for intersection hazards and completely automated vehicle operations on specific sections specially designated and instrumented roadways; candidates would include arterial expresses to major metropolitan areas.

Commercial Vehicle Operations (CVO) really comprises a variety of techniques described under ATMS, ATIS and AVCS but as specifically applied to commercial vehicles. Commercial vehicles have a number of unique issues including: additional restrictions on route choice based on size or weight, requirements for tariffs or inspections, safety requirements and subsequent route restrictions, etc. Eventually CVO will be embodied in electronic tax and permit systems and automated vehicle and driver condition monitoring and reporting.

The Advanced Public Transportation System (APTS) apply to all manner of public transportation modes (except for inter-city buses which are covered by CVO). It is noteworthy that this component of IVHS also includes ride-share schemes and the employment, monitoring and use of High Occupancy Vehicle lanes. One might not normally consider these to be part of the mass transportation system. APTS has four primary goals: decrease roadway congestion, improve traveler safety and security, reduce transit systems' operating costs and support legislative mandates.

Now that we have had the opportunity to examine IVHS and its components, the question remains how does this specifically relate to the current document? There are several answers to this question. First, a tremendous amount of new data ("potential information") will be available as the programs in IVHS become reality. Current traffic signal control methodologies cannot utilize this additional information which leads directly to the second point. To properly implement IVHS, the ATMS system must be theoretically sound. ATMS requires a large *predictive* element.

Current systems are not designed to use that predictive element. In fact, Smith (1979) has shown that signal timing policies based on incremental responses to motorists changing routings may lead progressively *away* from the optimum. Procedures which do global optimization do not fair much better. There is a limit to how much computing one can perform on -line, using mixed-

integer linear programming as one example, using the incoming, real-world data. Chaudhary and Pinnoi (1993) have shown that, using mixed integer linear programming (MILP) formulations, the time required for even a moderate sized network to be solved "optimally" requires **hours** of CPU time on a mainframe! For reference, a moderate sized network would be on the order of 50 intersections and 20 arterials. The models on the market today make the fundamental assumption that the immediate future looks like the immediate past. When one sets traffic signals, it is to alleviate *projected* problems. Even if the traffic conditions *are similar or very close*, the problems on street segments and at intersections may not be solved by applying "optimal" solutions. Gazis (1992) demonstrated the existence of "phantom" bottlenecks which actually "flow" with the traffic. In other words, attempting to correct the bottleneck retrospectively is doomed to failure. The only reasonable way of dealing with this type of issue is to use an anticipatory system.

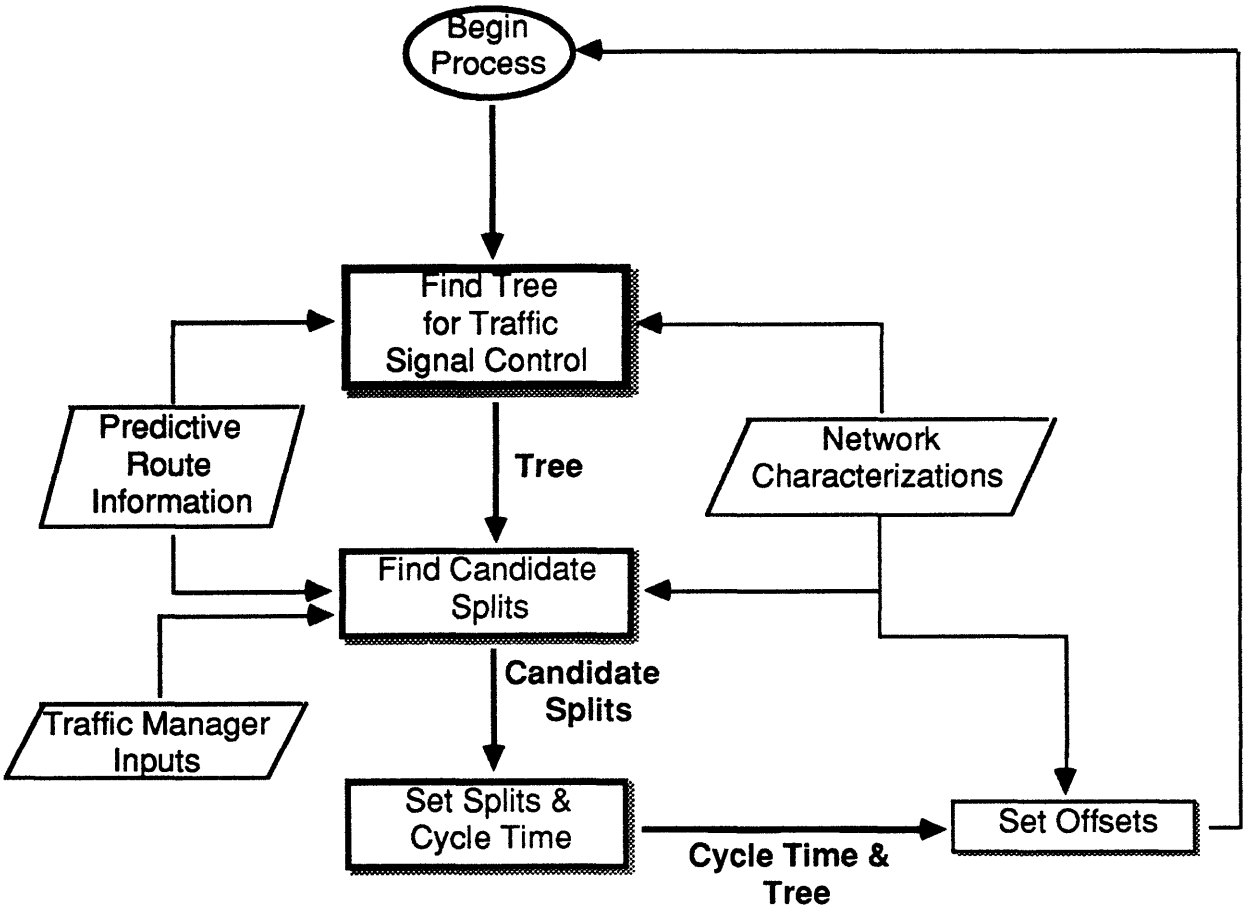
If one accepts the premises of the IVHS strategic plan (IVHS, 1992) then by the year 2004 some non-trivial fraction of the motorists on urban networks will have their routes catalogued with a central traffic coordination center under ATMS. How can this information best be utilized?

The framework for future traffic control systems should fulfill two requirements. One, it should present some system for determining how to integrate traffic signal settings in the area of control. Second, it should be able to quantify or bound the value of such a system to the theoretician or practitioner who would utilize such a framework.

SECTION 1.3 TSC System Structure

Since the thesis represents a first look into this new area, we will be primarily interested in *first order* results. The goal of the thesis is not to produce a system that is commercially marketable, but rather to examine the problem, establish a workable framework and evaluate its potential. Every vehicle and every driver on the roadways are different and have widely varying "system characteristics." Even the simple question of how will a random driver react to a prescribed set of information given a driving situation is *extremely* complex. So, we will be pleased to obtain workable, logical results which model PRI and TSC to the first order.

Rather than modifying an existing control system, we will tap into work done in network theory. The most closely related, recent work done by traffic researchers on the connection between graph theory and traffic management is the work of Wright, Appa and Jarrett (1989) concerning minimizing the number of crossings in an urban network to some graph theoretic limit through better design of urban streets. We will use an approach inherent in both PASSER-II and TRANSYT-7, the two most widely used commercial packages for TSC in the US. We will set the offsets and the splits separately. Figure 1.2 describes the flow of information used by the traffic signal control heuristics in this thesis.



Flowchart for data flow in PRISTINE

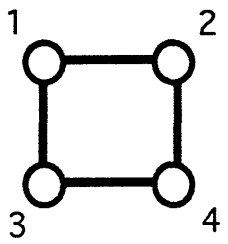
Figure 1.2

This begs the question *why* use this particular methodology? First of all, the thesis uses the master clock as the basis for traffic signal control. The master clock concept is that all traffic signals in the area of interest have the same total cycle-length. There has been a tremendous amount of research on dynamically varying cycle-lengths, but the master clock concept still has great potential for application in real world traffic (Gartner 1992). Cedar, Dressler and Ross (1989) developed a model for determining average delay at just *one* intersection with a

variable cycle time by modeling vehicle arrivals and red lights as two arrival streams where the red lights were priority customers. This clever model had some difficulties describing a single intersection with one-way flow; the paper did not begin to address the complexities of a network. We will explain the use of the tree structure in chapters two and three, but the rationale for setting the splits independent of the cycle time and offsets is that our procedure *inherently* incorporates a progression system for the most important street segments. There is also a psychological advantage to using a progression system. Drivers can react to a series of synchronized green lights. It gives credibility to the system in the minds of the motorists.

We rely for some information on the traffic manager; in particular, the traffic manager gives the model the minimum amount of acceptable green time along a major arterial and maximum average delay that is acceptable at an intersection in the network. Note that this is something that varies not only by easily measurably quantities such as traffic flow rate. One must also consider the psychology of driving in a particular area. Even with the same street network and the same amount of traffic, the minimum acceptable green period would almost certainly vary from driving in New York to driving in Kent, England or Bangkok, Thailand. For example, studies have shown that the maximum *acceptable* cycle time in the US is approximately 150 seconds; after that limit, drivers believe the traffic lights have broken down and start to do reckless things like running red lights. At the same time, it is routine to have cycle times over 5 minutes in Bangkok, Thailand.

We view the urban street grid, including arterials, as a network , in the mathematical sense. The entire grid can be described as the set of nodes, arcs, operations and sets of operations on the network. Before proceeding further, it is important to understand the concept of synchronization. Traffic signals A and B are said to be synchronized in the direction AB when a vehicle which leaves signal A at the beginning of its green cycle arrives at signal B during a green light. A good question to consider is if one wanted to *synchronize* the lights in a network then at worst case how many street segment, intersection combinations could be synchronized? As Morgan and Little (1964) stated, in the general case it is not possible to synchronize even a single street with numerous signalized intersections *in both directions simultaneously*. If one is very lucky, all the intersections can be synchronized each direction. For example, consider the network shown below.



In the sample, completely symmetric network shown above, the traffic manager is able to synchronize every arc in the network.

Figure 1.3

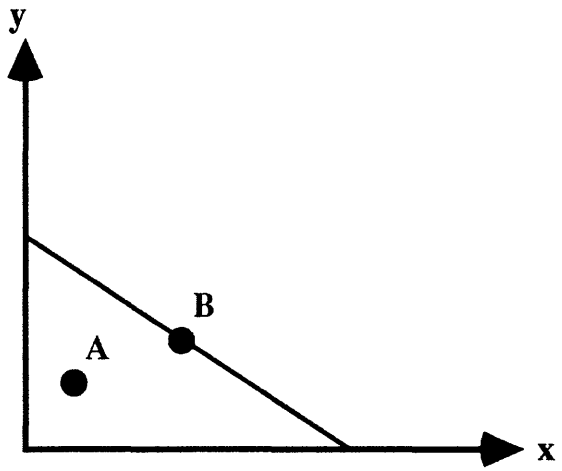
In the case where the figure is a square with equal nominal velocities on each arc in the diagram and without congestion effects, one can set the traffic signals at each intersection so that they are synchronized in each direction simultaneously.

SECTION 1.4 Determining "Optimality"

A natural query is how does one determine what is "optimal" for traffic signal control? There are nearly as many answers as there are motorists and traffic managers. Certainly from a self interested standpoint, we would all like our routes from workplace to home and vice versa as congestion free as possible. From an economical and social standpoint, minimizing the number of stops or the total delay for an entire network seem like good choices to stress in any "optimal" solution. Many traffic engineers would tell you that maximizing progression is the best overall method for controlling congestion. There are many factors which may be important to some parties which matter not all to other groups. For example, some groups may be interested in leaving certain "scenic" drives congestion free while other may be willing to have a substantially longer *average travel time* if the variance on the travel time is significantly reduced. There are no quick and easy answers as to what approach to take.

Multiobjective programming offers a framework to view some of the concerns. Specifically, the concept of a non-dominated solution is significant. Imagine we have identified all of the measurable factors which are important for determining the utility of a particular traffic signal timing plan. Clearly, you want to get solutions which make all the factors as good as possible. But, if there are constraints on the resources in the problem then eventually one reaches a point where you cannot improve one factor without worsening with respect to at

least one other factor. Such solutions are called non-dominated solutions. Consider the diagram shown below. Let the utility functions be the sum of the x and y components.



The diagram shown above demonstrates a Pareto frontier in multiobjective programming. If more is better for both the "X" and the "Y" axises then point "B" is better than point "A", because "B" has more of "X" and "Y". The line "B" is on represents a Pareto frontier. One must give up either "X" or "Y" to move from point "B" along the line.

Figure 1.4

Solution **A** is dominated by solution **B**, because both the x and the y factors increased. Solution **B** is in fact a *non-dominated* solution because given the constraints, it is not possible to increase the x or the y factor from solution **B** without decreasing the other. The line which contains solution **B** is called a Pareto frontier in multiobjective programming.

In practice, the factors which are important are difficult to determine in and of themselves, and getting a rating of the importance of the various factors in nearly impossible. In this thesis, we will explore the three most commonly used measures of effectiveness for traffic control schemes, average delay per vehicle due to traffic signal control, average number of stops due to traffic signal control, and progression (fraction of green lights encountered along specifically designated thoroughfares). These measures of effectiveness have varying utilities depending on the group scrutinizing them.

Chapter Two

Literature Review

SECTION 2.1 Introduction and Overview

Traffic signal control has a long history. This literature review cannot but splash the surface of the oceans of accumulated knowledge in this field. Thus, the purpose of this chapter is to place the current work in a historical perspective such that the reader can appreciate the state of the field. Additionally, relevant selections from the literature will also be cited in subsequent chapters where appropriate to establish grounding for the concepts described. We start with a historical look at TSC and then become acquainted with representative works in the various approaches to TSC. The interested reader is directed to Ballman (1991) which contains an extensive survey. One can hardly doubt the value of intelligent traffic signal control as Hauer (1994) pointed out in his article "Can One estimate the Value of Life or is it better to be Dead than stuck in Traffic?" He suggests that given a current estimation on the value of a human life as $1.5M and the value of time set at $6.71/hour, it is economically more beneficial that someone should die than endure the projected traffic delays and consequential costs associated with them .

SECTION 2.2 Historical Perspective on TSC and Graph Theory

The first use of graph theory for traffic control that we are commonly aware of was Euler's (1736) use of networks to route the King of Prussia's band across the seven bridges of Königsberg in 1736. The problem was to determine a path so that the royal band passed over each bridge once and only once. Euler showed that such a proposition was impossible. The modern use of graph theory had begun.

The first description of trees as mathematical structures related to graphs was by Cayley (1857). Cayley focused on the use of trees to represent special types of polynomials.

Modern traffic signals did not appear until 1868 where they made their first debut in London streets, and we did not see the advent of electric signal lights before 1914 when James Hoge installed several in Cleveland, Ohio. The first attempts to maximize progression by coordinating adjacent intersections occurred in 1922, and the world saw the advent of dynamically controlled traffic signals in 1928 when New Haven, East Norwalk, and Baltimore installed actuated signals (Homburger, 1988).

Engineers and mathematicians began working on traffic control problems almost immediately. The earliest work of note, using probabilistic methods to

analyze traffic was published by Kinzer (1933), which was followed quickly by Adam's (1936) report in the Journal for the Institute of Civil Engineers. Gerlough (1955) looked at modeling traffic using the Poisson distribution in a seminal paper.

Gerlough's paper was both broad and enlightening, and he was the first to formally establish bounds on the maximum throughput of a street given some Measure of Effectiveness (MOE.) Gerlough's goal was to discover the maximum number of cars which could pass through the crossing per hour and still allow the children to have 60 crossing opportunities per hour. This work assumed constant arrival rates and no queuing effects. His result for the crossing problem was:

$$\text{"critical" volume (cars / hour)} = V_c = \frac{29{,}000}{D}(2.322 - \log D).$$

Gerlough was joined by a colleague who also published in 1955. Schul (1955) published his paper "The Probability Theory Applied to Distribution of Vehicles on Two-lane Highways" which demonstrated some early attempts at predicting congestion and vehicle delays using "brute force" mathematical techniques. But, Schul's work did not go unnoticed. A young engineer at the British Road Research Laboratory used the techniques elucidated in Schul's work to perform some calculations of his own.

Webster (1958) published a paper called "Traffic Signal Settings". In this paper, Webster established the following useful relationships (Hobbs, 1979).

$$d = \frac{c(1-\lambda)^2}{2(1-\lambda x)} + \frac{x^2}{2q(1-x)} - .65x\left(\frac{c}{q^2}\right)^{\frac{1}{2}(2+5\lambda)}$$

d = average delay per vehicle on a particular arc
c = cycle time
q = flow (in vehicles per second)
λ = proportion of the cycle which is effectively green
x = degree of saturation = $\frac{q}{\lambda}$ [2.1]

$$C_0 = \frac{1.5L + 5}{1 - Y}$$

C_0 = Optimal cycle time to minimize average delay
L = Lost time (e.g. amber time + acceleration / deceleration time)
Y = Practical correction factor =.9-.0075L . [2.2]

Webster assumed Poisson arrivals with a constant arrival rate, uniform departure rates and a constant loss time during each cycle of 2 seconds, i.e. the amount of time lost due to switching lights, accelerating/decelerating, etc. In practice, Webster's formula works well. In fact, this will be one of this thesis' points of comparison in chapter six.

Graph theory never completely divorced itself from its first love of traffic management and routing. Holroyd and Miller (1966) demonstrated a statistical correlation between the number of lane changes and accident rates in urban

areas. Wright, et. al. (1989) later used graph theory to design networks which dramatically cut down on the number of crossing movements required, but these designs are unlikely to be put in practice as they consisted of infinite lane rings circumsected by an even number of radial connecting axis's. Later in the thesis, we will examine the use of spanning trees to help determine the offsets in the network, but this thesis is not the first work to examine the use of trees in the context of optimizing an objective function. Hutson and ReVelle wrote an article called "Maximal Direct Covering Tree Problems" which developed several formulations for sub-graph coverage; the applications to traffic signal control are direct with the proper choice of objective function. The major contribution of Hutson and ReVelle's paper is that it examined the use of non-spanning trees. That is they examined the possibility of excluding some nodes from the tree if the cost of covering the nodes were too high.

Approximately 30 years ago, one of the most widely used contemporary methods of setting traffic signals was developed. Little and Morgan (1964) wrote an article in Operations Research call "Synchronizing Traffic Signals for Maximal Bandwidth". This technique was translated into the MAXBAND computer program for signal timing (Little, 1966). At this juncture, it is best to leave a historical sequence and look at the four major current approaches to traffic signal control and how each developed.

SECTION 2.3 The Four Traditional Approaches to TSC

Taking a larger view of traffic signal control, there are four major approaches for traffic signal timing which are currently employed in the public sector. The first approach is maximizing "bandwidth" or amount of green time as vehicles progress in a given direction along a major thoroughfare. An example of a greenband is shown in the space-time diagram below. To better understand this and subsequent methods for TSC, it is beneficial to understand some of the nomenclature used to describe a particular TSC strategy.

The cycle time is the amount of time from the beginning of the red light in a given direction until the beginning of the next red light in the same direction. In our example below, the cycle time is 6 time units (e.g. seconds, minutes, etc.) The offset between two lights is the time difference between when the red light starts at one intersection in a given direction and when it starts at the intersection and direction you are comparing it to. In the example below, the offset between Main and Cedar streets is approximately one time unit. Now we can examine methods which maximize bandwidth in more detail.

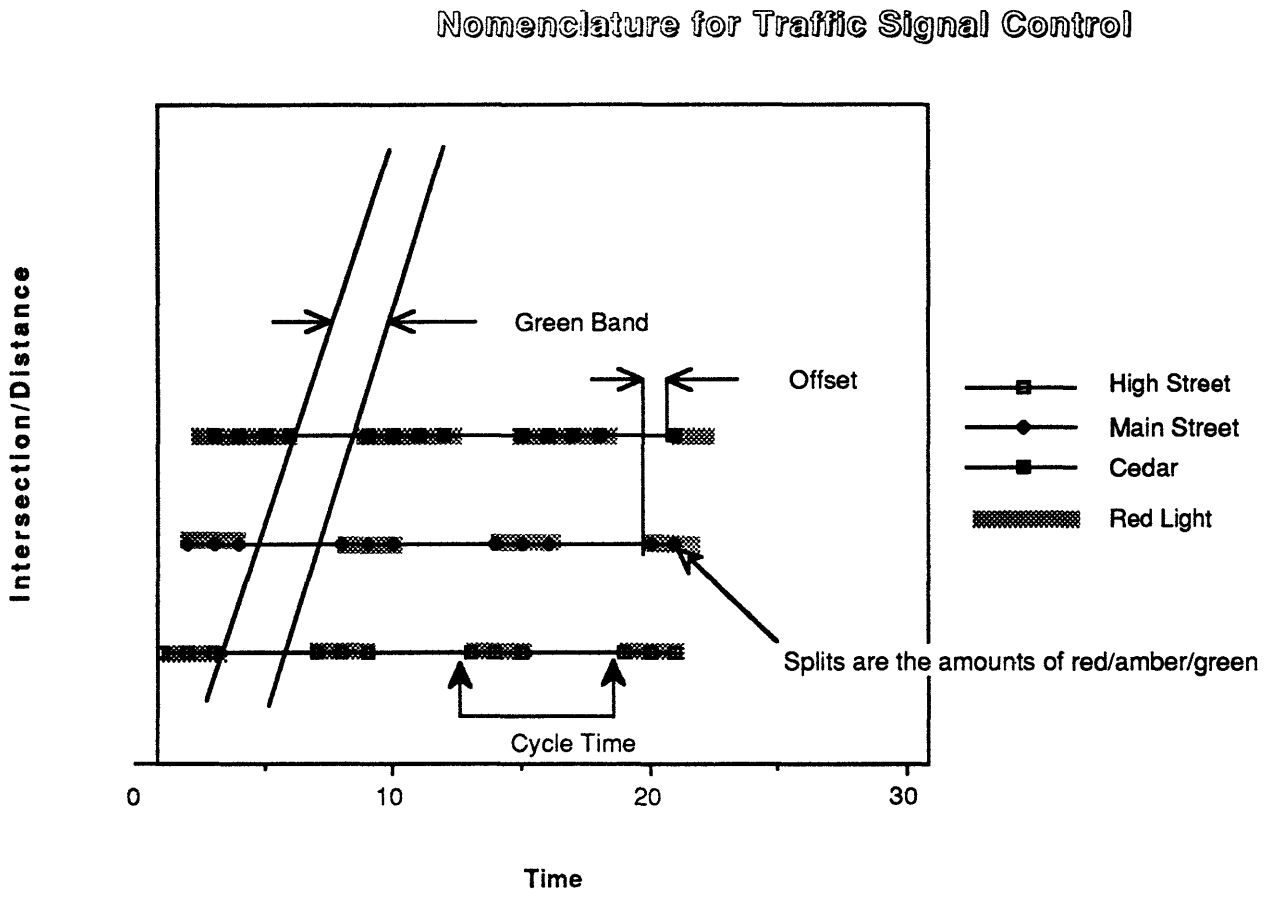
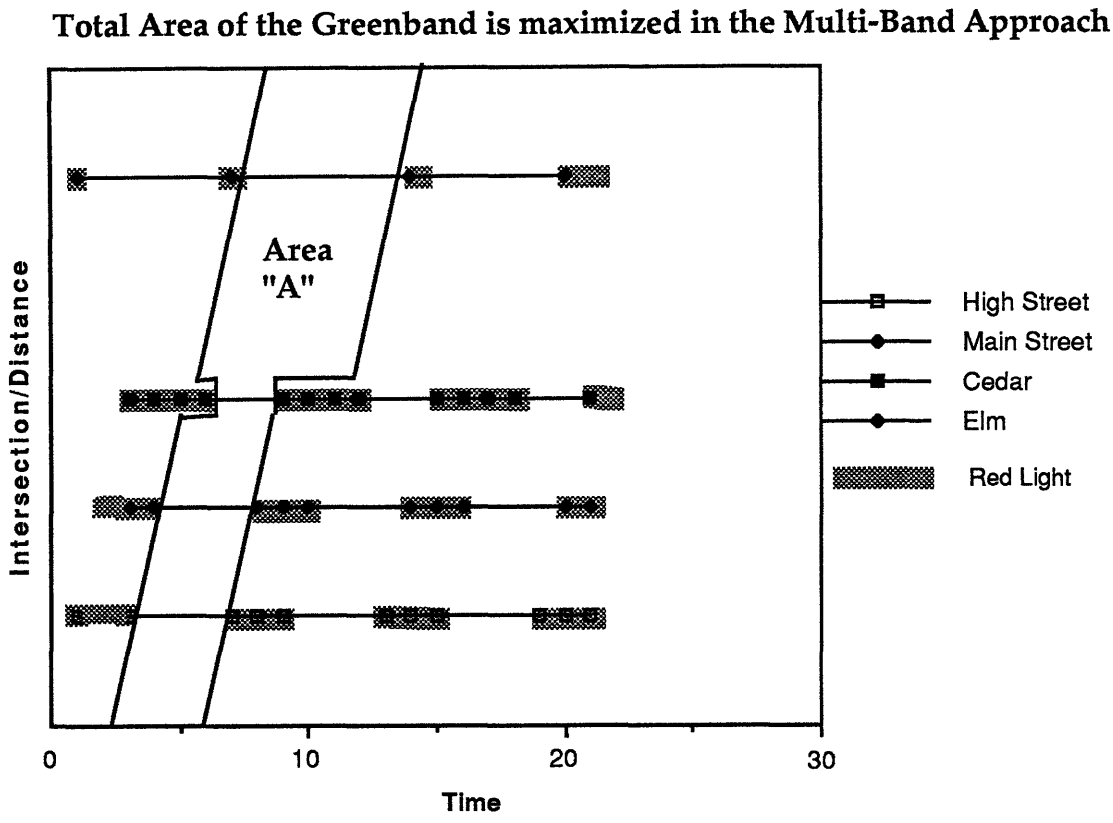


Diagram above pictorially represents the meaning of the terms: split, cycle time, greenband and offset for a simple three intersection street segment.

Figure 2.1

These methods maximize something called bandwidth, but maximizing the bandwidth inherently involves maximizing progression, the fraction of green lights a motorist encounters traveling along a specific route. The MAXBAND computer program (Cohen, 1982) sets cycle times, offsets and splits in order to maximize an affine combination of progressions for various user specified routes. The method uses a mixed-IP optimization technique to solve the problem.

Passer-II was developed by the Texas Transportation Institute (1984). Passer II examines turning movements, saturation capacities, distances, nominal speeds and queue clearance intervals. The main advantage Passer-II has over MAXBAND and its successors is that Passer-II implicitly considers the capacities of the streets in its calculations. Passer-II performs its heuristic in a two phase manner. First, it examines the demand-to-capacity ratios to set splits. Then, it sets offsets to maximize a user specified affine combination of route progressions. MULTIBAND is the most recent major system using progression as its primary solution (Gartner 1990). MULTIBAND also uses a mixed-IP solution technique. It differs from MAXBAND in that it does not require the bandwidths to be equidistant. Consider the diagram shown below.



Under the Multiband concept, one attempts to maximize the area of the greenband shown as Area "A" in the diagram above.

Figure 2.2

MULTIBAND seeks to maximize the *area* labeled "A" in figure 2.2. In this way, MULTIBAND implicitly decreases delays for motorists who are not part of a platoon along a thoroughfare.

The second set of TSC strategies seek to minimize delays and stops per vehicle across the entire network. TRANSYT is an example of the second class of TSC methodologies. TRANSYT was developed by Webster's organization, the Transport and Road Research Laboratory, United Kingdom (Robertson, 1969). TRANSYT-7F, the current US version of the historical TRANSYT, is the most widely used traffic signal control setting routine in the USA (Strong, 1991). TRANSYT also uses a combination of simulation and incremental improvement approach to calculate speeds, offsets and splits, but its objective function is minimizing total delay in the network (Hadi, 1992) as opposed to progression. All of the methods mentioned above have a common element in that each of them collects either physical data or vehicle usage data or both and calculates the optimal settings off-line. Thus, traffic signals set according to these routines alone would not vary with the current traffic conditions.

The third major grouping of traffic signal control techniques combines the first or second group with real time control (Hawat, 1992). Specifically, these methods will involve a collection of traffic data for a period of perhaps months, and then this data is divided into traffic patterns called *signatures*. An optimal traffic signal control setting is determined for each signature using one of the methods described previously. There are generally a large (over 100) number of signatures. During the day to day operations, the traffic control center collects data (which is recorded to be used during the next run of TRANSYT-7F, Passer-II, etc.) and compares this to the signatures, selecting the closest match. The third grouping also includes Critical Intersection Control (CIC). Here, selected "critical" intersections allow their cycle times and splits to vary dynamically based on observed traffic flows in real time; CIC typically uses pure Webster or

Newell formulas to determine appropriate settings. The third group is sometimes called generation 1.5 technology, falling somewhere between generation 1 , totally off-line optimization, and generation 2, on-line optimization with fixed cycle lengths for groups of intersections (Gartner, 1981).

The fourth grouping of traffic signal control strategies are third generation controls; there are no fixed cycle lengths, splits or offsets (Gartner, 1981). We will develop an optimized version of a third generation control system in chapter seven. The most widely implemented of third generation control system worldwide is called SCOOT (Robertson, 1991). SCOOT is essentially a real-time version of TRANSYT. SCOOT is main frame based, written in a high level language, CORAL. SCOOT samples current traffic conditions every four seconds and updates the timing on the lights every five minutes. Newell and Grafton proposed a continuous Dynamic Program (DP) in 1967 for traffic signal setting (Papageorgia, 1991), and Robertson and Bretherton proposed a discrete DP formulation as early as 1974. The problem with DP since its inception by Bellman in 1957 has been the "curse of dimensionality". In an effort to limit the extraneous data retained by the model, most traffic signal setting methods using DP invoke Optimal Sequence Constraints (OSCO). Essentially this means the models will only consider the last arbitrary number of traffic cycles. One model of this type which has gained widespread acceptance is Optimization Policies for Adaptive Control (OPAC) (Gartner, 1983). Shepherd (1994) wrote an *excellent* survey of currently used dynamic traffic signal control programs in his article for Transport Reviews where he examines the primary mechanisms, strengths and weaknesses of: OPAC, PRODYN, SAGA, SCAT, SCOOT, STAUKO and UTOPIA.

Note that each of the four proceeding methodologies all assumes that the immediate past has the same characteristics as the immediate future. The methods are not specifically designed to take advantage of the predictive information which will be available under IVHS. The MOTION traffic signal control program written under the auspices of the European DRIVE program (Busch, 1993) makes use of the real time predictive routing information. Central to this thesis is the real time availability of accurate routing information for vehicles entering and proceeding through the network. Several current approaches are examining the difficult area of collecting and processing such PRI. Kaufman and Smith (1993) proposed such a system in their article "Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highway Systems". Another process is presented by Leonard, Ramanathan and Recker (1993) in "A Real-Time Information Processing Algorithm for the Evaluation and Implementation of ATMS Strategies". This thesis examines ways of applying the information from the ATIS and ATMS data bases to determine the TSC; between the thesis' problem and filtering/setting up the data bases to hold the PRI is the problem of actually selecting the routes the vehicles travel on. In the article "Algorithms for efficient Real-Time Traffic Assignment", the authors describe a state of the art, fast algorithm for optimal routing or traffic prediction under ATMS/ATIS (Jayakrishnan, R., Tsai, W., et.al., 1993). Perhaps the most notable contribution of the paper in light of the work done is this thesis is that the paper advocates the use of a gradient projection algorithm for solving for traffic flow problems. We will make use of the same technique in a slightly different application in chapter seven to solve our non-linear program. Kim and Hobeika (1993) have developed a model using a Auto Regressive Integrated Moving

Average (ARIMA) model based on collected traffic data. Last, Nelson and Palacharia (1993) have developed a neural network model for estimating current travel times and performing data fusion under the ADVANCE traffic management program. The system uses a counterpropagation neural network to solve the travel time data fusion problem.

The current techniques for traffic signal control have been developed and re-engineered over the past twenty years. Each of them is very complex. It is difficult if not impossible based on the heuristics' complexities to place bounds on how much the presence of these control techniques improves common MOE's for traffic. In chapter five, the thesis will also propose a method for estimating delays, stops and progression for vehicles traveling along uncoordinated routes assuming statistical independence and using convolutions. The methodologies proposed in the thesis are simple enough that they could be implemented on a microprocessor.

SECTION 2.4 Traffic Simulation and Testing

Traffic signal control is an area that naturally lends itself to simulation. We will use simulation techniques to test PRISTINE. In this section, we will take an overview of some of the more common and recent simulation techniques, and we will conclude the section by examining methods of testing the traffic simulations themselves.

The most commonly used and cited traffic simulation model is TRAF-NETSIM. We will discuss TRAF-NETSIM in greater detail in chapter seven. In brief, TRAF-NETSIM was first developed under a grant from the Federal Highway Administration (FHWA) twenty years ago and has been continually enhanced since that time (Rathi and Santiago, 1990). It integrates such features as: pedestrian crossings, 16 vehicle types, three driver types, transit operations, interactions of adjacent vehicles, effects of blockers and parkers, spillbacks, etc. TRAF-NETSIM sets the standards for realism and robustness for traffic simulation, but there are other simulations in the literature.

Most traffic simulations are developed for very specific applications. Roke Manor Research Limited in England developed a simulation to examine the interaction of vehicles guided by the EURO-SCOUT system and those vehicles without the on-board guidance aid (Harris, S., Rabone, A., et.al., 1992). The simulation developed was called ROute GUidance Simulation (ROGUS). Liu and Kanaan (1992) developed a model call CORFLU which is an integrated traffic simulation system for corridors. CORFLU is part of the TRAF system of traffic models. Huijun and Fu (1992) developed the Urban Traffic Simulation Model (UTSM) to examine the inner ring viaduct project in Shanghai. Willumsen, Bolland, et.al. developed SATURN and the SAturn Travel CHoice MOdel (SATCHMO) models to simulate congested traffic. SATURN and SATCHMO are designed to evaluate the effects of public transportation schemes, road pricing, traffic calming, and safety measures on congestion. Sullivan, Staley and Taff (1993) developed an off-line testing method for examining proposed

ATMS and ATIS applications. The program they developed was called Mesoscopic Event-driven Traffic Simulation (METS). The DYnamic Network Assignment Simulation Model for Advanced Road Telemetrics (DYNASMART) is a simulation package that seeks to integrate the route selecting aspects of IVHS with some freedom of driver choice. In chapter seven, we develop our own simulation model, LS3, for testing the integration of PRI with dynamic traffic signal control. One area of importance that does is not extensively covered in the literature is how to test a traffic simulation to determine suitability.

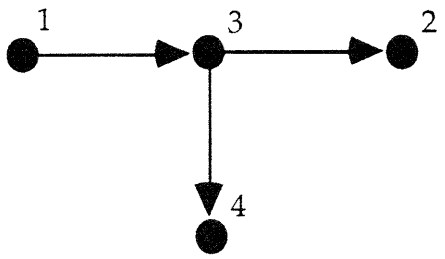
We found two papers in the literature which addressed issues surrounding simulation that would prove applicable in the area of traffic and TSC modeling. The first of these was a paper by Barlas (1989) called "Multiple Tests for Validation of Systems Dynamics Type of Simulation Models" which described how to test not only the structural basis of simulation models but also the *behavioral* aspects as well. We will examine Barlas' paper in more detail in chapter seven where it forms the basis of our examination of the LS3 simulation model. Towsley and Heidelberger (1989) wrote an article called "Sensitivity Analysis from Sample Paths using Likelihoods" which could be used to see if the vehicles were following the correct paths in the simulation, but we will use a virtual 100% sample of the vehicles exiting the network to verify that the vehicles followed the correct paths in chapter seven. But, if one was faced with an unknown simulation platform, a "black box", this could prove an effective technique for evaluating the model's structural integrity.

Chapter Three

Basic Mathematical Model, Notation and Assumptions

SECTION 3.1 Definitions and Notation

One can represent an urban street complex in terms of a network model, G(N,A). G(N,A) consists of several sets. *[Note: there is a glossary at the end of this document.]* "N" is the set of nodes which correspond to the intersections with traffic signal control. Let "n" be the number of elements in **N**, i.e., n=‖**N**‖. "**A**" is the set of <u>one-way</u> arcs corresponding to the individual traffic lanes in the network. Let m=‖**A**‖. "**D**" is a matrix whose elements correspond to the distances between nodes along arcs which are elements of set **A**. Consider the simple urban street complex shown below.



A={(1->2),(2->3),(2->4)}
N={1,2,3,4}
m=3
n=4

Sample diagram above demonstrates node and arc sets.

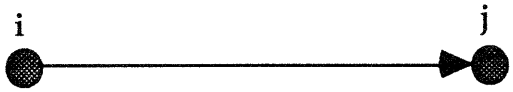
Figure 3.1

Let **V** be the matrix whose elements V_{ij} represent the speeds at which vehicles traverse the arcs $(i \rightarrow j) \in A$. Thereby, the time it takes a vehicle to go from node **i** to node **j** along arc $(i \rightarrow j)$ is given by:

$$\hat{t}_{ij} \equiv \text{time from "i" to "j" along arc } (i \rightarrow j) = \left. \frac{d_{ij}}{v_{ij}} \right| \begin{matrix} (i \rightarrow j) \in A \\ v_{ij} \neq \emptyset \end{matrix} \quad . \qquad [3.1]$$

The arcs of G(N,A) are traversed by vehicles. To represent vehicular travel, let X_i be the vector representing vehicle i's path as it traversed G(N,A). Specifically, $X_i(j) \equiv$ the j^{th} node visited by vehicle "i". Corresponding to X_i, let T_i be the vector of times for vehicle "i" during its travel through G(N,A). Here $T_i(j) \equiv$ the time vehicle "i" arrived at the j^{th} node on its path through G(N,A).

Consider the following illustration:



A simple one-way arc with nodes "i" and "j"

Figure 3.2

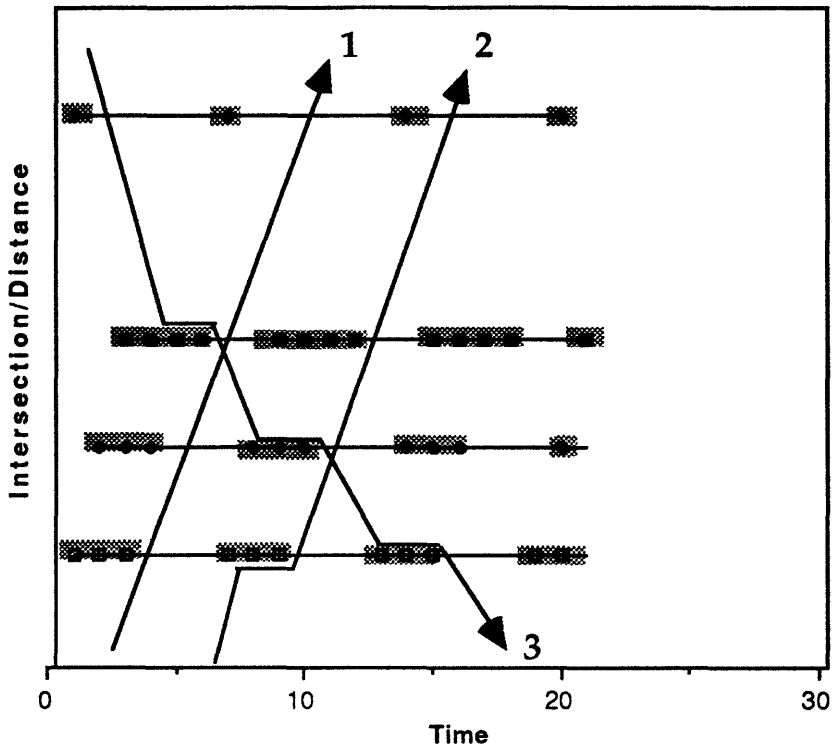
Let us say that node **i** is the third node that vehicle **k** arrives at, and the vehicle continues unimpeded to node **j** Then we have the following relationships:

$X_k(3)=i$ and $X_k(4)=j$, and if there are no delays, $T_k(3)+\hat{t}_{ij}=T_k(4)$.

Let **R** represent the set of routes through **G(N,A)**. Let $n_R \equiv \|R\|$. Specifically,

$$R_i(j) \equiv \text{the } j^{th} \text{ node on route "i".}$$

We will define the desired usage (vehicles per unit time) of the routes to be the vector Λ where λ_i represents the desired usage rate of route "i". One can introduce the concept of traffic signal control on G(N,A) by defining the *cycle length, off-sets,* and *splits* for the network. The cycle time refers to the amount of time between the beginning of successive green cycles at a given intersection for a specific direction of travel. Under the *master clock concept,* all signal control units have the same cycle time. Offsets represent the difference between the beginning of a green cycle for a specific intersection going a designated direction and the master clock zero. ***Throughout this thesis we will use the term splits to refer to the amount of green time in seconds in a particular direction.*** Figure 3.3 (below) describes the traffic signal control for an street with four traffic signals.



The space-time diagram above shows sample paths for vehicles 1, 2 and 3. Vehicle 1 was able to pass through all four intersections without stopping while vehicle 2 stopped once. Vehicle 3 is moving in the opposite direction.

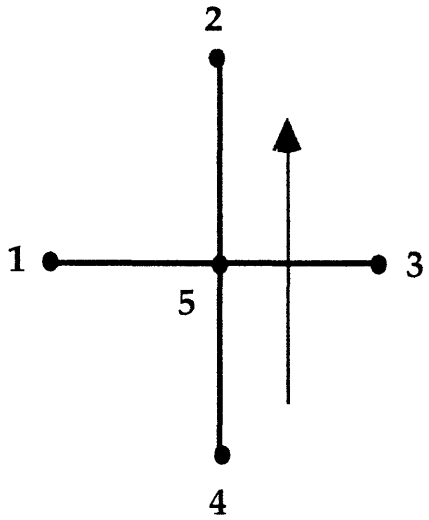
Figure 3.3

Figure 3.3 demonstrates the use of the space-time diagram to plot the relationship between cycle time, splits and offset. The horizontal lines on the figure represent signals seen from a given intersection *in a fixed direction.* The small shapes (e.g. squares, circles and triangles) on the lines represent times when the light is red. Note the three vehicle sample paths shown. Vehicle 1 goes through all four street lights unimpeded while vehicle 2 must stop at one light,

and vehicle 3, moving the opposite direction, must stop at three lights. The slope of the paths at any instant "t" represent:

$$\frac{\textbf{Distance}}{\textbf{Time}} = \textbf{Velocity}$$

Let Θ represent the vector of offsets for the intersections in **N** and θ_{ij} represent the offset for the beginning of the green light from the master clock zero for intersection **j** when traveling from intersection **i**.. Why use two subscripts? The question really is how can one uniquely specify the off-set at a particular intersection? For example consider Figure 3.4 (below).

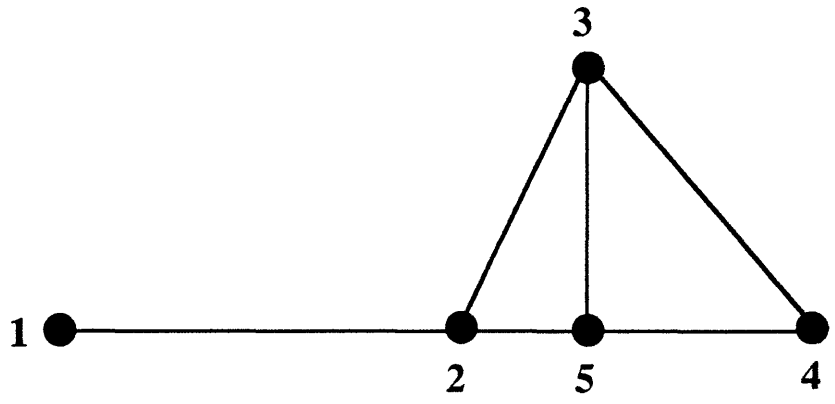


An offset for node 5 is not *completely* specified unless we have *both* a quantity *and* an orientation. In this case, we specify both a quantity and indicate this is the offset as seen by a vehicle travelling from node 4 to node 2 through node 5.

Figure 3.4

A value for θ_5 does not in and of itself give you the actual offset for the intersection; θ_5 is ambiguous, unless it includes some type of orientation along with the numeric offset. If one includes the orientation, the offset for node 5 is θ_5 *when traveling from 4 to 2,* the offset is completely specified.

This leads directly into an alternative offset designation which includes a defining spanning tree, **S.** A spanning tree is a set of arcs which connect all nodes and contains no cycles. Now, select a node $i^* \in N$. (By the definition of a spanning tree, i^* will be a member of any spanning tree.) The Θ vector represents the offsets at the intersections when traveling *from* i^* along **S,** the spanning tree. By convention, $\theta_{i^*} = 0$. Note, every spanning tree contains (n-1) arcs. **There is a *unique* path between any two point in G(N,A) through a spanning tree.** Ergo, any classification of off-sets is uniquely determined once we specify the point (i.e. the root node, i^*) where we begin tracing our paths from. Consider the G(N,A) shown below.



If node 1 is the root node, we still have multiple paths to reach node 5. We need to uniquely specify a path to set the offset from the root node.

Figure 3.5

If one did not define Θ *with* **S**, there would be confusion as to when the green cycle started. Specifically, would θ_5 refer to the green cycle when traveling *through 5 from intersection 2 to intersection 4* or would it refer to *traveling from intersection 3 to intersection 4 through intersection 5*?

One of the most important consideration for determining traffic signal timing in the master clock environment is the master clock cycle length. Let **C≡**cycle length for G(N,A). Then C refers to the time from the beginning of a green cycle to the beginning of the next as a specified intersection in a given orientation.

One can make interesting observations about the vehicles traveling through the network as well. Assume vehicles arrive independently and each vehicle arriving to the network has a specific route to follow *upon arrival to the network* . (One may or may not know this route a'priori.)

If one observes the network for some fixed time interval, he can *estimate* the *desired usage* of any particular street segment. Specifically, start observing G(N,A) and the vehicles passing through the network at time 0 and stop observing at time t.

Define the observed usage of the arc $(\mathbf{i} \rightarrow \mathbf{j})$ to be:

Vehicles observed on street segment (i → j) / **Time during which (i → j) was observed** , or in more symbolic terms, we have

$$\lambda_{ij}(0,t) = \frac{\gamma_{ij}(0,t)}{t} \qquad . \quad [3.2]$$

where $\gamma_{ij}(t, t+\Delta t)$ is the counting variable:

$$\gamma_{ij}(t, t+\Delta t) = \sum_{\forall X_k(l)=i, X_k(l+1)=j, [T_k(l)\le t \cap T_k(l+1)\ge t+\Delta t]} (1) \quad . \quad [3.3]$$

The expression counts every occurrence where a vehicle was traveling across $(i \rightarrow j)$ between times t and $t+\Delta t$.

One can also *estimate* $\lambda_{ij}(0,t)$ by considering the point vehicle flow rate along arc $(i \rightarrow j)$ and use that value as the representation for the entire time period. Now $\lambda_{ij}(0,t)$ is a monotonically increasing function of the flow rate on the arc if the average velocities of vehicles along arc $(i \rightarrow j)$ are fixed. For a fixed time, t, one can abbreviate $\lambda_{ij}(0,t) \triangleq \lambda_{ij}$. It will be useful to develop some method of measuring traffic signal settings' effects on specific sample paths in the next section. To do this, we need some measure of the number of street segments and intersections a given sample path, X_i passed through while traversing G(N,A). Define $n_i \equiv$number of nodes X_i passed through in G(N,A), i.e. $n_i = \|X_i\|$. It is evident that, because X_i contains no cycles, the number of street segments traversed in G(N,A) equals n_i-1.

SECTION 3.2 Assumptions

We make the following basic assumptions.

(1) The cycle time, C, for signalized intersections in G(N,A) is constant.

(2) A vehicle will travel at a constant speed along arc $(i \rightarrow j)$ of v_{ij}.

(3) Velocity changes are instantaneous.

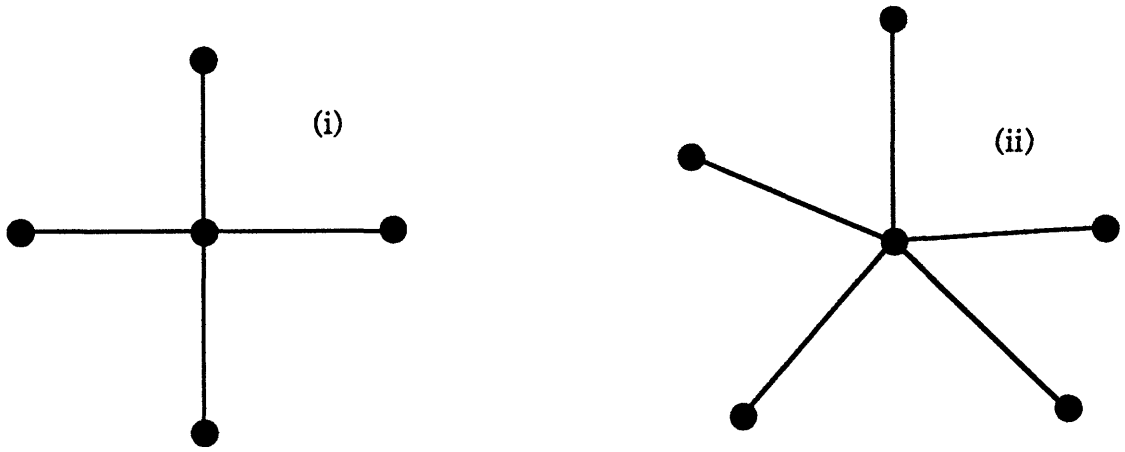
(4) Traffic is of light intensity; every vehicle which is stopped at an intersection for a red-light cycle will pass through the intersection on the subsequent green cycle.

(5) Drivers follow routes through **G(N,A)** which do not intersect themselves; these routes are all contained in **R**. Specifically, $(\forall X_i \in R) \cap [\exists i,l,k \text{ s.t. } X_i(l) = X_k(k) | l < k]$. Note: the X_i are still sample paths, but the set of all possible sample paths (with non-zero desired usage) are contained in **R**.

(6) One knows or can easily measure Λ, a vector containing the drivers' desired usage rates (in vehicles per unit) for each route. In particular one has:

$$\Lambda \ni \lambda_i \equiv \text{desired usage of route } \mathbf{R}_i \in \mathbf{R}.$$

(7) There are a maximum of two conflicting directions at each intersection, i.e. one can have up to four way intersections, but two of the streets are *aligned*. By aligned, we mean the streets share green cycles. Consider the diagram shown below.



The intersection to the left requires only two green splits to accommodate all of its incoming arcs while the intersection to the right has to have a minimum of three separate green splits.

Figure 3.6

The first case meets the criteria of assumption (7), but the second case has too many incoming street arcs which do not share common green splits.

SECTION 3.3 Measures of Effectiveness (M.O.E.'s)

Given a network and attendant traffic signal control parameters, **G(N,A)**, **D**, Θ, **S**, **C**, **R**, Λ and a set of sample paths $\{X_i \mid i \in \{1,...,I\}\}$, one can rate the efficiency of the particular traffic control plan using the following measures of effectiveness: progression, average delay, average number of stops, average time in system, and expected delay per vehicle-intersection.

(a) **Progression** is a measure of the ability to travel along a selected artery without traffic signal delay (Homburger, 1988). Progression is a measure that has a large psychological appeal. Drivers can relate to progression systems. Drivers will adjust their pace to accommodate the offsets for the green lights. Since progression is defined for an artery, one must first define $\mathbf{A_r}$ which is the set or arcs comprising the artery. In this thesis we will define progression to be a measure of the number of green lights that a vehicle would encounter beginning at the first intersection in $\mathbf{A_r}$ at the beginning of the green cycle and traveling at the speed limit through each intersection in $\mathbf{A_r}$ divided by the total number of green lights possible. Specifically:

$$\textit{Progression} \equiv \frac{\textit{Actual Intersections Passed without Stopping}}{\textit{Potential Intersections Passed without Stopping}}$$

Note that progression is a function of a particular *route or artery* and does not depend on the actual traffic flow. Progression is *only* a function of the speed limits along the route and the traffic signal settings.

The highest (best) progression rating is 1, the lowest is 0.

(b) **Minimize the Number of Stops** is another common objective when evaluating traffic signal control. Many groups are interested in the number of stops. From an environmental standpoint, the number of stops is correlated to the amounts of pollutants emitted. A safety engineer would be quick to point out that the number of accidents is related to the amount of starts and stops a

vehicle has to make in light to medium traffic. Minimizing the number of stops is much like progression, but whereas progression is specified for a particular path (e.g. the arterial), minimizing the total number of stops is a global objective. i.e., it looks at all of G(N,A) not just (i,j)$\in A_r$. For illustrative purposes, we will look at the average number of stops from a functional standpoint and examine how one might actually determine the observed number of stops from a sample path. How does one determine how many stops a particular sample path experienced due to traffic signal delays? Assume that when a vehicle experiences a green light at an intersection, the vehicle immediately moves to the next intersection, without delay (except for the amount of time, $\hat{t}_{ij}$, it takes to traverse the street segment, i.e. ignore queue effects for the moment). Thereby, if a vehicle is traveling from node i to node j in G(N,A), we saw in equation 3.2 that the travel time is just $\hat{t}_{ij}$. Thus, one can count the number of delays any particular sample path X_i experiences by using the following relation.

$$\Delta_i \equiv \text{Number of Delays from Traffic Signals} = \sum_{\forall T_i(l) \text{ s.t } T_i(l) > T_i(l-1) + \hat{t}_{X_i(l)X_i(l-1)} \mid l \in \{2,..,n_i\}} (1) \quad . \quad [3.4]$$

This is another expression with a counting variable as the central feature. In this case, the variable counts all those times when the vehicle took longer to get from one intersection to the next than it would have without delays.

Alternately, define an indicator variable $\delta_i(l)$ where:

$\delta_i \equiv$ Indicator variable for event vehicle "i" stopped at intersection "l" due to traffic signal control.

We define the indicator variable the same way we did with the counting variable above. The value of the indicator is **one** whenever the vehicle is delayed on the street segment, i.e. it takes longer than is required to traverse the street segment.

$$\delta_i(l) = \begin{cases} 1 \text{ if } \exists j \in \{1,\ldots,n_i\} \text{ s.t. } T_i(l) > T_i(l-1) + \hat{t}_{X_i(l)X_i(l-1)} \mid l \in \{2,..,n_i\} \\ 0 \text{ else} \end{cases}.$$

Now, let $N_s \equiv$ Number of sample paths evaluated$\mid N_s \leq I$. Then,

$$\Delta_i = \sum_{l=1}^{m} \delta_i(l). \qquad [3.5]$$

If we knew the sample paths for the time frame of interest in advance, the objective could be described as:

$$\underset{\theta,T}{Min} \sum_{i=1}^{N_S} \Delta_i \qquad . \qquad [3.6]$$

One can define the average stops per vehicle in G(N,A) as:

$$S_{G(N,A)} \triangleq \frac{\sum_{i=1}^{N_s} \Delta_i}{N_s} \qquad . \qquad [3.7]$$

Even in this simplified model, the number of stops per vehicle is a function of many things including: the network characteristics such as physical

construction and speed limits; the traffic signal plan such as the offsets and splits; and the nature of the arrival process for vehicles into the system such as the average number of vehicles arriving per unit time and how the arriving vehicles are spread out in terms of time between vehicles. We will make some assumptions about characteristics of this arrival function in later chapters.

Another perspective is to examine the effects of any particular traffic signal plan on the average stops at a given intersection per sample path, using that intersection. Let $\bar{S}_\theta$ be the vector, arranged by traffic signal, containing that data. Again examining sample paths to illustrate the measure we have:

$$\bar{S}_\theta(j) \triangleq \sum_{i=1}^{N_s} \left(\frac{\delta_i(j)}{\sum_{k=1}^{n_i} \sum_{X_i(k)=j} \langle 1 \rangle} \right) \quad . \qquad [3.8]$$

This particular M.O.E. does not discriminate against longer paths. Both minimizing the average delay and minimizing the average number of stops tend to favor situations with shorter routes. Why? The M.O.E.'s are not only a way of measuring absolute performance, but they are also a way of comparing various traffic signal plans. If we said we had a traffic control system that gave us an average of one stop per vehicle (on the average), we may be tempted to say that sounds like a relatively good system, but would it make a difference if we said the average driver passed through only two intersections in this network. Would it sound better if we said the average driver passed through 35 intersections while passing through the network? So, the second measure normalizes the number of stops across the number of intersections the vehicles pass through.

(c) **Minimize the Average Delay** is another commonly accepted measure of effectiveness for traffic signal control. Average delay is most closely related to costs. The amount of fuel consumed during a trip through the network is most closely related to the delays experienced. Processes that rely on speedy delivery of products are subject to adverse effects from travel delays. There is an adage that time is money, and a company stands to lose a significant amount of money if one of its influential executives is stuck in traffic. Again we will examine this M.O.E. from a sample path and purely functional standpoint. One can easily see that delay occurs at intersection "j" for sample path "j" iff $\delta_i(j)=1$. Now, the total delay per vehicle is described as:

$$\frac{\text{(Actual total time to pass through network - Amount of total time to get through with no delays)}}{\text{Number of Vehicles}}$$

$$T_{G(N,A)} \triangleq \frac{\sum_{i=1}^{N_S}\{T_i(n_i) - \sum_{j=2}^{n_i}\hat{t}_{X_i(j-1)X_i(j)}\}}{N_S} \quad . \qquad [3.9]$$

From a functional standpoint, the average delay per vehicle would be dependent on the network characteristics such as physical construction and speed limits; the traffic signal plan such as the offsets and splits; and the nature of the arrival process for vehicles into the system such as the average number of vehicles arriving per unit time and how the arriving vehicles are spread out in terms of time between vehicles. As with average number of stops per vehicle, we must make some assumptions about the arrival process to further analyze it. We will do this in subsequent chapters, and in particular we will use a bulk arrival with exponential times between arrivals in the Queue Effects Model (QEM). Later in the LS3 simulation in chapter seven, we will evaluate the delay per vehicle in

terms of the amount of time the average vehicle is travelling less than five MPH. There is a high correlation between average transit time and average delay as we will see in chapter seven. Our first area to examine will be how to set the offsets in such a way as to set up an inherent progression scheme? We will explore the answer to this question in the next chapter.

Chapter Four

Heuristics for Determining Offsets

SECTION 4.1 Assumptions and Derivations

Now that the symbolic groundwork has been laid, we can examine heuristics for traffic signal setting. Remember, we have made several "first cut" assumptions. *Many of those assumptions will be relaxed in later models.* It is also significant to note that we assume that some cycle-time exists prior to actually setting the splits, but in actuality, we will perform the split setting in two separate operations. In chapter four, we will explore three separate heuristics for determining the tree. Then, we will use the predictive route information in conjunction with the tree to determine the cycle time and splits for the network; this will be explored in chapters five and six. Last, we will set the splits based on **C** and **S**. Let us begin by examining the assumptions we made in chapter 3 again.

SECTION 4.1.1 Assumptions

(1) The cycle time, C, for signalized intersections in **G(N,A)** is constant throughout the network, i.e. every intersection in the network has the same cycle time.

(2) A vehicle will travel at a constant speed along arc $(\mathbf{i} \rightarrow \mathbf{j})$ of v_{ij}.

(3) Velocity changes are instantaneous.

(4) We will assume traffic is of light intensity; just as in chapter one, our definition of light traffic implies that every vehicle which is stopped at an intersection for a red-light cycle will pass through the intersection on the subsequent green cycle. Vehicles which arrive at an intersection during a green light will pass through that intersection unhindered.

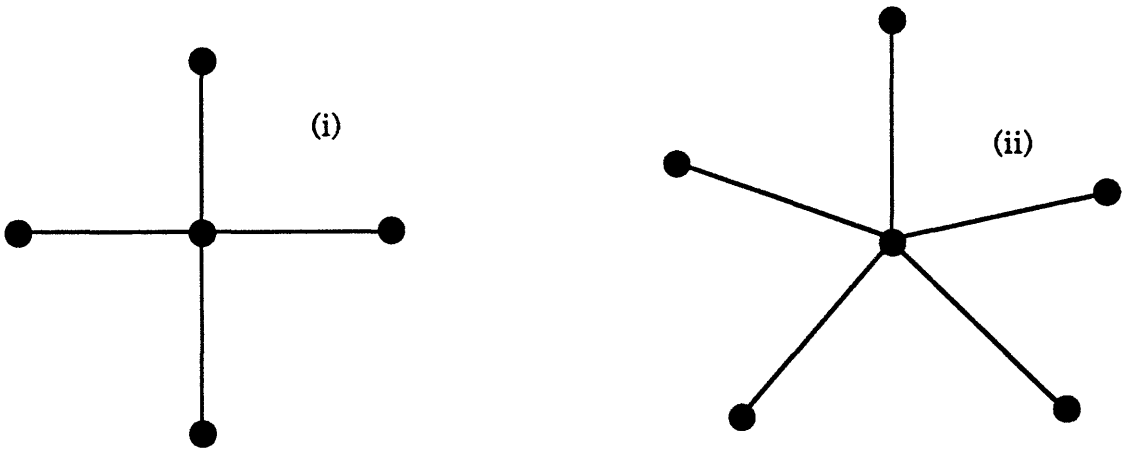
(5) Drivers follow routes through **G(N,A)** which do not intersect themselves; these routes are all contained in **R**. Specifically, $(\forall X_i \in R) \cap [\nexists i,l,k \text{ s.t. } X_i(l) = X_k(k) | l < k]$. Note: the X_i are still sample paths, but the set of all possible sample paths (with non-zero desired usage) are contained in **R**.

(6) One knows or can easily measure Λ, a vector containing the drivers' desired usage rates (in vehicles per unit) for each route. In particular one has:

$$\Lambda \ni \lambda_i \equiv \text{desired usage of route } \mathbf{R}_i \in \mathbf{R}.$$

(7) There are a maximum of two conflicting directions at each intersection, i.e. one can have up to four way intersections, but two of the streets

are *aligned*. By aligned, we mean the streets share green cycles. Consider the diagram shown below.



The intersection to the left requires only two green splits to accommodate all of its incoming arcs while the intersection to the right has to have a minimum of three separate green splits.

Figure 4.1

The first case meets the criteria of assumption (7), but the second case has five *non-aligned* directions, which is more than allowed by assumption (7).

(8) The amount of red and green time is evenly split for each direction at each light with a value of **C/2** for each.

SECTION 4.1.2 Derivations

Knowing Λ along with **R**, one can estimate λ_{ij} for each arc. Call this estimate $\hat{\lambda}_{ij}$. Now,

$$\hat{\lambda}_{ij} \triangleq \sum_{\substack{X_l(k)=i, X_l(k+1)=j \\ \forall l \in \{1,\cdots,n_R\} \\ \forall k \in \{1,\cdots,(n_l-1)\}}} \lambda_{ij} \quad . \qquad [4.1]$$

As we discussed in section 1.4, one can only truly describe an *optimal solution* in the case of a single objective function. In the case of multiple objectives, as in the multiple M.O.E.'s of traffic signal control, one can only describe "non-dominated" solutions (such solutions are said to lie on a Pareto frontier). That is solutions where you cannot improve one M.O.E. without hurting another. One technique in multi-objective programming is to weight conflicting objective functions by taking affine combinations of them. Consider the case where one is interested in finding a multi-objective non-dominated solution where we weight the progression rating, average stops per vehicle, and average delay per vehicle using α, β and γ respectively. Let α, β, $\gamma > 0$, and $\alpha+\beta+\gamma=1$. Then, for a *known set* of **X**, sample paths, and using our descriptions of the M.O.E.'s from chapter three, we could describe our objective function, **F**, as:

$$F(\bullet) = [\frac{\alpha}{\|A_r\|} \sum_{i=1}^{N_s} \sum_{l=2x_{i,(l-1)}x_{i,l} \in A_r}^{n_i} \sum [1 - \delta_i(l)] - \beta \frac{\sum_{i=1}^{N_s} \Delta_i}{N_s} - \gamma \frac{\sum_{i=1}^{N_s} \{T_i(n_i) - \sum_{j=2}^{n_i} \hat{t}_{X_i(j-1)X_i(j)}\}}{N_S}] \quad ,$$

Objective Function 4.1

and the problem of selecting offsets for the traffic signal control scheme could be described as:

$$(\Theta, S) = \underbrace{\arg\max}_{\vec{0} \le \Theta \le \vec{e},\ S \subseteq A,\ \|S\| = n-1,\ S \text{ is a tree}} F(\bullet) \quad .$$

The actual outputs of this program would be an optimal set of splits and **S**; throughout the remainder of the thesis we will refer to *the value of math program* 4.1 as the value of the objective function for the program. The theoretical maximum of the objective function is α. *If we select our arterials to be the same routes selected to be part of the tree, S, then we automatically give ourselves a progression based system.* [Note: the arcs chosen to measure progression along do not have to correspond to an actual route at all.] One may wish to consider the main thoroughfare as the arterial for progression purposes, and although many vehicles may travel along it during part of their journey across **G(N,A)**, few if any of the vehicles will use it the entire way.] Then the points of comparison are between which TSC plans give the smallest overall delays or stops.

We are not going to explicitly solve the math programming model (see above) in this thesis, because to adequately do so would require us to *explicitly* know the relationship between: drivers, the vehicles they drive, their routings, driver behavior, congestion, driver interactions, unforessen contingencies, the traffic signal settings, etc. which is not possible (and probably not even legal). We *will* use objective function 4.1 as a *benchmark*, making several statistical assumptions later in the document. In particular, we will develop a non-linear programming example based on objective function 4.1 in chapter six.

We know that in traditional traffic signal control approaches, the heaviest travelled routes are either explicitly or implicitly given priority in setting the traffic signals. So, when we select our tree of "most important" links, **S**, why not just start with the busiest route and fill out the rest of the tree in descending order of use? This approach leads directly to the route augmentation heuristic for selecting the tree.

SECTION 4.2 Route Augmentation (RA) Heuristic

It seems clear that one should pay the most attention to the most highly traveled routes when determining traffic signal settings. Using the notion of a spanning tree, "rooted" at i^*, to give orientation and storing the numeric off-sets in Θ, one can construct the off-sets using a greedy heuristic based solely on desired route usage. First, one must understand several definitions.

Define an *index vector*, φ s.t. $\lambda_{\varphi_i} > \lambda_{\varphi_{i-1}} \forall i \in \{1, \cdots, n_R\}$.

Thus, φ is the descending index for the elements of Λ. Also define a logical variable **L** which is a vector of logical variables, i.e. $L_i \in \{.true., .false.\} \forall i \in |Z|$.

4.2.1 Verbal Description of RA Heuristic

(a) Set the first node of the busiest route to have a zero off-set from the master clock. Make this the root node for **S**. Set **S**={}, i.e. we begin with no arcs in our proto-tree.

(b) Following this same route, add subsequent arcs to **S** if the arcs fulfill the following **two** conditions:

(i) The arc is not *already in S*.

(ii) The addition of the arc to **S** does not create a cycle, i.e. the heuristic is building a *tree*. Thereby, one cannot add arcs which create a cycle.

(c) Set subsequent nodes' off-sets such that a vehicle traveling along this route experiences no delays due to traffic signal control *unless* the node's off-set is already been set, in which case we just accept the current setting.

(d) Continue with (b) *unless*:

(i) The current node is the last node in the route. In this case, we proceed to the next route as indexed by φ.

(aa) If the next route does not intersect **S** (as currently constructed) then set the off-set of the first node of this next route to zero and proceed with step (b) above.

(bb) If the next route *does* intersect the current **S** then we must divide our procedure into two steps.

(bb-1) Follow the current route from its origin node, i.e. $X_{\varphi_j}(0)$ to the first node which intersects **S**. Specifically, find

$$j = \underbrace{\arg\min}_{j\in\{1,\cdots,n_i\}}[X_{\varphi_i}(j) \in S]$$

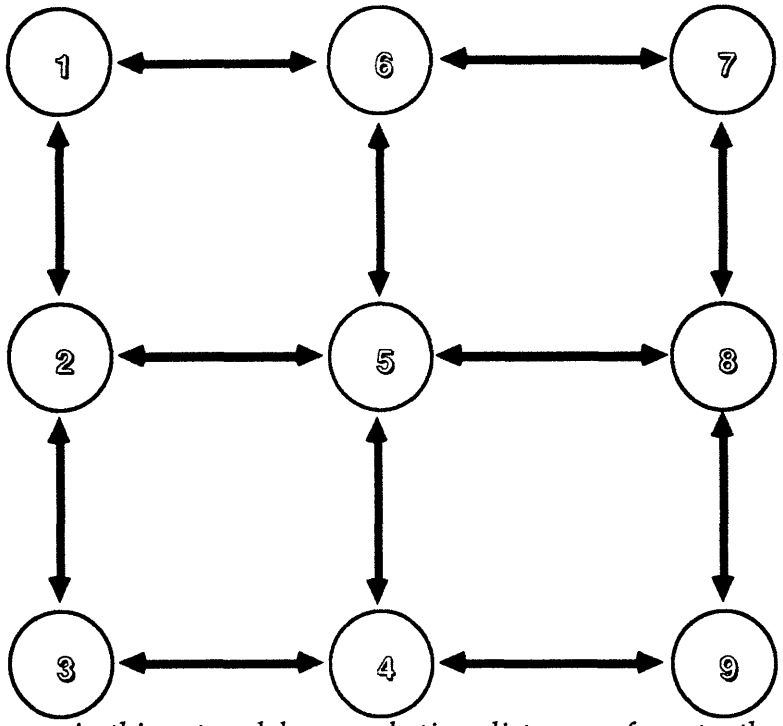
. Then set the off-sets tracing **S** from $\mathbf{i}^*$ to each node $X_{\varphi_i}(j-1)$ to $X_{\varphi_i}(0)$.

(bb-2) Now, beginning with the intersecting node, i.e. $X_{\varphi_i}(j)$, continue with step (b) above.

(ii) We have already selected (n-1) arcs for **S**. In this case, stop, the off-set setting routine is complete.

4.2.2 Example of RA Heuristic

We will use the same example for each of the off-set fixing heuristics. The base network is as shown below in fig. 4.2.



All the arcs in this network have cycle time distances of one tenth.

The sample network shown above is used to demonstrate the route augmentation heuristic.

Figure 4.2

The routes for the network are displayed in the table shown below.

Route	Desired Usage Rate
7->8->9->4->3->2->1	2.5
6->5->4	3
7->6->1	2.3
9->4->5->6	2.4
2->5->6	2

The table above shows the desired usage rate (vehicles/time) for the network shown in figure 4.2.

Table 4.1

Using the RA heuristic, the arcs for S would be selected in the following order; note, the value displayed in the table refers to the maximum desired usage

rate for any route containing the arc. The method selects arcs for the tree beginning with the first arc for the most traveled route and proceeding until the tree is complete. The value depicted in table 4.2 represents the desired usage rate for the most heavily travelled route for the arc shown.

Arc (In order of selection)	Value
6->5	3
5->4	3
7->8	2.5
8->9	2.5
9->4	2.5
4->3	2.5
3->2	2.5
2->1	2.5

The table shows the order the arcs were selected in using the route augmentation heuristic.

Table 4.2

This is the tree derived using the RA heuristic. To set the off-sets, one arbitrarily sets a direction into a particular node as the root or baseline and then defines all of the other off-sets in terms of this one by *climbing* the tree. Say for example that node 6 is selected ast the base node in this case in direction (5->6). Then the off-set for (6->5) would be .1, and the off-set for (5->4) would be .2. If the arc in **S** goes in the same direction its cycle time distance is added to the total; if the arc is reverse to the direction of travel, the cycle time distance is subtracted.

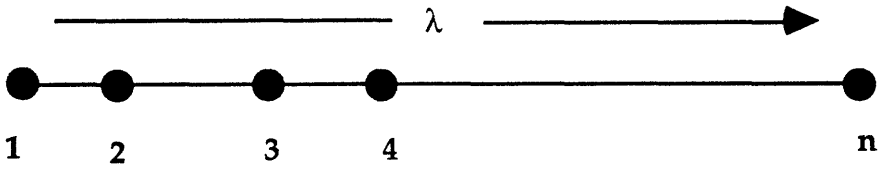
4.2.3 Theoretical Results for RA Heuristic

Route augmentation works very well in some circumstances and very poorly in others as we see in the following theorem.

Theorem 4.2.1: *For any positive set of* α, β, γ *for objective function 4.1, one can select* **G(N,A), R** *and* Λ *such that the route augmentation (RA) heuristics' solution can range from the optimal solution to the worst possible solution.for* $F(\bullet)$ *(as measured in terms of the M.O.E.'s defined earlier).*

Proof: Theorem 4.2.1 is an existence theorem; we will be constructing the simplest **G(N,A)** that meets the criteria presented in the statement of the theorem above. There are two bounds presented in the theorem above, an upper and a lower bound. The upper bound or the optimal solution case can be shown using a construction argument. Consider the diagram shown below. **G(N,A)** consists of **(n-1)** two way street segments (by convention, line segments without arrows represent two way streets) or equivalently **2(n-1)** one-way street segments and **n** intersections; although **n** itself will be allowed to increase below, the network always retains the same form. Traffic is allowed to freely flow either direction on the arcs. Traffic can enter and leave **G(N,A)** from any of the **n** intersections, but by assumption 4.1.1(5), traffic would never change directions on **G(N,A)**. We will construct the network such that it takes a vehicle $C/2 + \delta$ time units to travel from one node to the next along its route where δ be positive but close to zero. Now, our claim is somewhat counter-intuitive, namely that the RA heuristic can give offset settings which can range from producing the optimal value for

objective function 4.1 to producing settings which are arbitrarily worse (negative) than the optimal possible value for $F(\bullet)$, α.

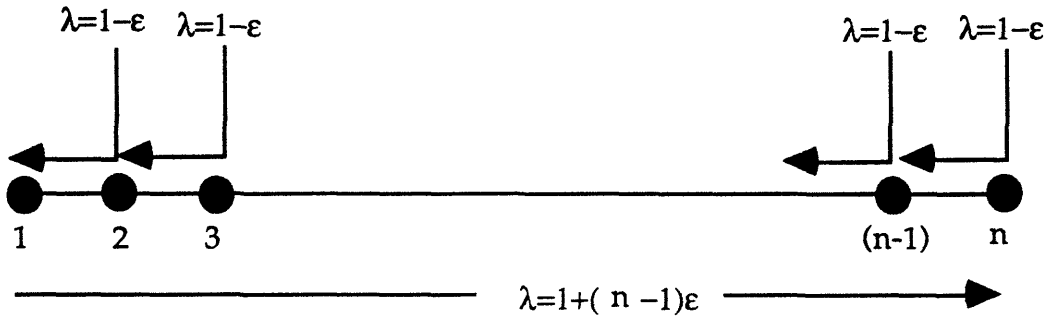


There are no conflicting routes; so, the route augmentation heuristic provides the best possible theoretical expected number of stops and wait per vehicle.

Figure 4.3

In fig. 4.3, there is only one route with non-zero desired usage. Thus, using the RA heuristic the route will be completely synchronized in the sole direction of travel. One achieves a progression of one, and average stops per vehicle and average delay per vehicle are equal to zero, *no matter how large* ***n*** *is*. This is the optimal performance for our M.O.E.'s, and $F(\bullet)$ gives back a value of α.

The worst case performance is a bit more complex. Consider the diagram shown below.



The route augmentation heuristic will set the offsets to favor the route from node 1 to node "n", but this will disadvantage the majority of the traffic flow which travels in the opposite direction.

Figure 4.4

It is the same as figure 4.3 except now there are **n** routes in fig. 4.4, and all of the routes have a desired usage of 1-ε except for the contra-flow route which has a desired usage of $\lambda_{1\to 2\cdots\to n} = 1 + (n-1)\varepsilon$. By the route augmentation heuristic, the network would be synchronized for the route $1 \to 2 \to 3 \cdots \to n$ with a total desired usage of $\lambda_{1\to 2\cdots\to n} = 1 + (n-1)\varepsilon$. Now, the total flow in the network is $\sum \lambda = n$. Using assumption 4.1.1(8) and the properties of the RA heuristic, we know that each of the **(n-1)** retrograde routes will experience a stop at each intersection in **G(N,A)** of duration $C/2$ time units. When ε is positive, but close to zero, objective function 4.1 returns the following value:

$$\alpha\left(\frac{1+(n-1)\varepsilon}{n}\right)-\beta\sum_{i=1}^{n-1}\frac{(1-\varepsilon)i}{n}-\gamma\sum_{i=1}^{n-1}\frac{(1-\varepsilon)i\,C/2}{n}=$$

$$\alpha\left(\frac{1+(n-1)\varepsilon}{n}\right)-(1-\varepsilon)\left[\frac{\beta}{n}\frac{n(n-1)}{2}+\frac{\gamma}{n}\frac{Cn(n-1)}{4}\right]=$$

$$\alpha\left(\frac{1+(n-1)\varepsilon}{n}\right)-(1-\varepsilon)(n-1)\left[\frac{2\beta+\gamma C}{4}\right].$$

This is a monotonically decreasing function with respect to **n**. So, no matter how negative a value one initially selects, we can always find a **G(N,A)** and Λ such that objective function 4.1 produces an answer less than or equal to the selected value. We could explicitly select an **n** which would produce the desired level of "disfunction". Even for this **G(N,A)** we can modify the flows by making ε arbitrarily close to one, then in the limit we have the situation illustrated in fig. 4.3 and **F(•)** returns a value of α, the best possible solution. Q.E.D.

One can see that the *aggregate* flows must somehow be considered when choosing the spanning tree for setting the off-sets between traffic signals in **G(N,A)**. This conclusion leads us directly to the next heuristic, the Maximal Spanning Tree (MST).

SECTION 4.3 Maximal Spanning Tree (MST) Heuristic

The central concept of the MST heuristic is that the most critical sets of arcs in **G(N,A)** to coordinate are the those with the highest *aggregate* desired usage. This concept immediately abrogates the constructed example which made the route augmentation method perform so poorly.

4.3.1 Verbal Description of MST Heuristic

There are two levels of information considered in the MST heuristic. In the first case, one knows a'priori the routes of all the vehicles entering **G(N,A)** from the present until some unspecified time in the future. (This corresponds to the notion of route planning under the literature for IHVS.) The other level of information is to assume one can accurately measure *current flow rates* along each of the street segments in **G(N,A)** and *approximate* desired usage based entirely on observed data. This corresponds to the level of information many cities currently accumulate using sensors (e.g. induction loop sensors which can measure occupancy and velocity).

(a) In the first case, where one knows the actual routes a'priori, let $\hat{\lambda}_{ij}$ be the aggregate of the desired usage rates of all routes traversing $(\mathbf{i} \rightarrow \mathbf{j})$.

In the second case, let $\hat{\lambda}_{ij}$ be the observed flow rate along $(i \rightarrow j)$ over some fixed time period, i.e. $\hat{\lambda}_{ij} \triangleq \lambda_{ij}(\emptyset, \hat{t})$ where $\hat{t}$ is known..

(b) Designate the MST, **S**, where

$$S \triangleq \max_{S \text{ s.t. } S \text{ is a tree}} [\sum_{(i \rightarrow j) \in S} \hat{\lambda}_{ij}] \quad . \qquad [4.2]$$

Define a "branch count index" for **S** called **c** such that

$$c_i = \sum_{\substack{(i \rightarrow j) \in S \\ \forall j \in \{1, \cdots, n\}}} \langle 1 \rangle \quad . \qquad [4.3]$$

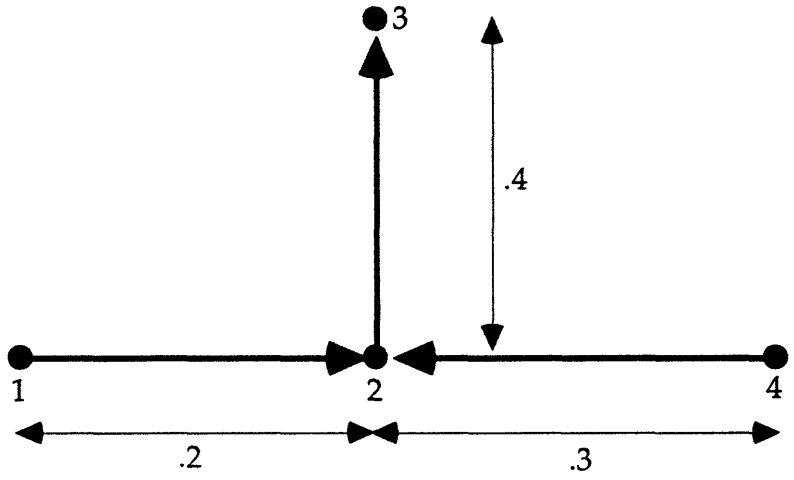
This is a count of how many other nodes in the tree are connected to node **i**.

Note that all leaf nodes in **S** have c_i equal to one.

(c) Select an arbitrary root node i^* s.t. $c_{i^*} = 1$. Let $\theta_{i^*} = \emptyset$.

(d) Set the off-sets for θ_{ij} such that $\forall (i \rightarrow j) \in S$, the MST, a vehicle traveling from node "k" through node "i" to node "j" experiences no delay if $\psi[(k \rightarrow i), (j \rightarrow i)] = .\text{true}$. We can do this, because there is a unique

path from i^* to any other node in S, and by definition, all nodes are part of the MST. An important point is that the arcs are directed. Additionally, one must take into account the geometry of the physical network. Consider the following example.



The figure is used to demonstrate how the MST heuristic sets offsets (see below).

Figure 4.5

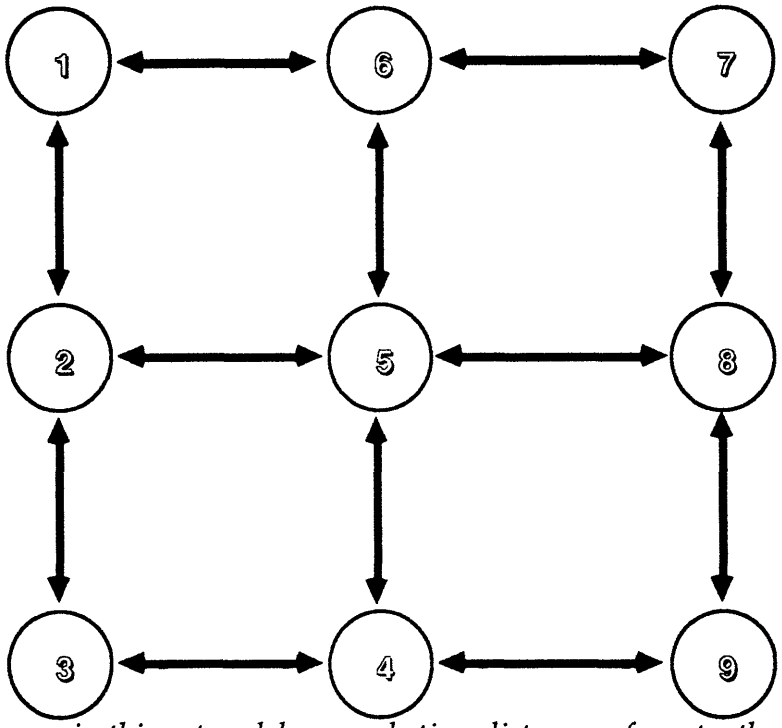
Assume that all of the arcs shown are in S, the MST. Let $i^*=1$. Let $\theta_1=0$. Now, the MST heuristic assigns:

$\theta_{12} = .2$, $\theta_{23} = .6$, $\theta_{42} = \theta_{12}$ [by alignment], and $\theta_{42} = .9$.

This assumes that $\hat{t}_{42} = \hat{t}_{24}$.

4.3.2 Example of MST Heuristic

We will use the same example for each of the off-set fixing heuristics. The base network is as shown below.



All the arcs in this network have cycle time distances of one tenth.

The sample network shown above is used to demonstrate the route augmentation heuristic.

[*Same as* Figure 4.2]

The routes for the network are displayed in the table shown below.

Route	Desired Usage Rate
7->8->9->4->3->2->1	2.5
6->5->4	3
7->6->1	2.3
9->4->5->6	2.4
2->5->6	2

The table above shows the desired usage rate (vehicles/time) for the network shown in figure 4.2.

[*Same as* Table 4.1]

Using the MST heuristic, the arcs for S would be selected in the following order; note, the value displayed in the table refers to the aggregate desired usage rate for the arc across all routes using that arc. For example, arc (9->4) has an anticipated usage of 4.9 vehicles per minute which is the sum of all the predicted routes usages for that arc. In this case, the route from 9 to 6 and the route from 2 to 6. The value shown in table 4.3 is the aggregate desired usage of the arc shown.

Arc (In order of selection)	Value
9->4	4.9
5->6	4.4
5->4	3
7->8	2.5
8->9	2.5
4->3	2.5
3->2	2.5
2->1	2.5

The table shows the order the arcs were selected in using the MST heuristic.

Table 4.3

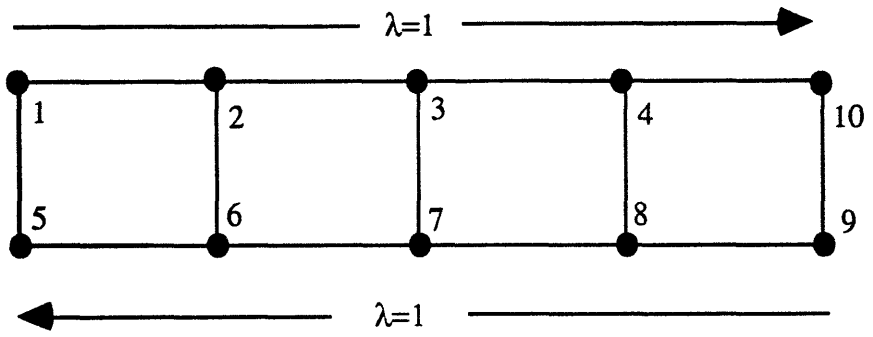
This is the tree derived using the MST heuristic. To set the off-sets, one arbitrarily sets a direction into a particular node as the root or baseline and then defines all of the other off-sets in terms of this one by *climbing* the tree. Say for example that node 6 is selected as the base node in this case in direction (5->6). Then the off-set for (6->5) would be .9, and the off-set for (5->4) would be zero. If the arc in **S** goes in the same direction its cycle time distance is added to the total; if the arc is reverse to the direction of travel, the cycle time distance is subtracted.

4.3.3 Theoretical Results for MST Heuristic

The MST Heuristic is more flexible than the route augmentation heuristic. In situations where there are several non-intersecting routes which make up nearly all of the flow in **G(N,A)** (e.g. artery during commuter cycle, etc.), the MST solution performs near optimality.

Theorem 4.3.1: *The upper bound on Objective Function 4.1 for the MST Heuristic is the global optimal value.*

Proof: [By construction] Consider the diagram shown below.

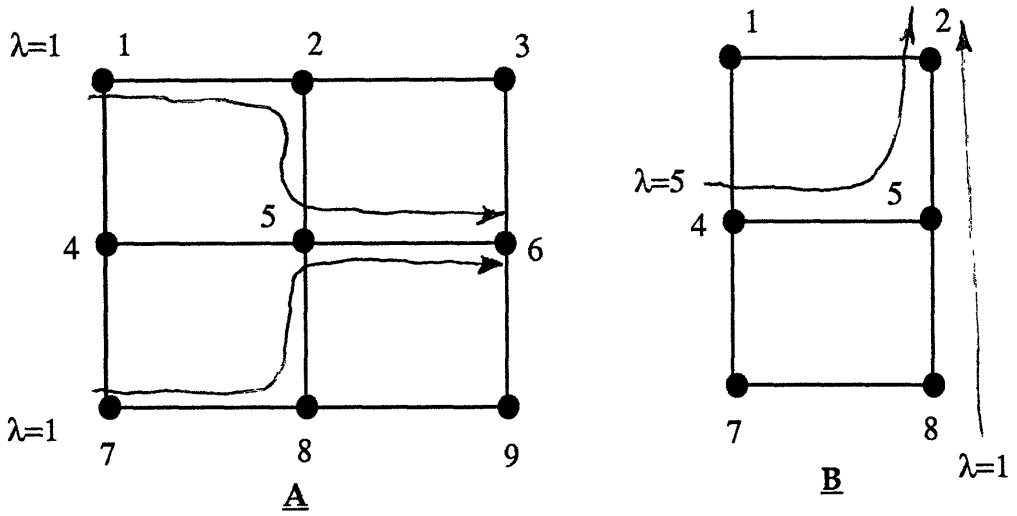


In this diagram, the two routes do not cross each other; so, both can be synchronized.

Figure 4.6

In this case, the routes do not intersect, and the MST will include both routes and one "connecting arc (e.g. $(2 \rightarrow 6), (7 \rightarrow 3)$, etc.). The MST heuristic will give off-sets which produce the optimal M.O.E.'s. Note, one receives optimal (θ, S) settings from the MST algorithm whenever there is no overlapping of routes. Q.E.D.

Additionally, the MST heuristic produces the optimal (θ, S) for Objective Function 4.1 whenever there is no orthogonal or retrograde flow vice the elements of the MST. Consider the two examples displayed below.



In the network at the left, both routes can be synchronized, but in the figure at the right, one of the two routes will experience an alignment delay. The network cannot synchronize both for arc (5->2), because the two routes do not share a common green cycle at node 5.

Figure 4.7

In figure 4.7.A, all the flow in **G(N,A)** occurs along the MST, and although there are overlapping flows, θ_{56} will be set to accommodate the aligned flows from 8 to 5 and 2 to 5. In figure 4.7.B in contrast, there is both orthogonal and retrograde flow. One could devise nominal velocities and distances between intersections which cause sever delays along the route going from 2 to 5 to 4. Additionally, θ_{52} will be synchronized to accommodate flow from 4 to 5 to 2, which means that traffic flowing from 8 to 5 to 2 will be delayed for C/2 units at node 2. We will call such a delay an "alignment delay"; this is also important for a vehicle turning onto the MST for the first time. This alignment delay is one of the chief reasons for changing the splits from a 50/50 arrangement. [Ed. - foreshadowing]

We have looked at the MST heuristic's upper bound of effectiveness, but what about its lower bound of effectiveness. A very important question is how many arcs are covered by the MST in a worst case scenario?

To answer this let us define "g" to be the number of the "generation" of nodes in a g*g network. For example, a single node (i.e. g=1) has 1 node and 0 arcs. If there are g^2 nodes then there are a total of 4g(g-1) arcs (each one-way) possible in the network, assuming that each node in the center of **G(N,A)** is connected to its four closest neighbors. Likewise, the MST will contain g^2-1 arcs. The results for the first four generations are contained in the table below.

g	node	arcs	arcs in MST	fraction
1	1	0	0	NA
2	4	8	3	3/8
3	9	24	8	1/3
4	16	48	15	15/48

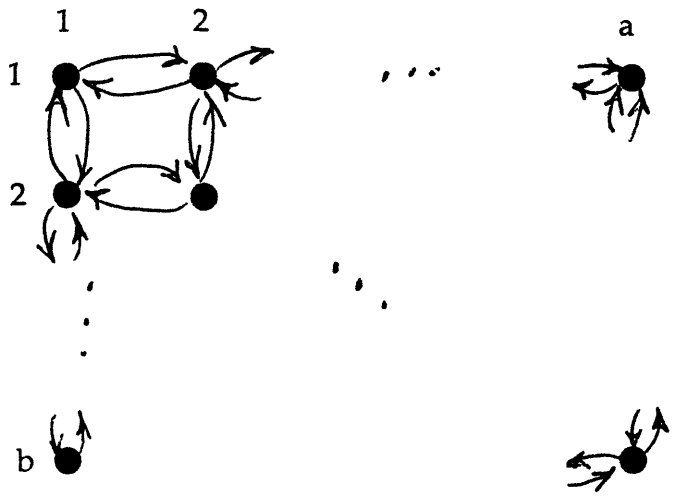
Table displays the relationships between arcs, nodes, and fraction of arcs in the spanning tree for a square network.

Table 4.4

The results in the table immediately motivate the next theorem.

Theorem 4.3.2: *The minimum fraction of arcs in the MST is one fourth.*

Proof: Consider the following network with **ab** nodes. There are "a" columns and "b" rows. Now, connect each node with its closest orthogonal neighbors using two one-way arcs. Then we have the following diagram.



The figure shows a rectangular a*b network.

Figure 4.8

Note this diagram is a network approximation of traffic flow in an urban grid where the maximum number of non-aligned intersecting streets equals two. In the grid described in figure 4.7, there are **ab** nodes; ergo, there are **ab-1** nodes in the MST. Consider the top of the graph and the extreme left hand side. Between these two portions there are a total of **2(a+b-2)** one-way arcs. There are a total of **4(a-1)(b-1)** one-way arcs in the remainder of the graph.

$\therefore$ The fraction of arcs included in the MST is:

$$\frac{[ab - 1]}{[4ab - 2a - 2b]}.$$

For $\forall\ (0 < a, b < \infty)$ the ratio is greater than $\frac{1}{4}$.

But,

$$\lim_{a,b \to \infty} \inf \left(\frac{[ab - 1]}{[4ab - 2a - 2b]} \right) = \frac{1}{4}.$$ Q.E.D.

One can see this intuitively if he considers the case of large but finite values of **a** and **b**. For the value of the fraction to equal or exceed one fourth, the denominator must be one fourth of the denominator. Thereby, the denominator must equal or exceed **4[ab-1] = 4ab - 4**. But, **(2a+2b)** will exceed **4** for any positive numbers **a** and **b**. Since the denominator is less than four times the numerator, the whole fraction must be uniformly greater than one fourth in the case of finite **a** and **b**. Before we can explore this more deeply, there is a simple proposition we must prove relating to the way we look at the cumulative arc flows when selecting the MST.

Theorem 4.3.3: *Examine any two adjectent nodes, and label them* i *and* j *respectively.If either* $(i \to j)$ *or* $(j \to i)$ *is an element of the MST then only one of the arcs will be an element and, that arc will be the* $\underset{\text{arc}}{arg\ max}\ (\hat{\lambda}_{(i \to j)}, \hat{\lambda}_{(j \to i)})$.

Proof: Both arcs cannot be a member of the MST, because this would form a cycle. Assume that the arc which is the member of the MST does **not** fulfill the arg max condition. Call the MST **S**. Arbitrarily assume that

$(j \rightarrow i) = \underset{arc}{arg\ max} (\hat{\lambda}_{(i \rightarrow j)}, \hat{\lambda}_{(j \rightarrow i)})$. So, $(i \rightarrow j) \in S$. Let $\hat{S} = S - \{(i \rightarrow j)\}$. Now, $\hat{S} + \{(j \rightarrow i)\}$ is a tree; moreover,

$\sum_{(k \rightarrow l) \in \hat{S}} \hat{\lambda}_{kl} + \hat{\lambda}_{ij} < \sum_{(k \rightarrow l) \in \hat{S}} \hat{\lambda}_{kl} + \hat{\lambda}_{ji}$. Therefore, S is not the MST which is a contradiction, and the origional premise is true. Q.E.D.

This is an interesting insight into the traffic control problem. If we focus on the direction with the majority of traffic, we can restrict our process to look at the direction with the most traffic flow on each two-way street segment. This is stated in the following lemma.

Lemma 4.3.3: *The MST Heuristic need only examine the maximum of each set of two way arcs for each set of nodes to determine the MST.*

Proof: The proof flows directly from the theorem above.

This lemma will prove very useful when considering the total flow examined by the MST heuristic. We know that the MST heuristic will always have at least half of the flow on the network available *to choose from,* because the heuristic will always choose the arc with the largest flow when selecting how to connect two adjacent nodes. Let us construct a method for selecting the MST to help us come up with a lower bound.

Procedure 4.3.1: Pick the arc with the largest flow between every set of connected nodes. Now, rank order all the arcs remaining beginning with the highest flow rate and ending with the lowest flow rate. Select arcs for the MST according to the following rule. Add arcs beginning with the highest flow rate. Add an arc only if it does not create a cycle. If it does create a cycle, skip that particular arc and continue to the next arc in the ordered list. Continue until you have chosen **(n-1)** arcs.

The procedure is simple $O(n^3)$, depending on one's data structures for the routine, but powerful. In fact, the procedure produces the MST. Even for a relatively complex urban grid; let us say 200 to 300 nodes, a current technology microprocessor could calculate the MST in a few seconds. Better yet, the procedure is assured of producing the MST.

Theorem 4.3.4: *Procedure 4.3.1 produces the MST.*

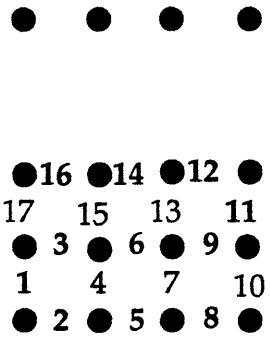
Proof: We know according to lemma 4.3.3 that we must only consider the arc with the highest flow rate between any two connected nodes. Now assume that the greedy procedure does **not** produce the MST. We know that the MST will contain exactly **(n-1)** arcs. Call the output from procedure 4.3.1 **T**, and call the true MST **S**. There are two cases here. Either **S** and **T** vary by only two arcs, or there is a sequence of trees between these two trees, each varying from the next by only two arcs. If the **S** and **T** vary by only two arcs then we can list the arcs in descending sequence of flow rate in each tree. Select the first arc in **T** which

differs from the arc list for S. Now the arc in T must be replaced by an arc with a lower flow rate, because procedure 4.3.1 assures us that a cycle will result if the arc is replaced by an arc with higher flow. In fact, it must be replaced by an arc *with a flow rate less than the second arc being replaced*, because there are no arcs which can be selected above the second arc which will not cause a cycle either. But, this is clearly impossible as if both of the arcs replaced by arcs of lower flow rate, the total flow rate for the new structure cannot be greater, and we have a contradiction for this case. In the case where there are a sequence separating the two trees, repeat the same line of reasoning first for one separating tree, showing the contradiction, and then use induction to show the result holds for an arbitrarily large number of intervening trees. Q.E.D.

But, we need to be a little cautious at this juncture. We are still not sure what fraction of the total flow will be encompassed by the MST. We *do know* that at least half of the total flow is in candidate arcs for the MST based on lemma 4.3.3. What is the lower limit of flow which can be included in the structure? An intermediate question is when one builds the MST how many arcs does one have to examine before selecting the MST? This question is addressed in the following theorem.

Theorem 4.3.5: *If procdure 4.3.1 was applied to the network shown in figure 4.8 it would examine a maximum of* $ab-1+(a-1)(b-1)\ \ 1=2ab-a-b-1$ *arcs before finding the MST.*

Proof: This occurs in the case shown below in figure 4.9.



The numbers on the arcs represent the order in which they were selected to be part of the MST. The arcs in the bold print are part of the MST; the arcs in the light print were rejected, because they would have created a cycle.

Figure 4.9

Figure 4.9 shows the psychotic case where the flows are ordered in such a way that, for the median case, every new arc selected implies that the next arc in the ordered list will produce a cycle. Lemma 4.3.3 implies that the procdure need only consider **2ab-a-b** arcs. Remember, each arc can only be examined once, because it only occupies one slot in the rank ordering. Thereby, the procedure will examine a total of **ab-1** arcs to complete the MST plus **(a-1)(b-1)-1** verticle arcs causing cycles for a total of

2ab-a-b-1. Q.E.D.

Theorem 4.3.6: *Procedure 4.3.1 selects the smallest proportion of total flow for the MST in the case where all flows are equal.*

Proof: In the case of equal flows, the MST will select a minimum of one fourth of all the flow in the network. This follows directly from theorem 4.3.2, because we could simply choose an arbitrary flow $\hat{\lambda}_{kl}$ (since all the flows are equal) and write the following relation:

$$\textbf{flow ratio in MST} \triangleq \lim_{\mathbf{a},\mathbf{b} \to \infty} \inf \left(\frac{[\mathbf{ab} - 1]\hat{\lambda}_{kl}}{[4\mathbf{ab} - 2\mathbf{a} - 2\mathbf{b}]\hat{\lambda}_{kl}} \right)$$ which has the value of one fourth. Now, consider the case where the flows are not equal. We know that procedure 4.3.1 will select the MST from the best **4ab-2a-2b-2** arcs (i.e. **2ab-a-b-1** times two for lemma 4.3.3). Since you are not considering the last two arcs, you will end up with a greater total than the amount if one considered all the arcs; thereby the ratio of total flow in the ordered case must exceed the ratio of total flow in the case of total equivalence.

Thus, the MST heuristic is logical and has some nice theoretical results. But, it would be nice if we could minimize the alignment delays by explicitly trying to favor *routes* as well as considering the total projected usage of the arcs. The concept of including routings as well as total usage leads us to the potential function heuristic which somewhat surprisingly is just an extension of the RA and MST heuristics.

SECTION 4.4 Potential Function (PF) Heuristic

The alignment delay caused theoretical problems for the MST heuristic. Logically then, one must consider not only the aggregate flow, but one must form the offset tree, **S**, incorporating information on contiguous numbers of arcs which support the same routes as well, i.e. we would rather favor longer routes if all other factors are equal. This leads quite naturally to the idea of a potential function, somewhat akin to the notion of potential energy in physics. *We will assume that there is some function, related to the routes' and network's attributes, which gives a good approximation of what the impact would be of including any particular arc in the tree.*

In mathematical terms, one would say

Assume $\exists$ a function $F[(i \rightarrow j), G(N,A), \Lambda, S]$ s.t. $F[(i \rightarrow j),\ldots] > F[(k \rightarrow l),\ldots]$
$\Rightarrow (\Theta, S + (i \rightarrow j)) > (\Theta, S + (k \rightarrow l))$ where $S \subset A$, S contains no cycles and $\|S\| < (n-1)$.

What does this really say? We assume that there is some function that will give "scores" to all the remaining candidate arcs for **S**; arcs which are already part of **S** are ineligible. The better the score, the better a choice the arc is for inclusion in **S**, but we still would not choose an arc with a very high score if the choice would cause us to form a cycle in S.

This is the notion (or notation) of a potential function. In words, the potential function "f" is a mapping from the network and a *proto-tree*, **S**, i.e. **S** contains no cycles and $|S| < (n-1)$, and the arc being examined to the set of real numbers. Additionally, the function assigns a value to this set in such a way as to establish a hierchy for selecting the arcs to make up the spanning tree to set the off-sets. Specifically, if an arc (i->j) has a higher value when evaluated by the potential function than arc (l->k) then the math program 4.1 using **S** plus arc (i->j) will have an equal or higher value compared to the same program using **S** plus arc (l->k).

The PF seems like a radically new concept when compared to the MST and RA heuristics, but it actually just an extension of the previous two heuristics. *(Those who are less mathmatically inclined may wish to fasten their math safety belts before reading the remainder of this notationally "turbulent" section.)* Consider the following two definitions of potential functions. Let

$$f(l \to k) = \sum_{i=1}^{\|R_i\|} \lambda_i I_i (l \to k) \text{ where}$$

$$I_i (l \to k) = \begin{cases} 1 & (l \to k) \in R_i \\ 0 & \textbf{else} \ldots \end{cases}$$

Note, that the value of the potential function does not depend on **S**, and using this definition of the potential function gives one the MST heuristic. Why? The indexing function, I_i, gives us a one if the arc is part of the route in question and a zero otherwise. So, for each arc the potential just gives the sum of the directed flows through the arc. Now consider,

$$f(l \to k) = \sum_{i=1}^{\|R_i\|} \lambda_i I_i (l \to k) \text{ where}$$

$$I_i (l \to k) = \left\lfloor \frac{\lambda_i}{\lambda_{\hat{i}}} \right\rfloor \text{ where } \hat{i} \hat{=} \underbrace{\arg \max}_{\forall j \text{ s.t. } (l \to k) \in R_j} (\lambda_j)$$

This definition of the potential function approximates the RA heuristic.

SECTION 4.4.1 Verbal Description of the PF Heuristic

Initially we will select the arc with the greatest flow on it to start **S** out. Then we will add arcs to S in order of potential function except when such an addition would create a cycle. In this case, we select the arc with the next highest potential function value. We continue the process until we have selected **(n-1)** arcs, i.e. our tree is complete. Note that because the potential function is an implicit function of S, we must recalculate the potential function values each time we select a new arc. In more mathematical terms, we define the potential function as shown below.

$$f(l \to k) = \sum_{i=1}^{\|R_i\|} \lambda_i I_i (l \to k) \text{ where}$$

(a) Set S={}. Let $I_i (l \to k) = \begin{cases} 1 & (l \to k) \in R_i \\ 0 & \text{else . . .} \end{cases}$.

(b) If $|\mathbf{S}|<(\mathbf{n-1})$ then select the arc with the highest PF value, i.e. (i->j)=arg max f(l->k)$\forall$(l->k), that is not contained in **S** and does not form a cycle when added to **S** .

Else goto (d)

(c) Add (i->j) to **S** and recalculate f(l->k) using

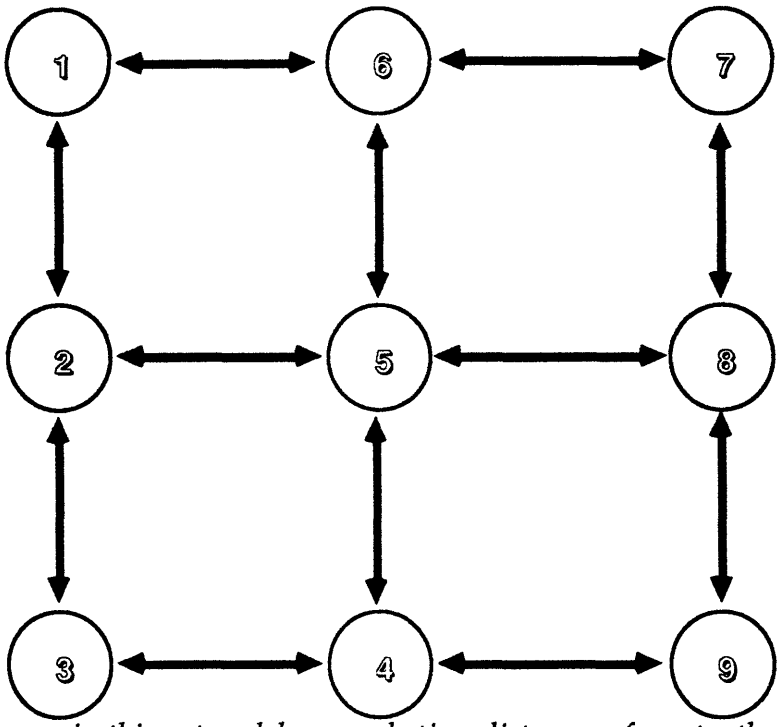
$$\mathbf{I_i\,(l \rightarrow k)} = \left\{ \begin{array}{c} \text{Number of contiguous arcs of route i, including } l \rightarrow k \\ 0 \text{ if } (l \rightarrow k) \notin R_i \end{array} \right\}.$$

Goto (b)

(d) The tree is complete. Stop!

SECTION 4.4.2 Example of PF Heuristic

We will use the same example for each of the off-set fixing heuristics. The base network is as shown below.



All the arcs in this network have cycle time distances of one tenth.

The sample network shown above is used to demonstrate the potential function heuristic.

[*Same as* Figure 4.2]

The routes for the network are displayed in the table shown below.

Route	Desired Usage Rate
7->8->9->4->3->2->1	2.5
6->5->4	3
7->6->1	2.3
9->4->5->6	2.4
2->5->6	2

The table above shows the desired usage rate (vehicles/time) for the network shown in figure 4.2.

[*Same as* Table 4.1]

The PF selects arcs based on an arc's "potential". An arc's potential is a function of the arc's projected usage and the number of contiguous arcs for each route passing through the arc if the arc in question is added. The arcs are selected in

the order shown below in table 2. The best way to explain the method is to step the reader through the selection process for the tree.

Arc (9->4) is the first arc selected it has a potential of 4.9 which is the highest potential value across all arcs in the network when S={}. Where did the value 4.9 come from? Well, two routes pass through (9->4). By adding (9->4), we would be making the total number of contiguous arcs equal to 1 for each route. Thereby, we have:

potential for arc (9->4)= 1*2.5 + 1*2.4=4.9 .

Arc (3->2) has a potential of 5 when S={(9->4)}. Why? There is only one route which passes through arc (3->2), route 7->8->9->4->3->2->1 which has a projected usage of 2.5 vehicles per minute. By adding arc (3->2) to the tree, S, it would give this route 2 contiguous arcs, and we have:

potential for arc (3->2)= 2*2.5=5 . [Note, the next highest competitor was arc (4->5) which had a potential of 4.8 .]

Using the PF heuristic, the arcs for S would be selected in the following order; note, the value displayed in the table refers to the potential function value calculated for the arc at the time it was selected. Remember that the value of the potential function depends on the arcs which have already been chosen. The value shown in table 4.5 is calculated as shown in the proceeding paragraph.

Arc (In order of selection)	Value
9->4	4.9
4->3	5
3->2	7.5
2->1	10
8->9	12.5
7->8	15
4->5	4.8
5->6	9.2

The table shows the order the arcs were selected in using the potential function heuristic.

Table 4.5

This is the tree derived using the PF heuristic. Note that arcs do not have constant PF values; for example, arc (4->3) started off with a value of 2.5 and had a value of 5 when it ws chosen to be part of **S**. The off-sets are calculated using the same technique as in RA and MST once the tree **S** is chosen.

SECTION 4.4.3 Theoretical Results for the PF Heuristic

Theorem 4.4.1: *The upper bound on Objective Function 4.1 for the PF Heuristic are the global optimal M.O.E.'s.*

Proof: The same proof applies from theorem 4.3.1. Q.E.D.

Theorem 4.4.2: *The lower bound on the M.O.E.'s for the PF Heuristic are the lower bounds of the MST Heuristic.*

Proof: In the worst case, one could construct an entire network composed of routes which only comprise one or two arcs in length. This would prevent the PF heuristic from achieving any of its advantages which are accrued for longer routes. Q.E.D.

In this chapter, we examined ways of selecting the most significant street segments to align. In the next chapter, we will examine various methods of predicting the M.O.E.'s based on the assigned splits and offsets.

Chapter Five.

Predicting Performance for TSC Settings

SECTION 5.1 Light Traffic Approximations

We have examined the basic math model for Traffic Signal Control (TSC) in chapter three and looked at three heuristics for determining the tree for setting the offsets in chapter four. Before we decide how to set the splits and cycle time for the network, it would nice to be able to make predictions about the performance of our TSC plan. Although we have only determined the offsets, we can make some approximations with just this information, and later in the chapter, we will loosen some of our restrictions from section 4.1 and develop a Queue Effects Model (QEM) which will be the basis of our split and cycle time setting heuristics in chapter six. Assume for the moment that we just completed one of the heuristics in chapter four.

We have a tree, **S**, that we use to define our offsets. By assumption 4.1.1(8), every split is set to **C/2**. A vehicle which starts on an arc in **S** and stays on arcs contained in **S** as it travels should not experience any stops or delays due to traffic signal control in the light traffic model. We remember from chapters three and four that the light traffic model assumes that the only reason that a vehicle will have to wait at a traffic light is because of a red light. If a vehicle arrives at an intersection during a green light, it will

pass through the intersection without delays, and even if the vehicle arrives during a red light, it will be able to pass through the intersection during the subsequent green light. But, in practice there will be vehicles which travel on **S** during only a portion of their trip through **G(N,A)**, and some vehicles may *never* travel along an arc in **S**. So, the question is how do we estimate the performance for the routes which contain at least some arcs *which do not belong* to **S**?

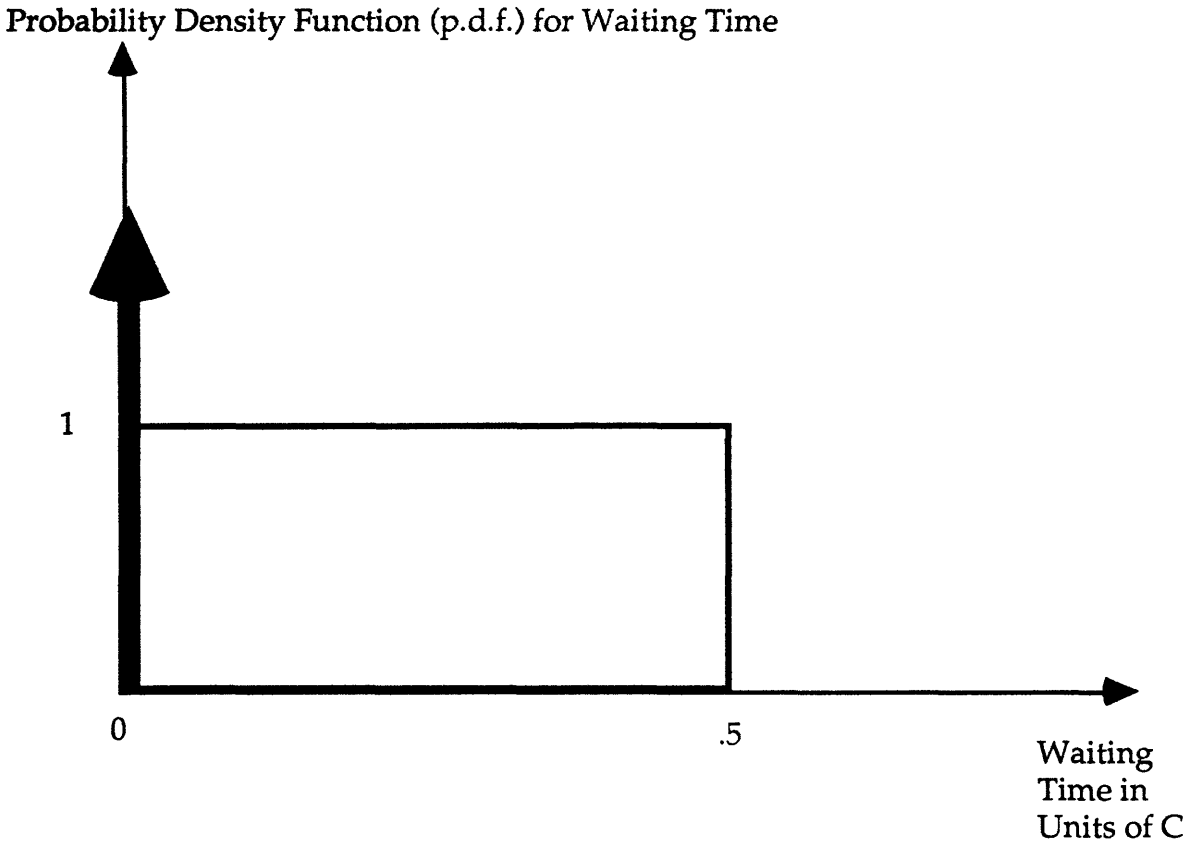
It would be convenient for analyzing large networks if one could view vehicles traveling along the arcs which are not part of **S** as having characteristics which were somehow both stochastic (retaining key random elements) yet were *independently and identically distributed*(which allows us to use some very powerful probability models). One can see that after the first red light, even vehicles which arrive according to a Poisson process to **G(N,A)** will be linked to other vehicles for number of stops, delays, etc., because the vehicles will tend to travel in platoons, even given our assumptions. This is not a bad thing though as vehicles tend to travel in platoons in the real world as well. To get a get an intuitive feel for some of the more complex results presented later, let us begin by examining a more straight forward case where the splits are equally divided between red and green time.

SECTION 5.1.1 Stochastic Approximations for Equal Splits

We will begin by assuming that the amount of red and green time is evenly distributed at each intersection, i.e. 50% red time and 50% green time. In other words, consider the case of this randomly chosen vehicle approaching an isolated traffic signal

obeying assumptions 4.1 through 4.7, and assume the *splits* between red and green portions of a traffic signal cycle at any particular intersection, facing any particular fixed direction are equal to $C/2$. Remember that the street segments which are not contained in **S** are *not coordinated*. We will make a slightly stronger assumption and assume that a vehicle's probability of stopping and the waiting time distribution on one non-coordinated street segment are independent of the same items on a different non-coordinated street segment. In other words, every time a vehicle approaches an intersection its chance of stopping or the amount of time it can expect to wait *does not depend on what has happened to the vehicle earlier in the network.*

Looking at an isolated intersection one would expect the chance of being stopped when reaching the intersection at a random time to be 1/2, and the waiting time distribution to be as shown below.



The diagram shows the probability density function for the waiting time distribution of a lone vehicle approaching an intersection with a cycle time of length C and red and green splits equal to C/2.

Figure 5.1

Thereby, if a vehicle passed through **u** uncoordinated arcs (i.e. arcs that are not part of **S**), the p.m.f. for number of stops experienced and the p.d.f. for the waiting time distribution would be as shown below.

Let **C** ≡ the cycle time for **G(N, A)**

Let **X** ≡ RV # of stops due to traffic signal control.

Let **w** ≡ RV waiting time due to traffic signal control.

Let **u** ≡ RV # of signal controlled intersections approached on an arc ∉ **S**

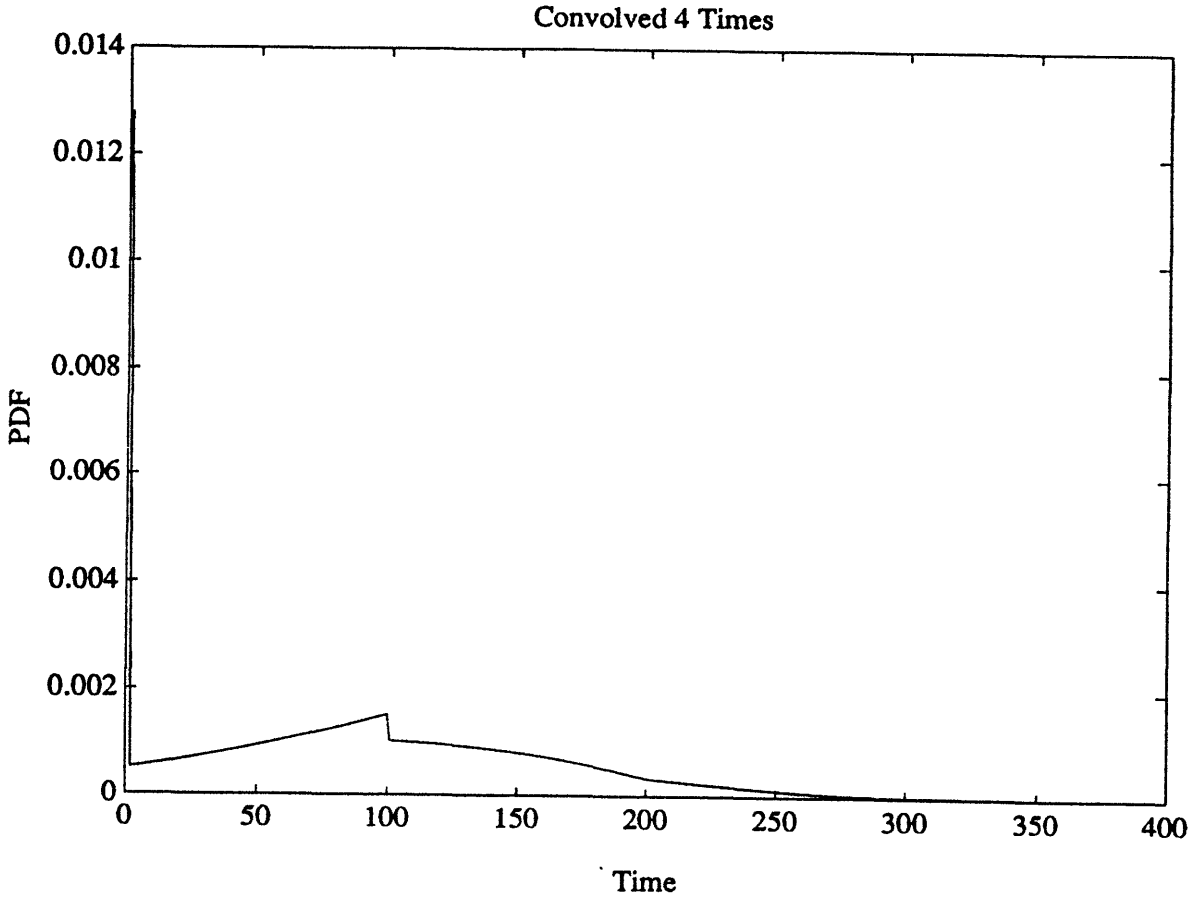
$$p(x_0) = \binom{u}{x_0}\left(\tfrac{1}{2}\right)^u \Big| \quad x_0 \in |Z| \text{ and } x_0 \le u.$$

$$p_x^T(z) = \left(\frac{1+z}{2}\right)^u.$$

Putting it all together, we have the s-transform of the waiting time distribution convolved **u** times.

$$f_w^T(s) = p_x^T\left(z = \tfrac{1}{2}\left[\frac{sc+1-e^{-sc/2}}{sc}\right]\right) = \left(\frac{3sc+1-e^{-sc/2}}{4sc}\right)^u.$$

It is easier to visualize the waiting time distribution if we examine a diagram of its appearance after several convolutions. Below, in figure 5.2, we observe the p.d.f. after being convolved four times. The cycle length for this example is 100 time units in duration.

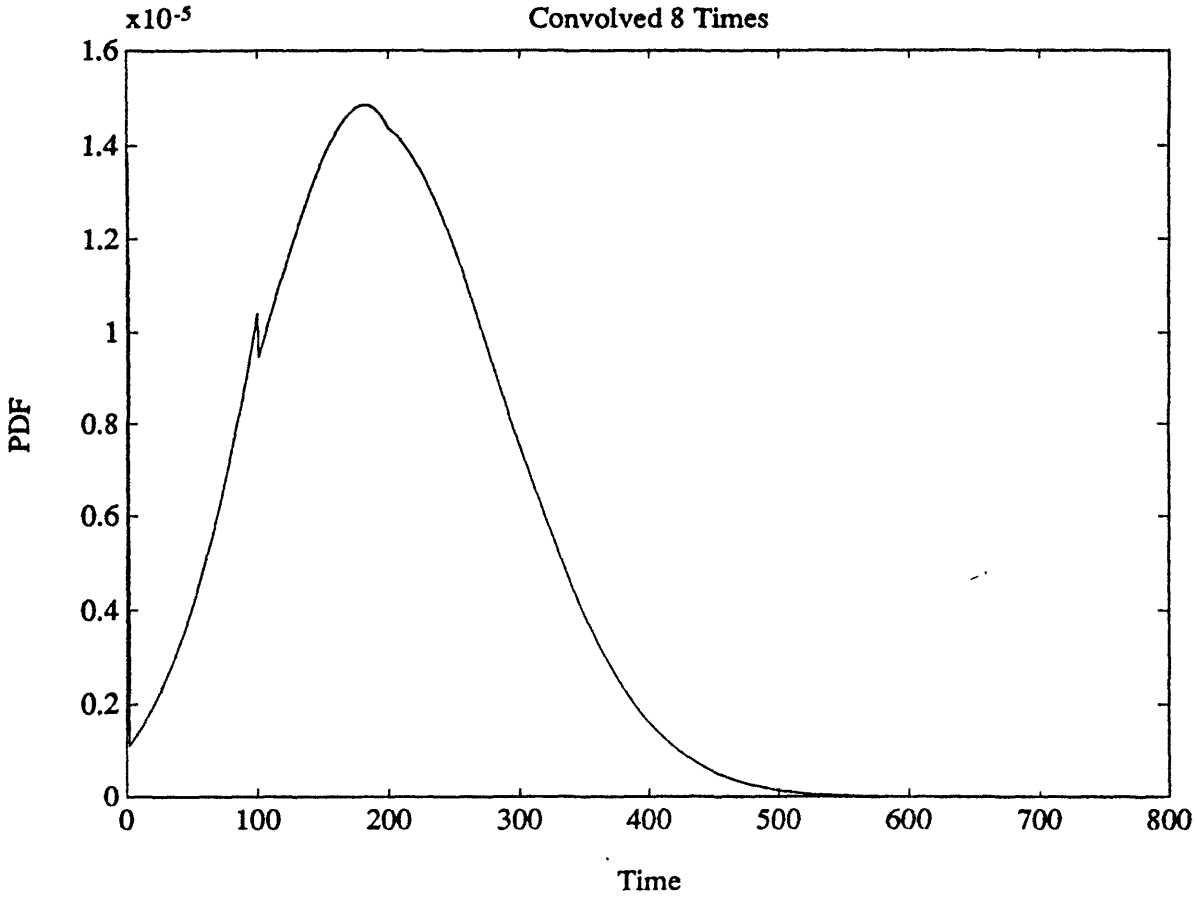


The diagram above shows the probability density function for the total waiting time a vehicle would experience if it passed through four independent intersections.

Figure 5.2 /

Notice that after four traffic signals, the distribution is still relatively flat and non-normal. Perhaps, if we convolve the distribution a few more times it will become a good approximation to the normal distribution.

In figure 5.3, the original p.d.f has been convolved eight times.



The diagram above shows the probability density function for the total waiting time a vehicle would experience if it passed through eight independent intersections.

Figure 5.3

The distribution is still not very normal looking. Notice that the left side is quite truncated and there are still obvious discontinuities at 100 and 200 time units. It turns out that one must convolve the p.d.f. about 20 times before the discontinuities settle out.

We can draw some very simple conclusions from this simple model. Recall that we are assuming that the vehicle's stops and waiting time at each intersection is independent. First of all, if a car travels along **u** uncoordinated arcs then the vehicle can *expect* to stop **u/2** times. The variance on the distribution of stops due to traffic signal control would be **u/4**. The expected amount of time a car would wait for traffic signal control would be **uC/4**. Now, a vehicle has a 50/50 chance of being held at the first intersection a vehicle encounters in **G(N,A)**. Remember, from chapter four, there is another source of potential stops as well, alignment delays. Let **v** represent the number of intersections where the vehicle enters the intersection from a direction with an alignment delay. In total, a vehicle traveling along a route with **v** coordinated arcs with alignment conflicts and **u** uncoordinated arcs could expect to stop **v+(u+1)/2** times and wait for **[v+(u+1)/4]C** time units.

SECTION 5.1.2 Stochastic Approximations for Varied Splits

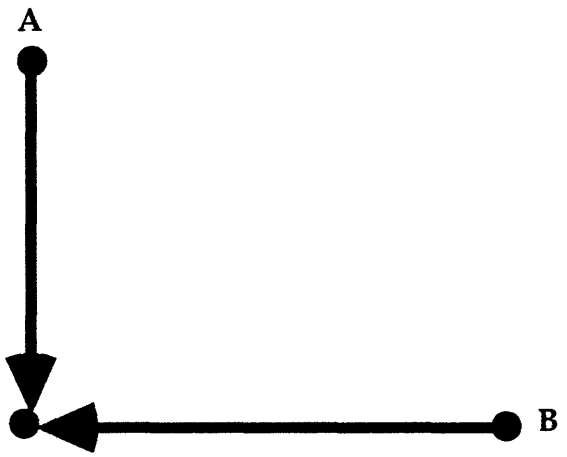
Now, we know from theorem 4.3.2 that in a large network roughly one fourth of the arcs can be coordinated. If we can determine something about the distribution of arcs per route then we can use some powerful techniques to generate bounds on the probability of stopping a given number of times or having to wait for a given period of time. **This is irrespective of the timing plan used.** Ergo, this powerful tool could be used to analyze networks and timing plans in the general case. Recall our independence assumption from the preceding section. Thus, if we knew the splits at all the intersections a vehicle were traveling on, we could express the s-transform of the waiting time for the uncoordinated arcs as:

$$f_w^t(s) = \prod_{\forall j \text{ s.t. } (X_i(j-1)\rightarrow X_i(j))\notin S} f_{w_{x_i(j)}}^t(s)$$

where $f_{w_{x_i(j)}}^t(s) \equiv s$ – transform of waiting time distribution for intersection $X_i(j)$

.

For the light traffic model, some results are quite startling. Imagine the situation displayed below in figure 5.4.



There are two approaches, labeled A and B, to this intersection.

Figure 5.4

Traffic enters the intersection from two directions, **A** and **B**. We will call the vehicular arrival rate from direction **A** to be λ_A and from **B** to be λ_B. Additionally, designate the green light for direction **A** to be of duration G_A and from **B** to be of G_B. Now, as long as the vehicular arrivals are non-periodic, renewal theory tells us that the chance of stopping for a particular vehicle is just the fraction of time when a vehicle would be required to stop over the cycle, i.e. the red time in that direction. For example, a motorist traveling from direction **A** would have to stop when the light was green for

direction **B**. So, a *randomly selected vehicle* traveling from direction **A** has a probability $\frac{G_B}{G_A+G_B}$ of stopping, or $P(stop \mid A)=\frac{G_B}{G_A+G_B}$. Likewise, $P(stop \mid B)=\frac{G_A}{G_A+G_B}$. Now, the probability that a randomly selected vehicle is traveling from the **B** direction is $\frac{\lambda_B}{\lambda_A+\lambda_B}$, and for **A** it is $\frac{\lambda_A}{\lambda_A+\lambda_B}$. We are going to use the interesting fact that for a single intersection, *the expected number of stops in the light traffic model is the same as the probability of stopping*. This allows us to draw on our knowledge of expectations. Conditional expectation tells us:

$P(Stop)=E(stop)=E(stop \mid A)P(A)+E(stop \mid B)P(B)=\frac{\lambda_A G_B+\lambda_B G_A}{(\lambda_A+\lambda_B)(G_A+G_B)}$. Remember that the number of stops for the system was one of our Measures of Effectiveness (M.O.E.) from chapter 3. So far, we have calculated the expected number of stops per vehicle at this intersection. In steady state, the expected number of vehicles entering the intersection over time **t** would be $(\lambda_A+\lambda_B)t$. Thus, the total expected stops for the intersection over time **t** would be: $\frac{(\lambda_A G_B+\lambda_B G_A)t}{(\lambda_A+\lambda_B)}$.

Certain current traffic signal control methods make a convexity assumption about affine combinations of average stops per vehicle and average waiting time per vehicle with respect to cycle time. (An affine combination is a weighted sum such that the sum of the weights is equal to one.) Such convexity assumptions are not necessarily true. Consider again the simple network described in figure 5.4. Now let **C**=the cycle time=G_A+G_B where G_A is allowed to vary and G_B is fixed. Now, performing the second derivative test on the expected number of stops with respect to G_A, one obtains:

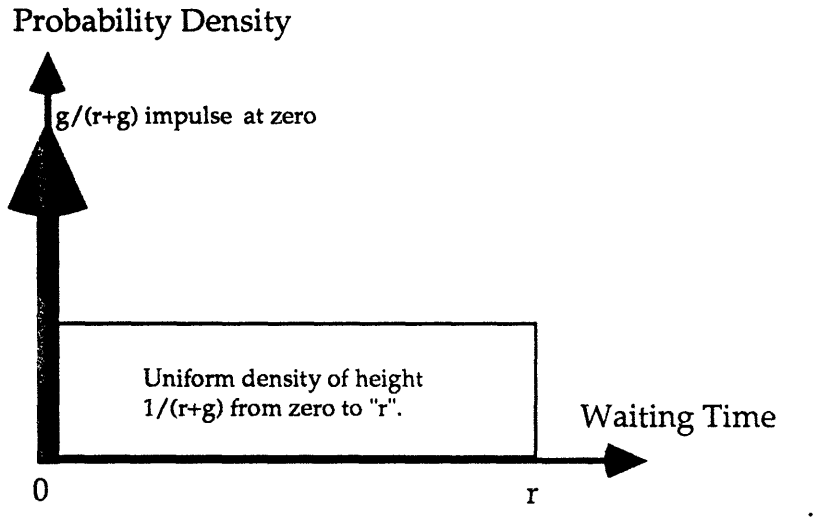
$\frac{2(\lambda_A-\lambda_B)(G_A+G_B)G_B}{(\lambda_A+\lambda_B)^2(G_A+G_B)^4}$. Note that this quantity is positive if and only if $\lambda_A>\lambda_B$.

The expected waiting time is derived using conditional expectations in a similar fashion to the average stops per vehicle and expected stops due to traffic signal control

over time t. Explicitly we obtain the expected waiting time per vehicle in figure 5.4 is $\frac{\lambda_A G_B^2 + \lambda_B G_A^2}{2(\lambda_A + \lambda_B)(G_A + G_B)}$ which is convex with respect to cycle time.

If we view the vehicle as arriving at a random time at intersection $X_i(j)$ along an uncoordinated street segment and that intersection has **r** red units of time and **g** green units of time then we could give the s-transform for the waiting time as:

$f^t_{w_{x_i(j)}}(s) = \frac{gs + r(1 - e^{-rs})}{(r+g)s}$ for that particular intersection. We can also directly describe the probability density function:



The diagram above shows the probability density function for the total waiting time a vehicle would experience at a generalized intersection with "g" seconds of green time and "r" units of red time.

Figure 5.5

Now that we have a better intuition for the light traffic model, we can apply that intuition to the moderate traffic model where queuing is allowed to occur.

SECTION 5.2 Queue Effects Model (QEM)

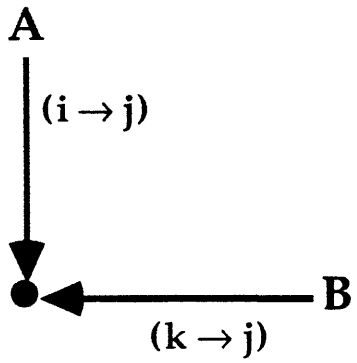
In this section, we will build a model of behavior for individual intersections based on our earlier network foundation and using some queuing theory to derive several results. Our objective is to predict the average amount of time a vehicle will have to wait at a random intersection when approaching an intersection along an arc that is *not part* of the spanning tree, S, for the offsets. We will use the average wait along with desired number of vehicles we wish to pass in arcs which are elements of S to derive the cycle time and assign splits. Before we can do this however, we must understand the model used to determine waiting time.

SECTION 5.2.1 Priority Queue Model for QEM

We make the transition here from a strictly light traffic model to a medium traffic model. Thereby, we change assumption 4.1.1(4) to read ***traffic is assumed to be at most of moderate intensity***; specifically, vehicles waiting at a red light will be able to pass through subsequent green cycles on a *steady rate*, but some vehicles many have to wait through more than one red cycle. So, moderate traffic constitutes any situation where there are congestion effects and the vehicles are *able* to follow the TSC plan; in other words, nearly all the urban traffic in the United States falls in the moderate category. This is opposed to a heavy traffic model where cars may *not be able to pass through a green light*, because **the follow on street segment is full**; cars which proceed anyway cause a phenomenon known as "gridlock". The heavy traffic model passes out of the realm of TSC into the areas of capacity and urban planning. We will also eliminate assumption 4.1.1(8), because *the splits and cycle time will now be allowed to vary*. Assume that vehicles

require $\overline{x}_1$ time units to pass through the intersection. View $\overline{x}_1$ as a degenerate random variable, i.e. it has a constant value. Vehicles traveling on arcs$\in S$ will arrive at intersections in platoons. Define H to be the minimum platoon size we wish to accommodate in going from an arc$\in S$ through an intersection in our TSC strategy. Then, we would require $\overline{x}_1 * H$ time units as a minimum for the green cycle in the vehicles on $\forall (i \rightarrow j) \in S$. Intersections come in two primary varieties in G(N,A). Either the intersection has one arc connected to it that is an element of S or it has two or more. Why? The intersection has to have at least one arc connected to it which is an element of S, because S is a spanning tree. Since we allow a maximum of two orthogonal directions at each intersection, this means that we can have either one or two directions which have a minimum green cycle time requirement of $\overline{x}_1 * H$. We will examine the various combinations of arc types at each intersection in the last portion of this section, but for the moment, let us concentrate on the more interesting case where there is one direction which contains no arcs$\in S$.

Let us calculate the average wait for a vehicle which approaches the intersection along the arc which is not an element of S. Just as Cedar (1989) did, we will use a priority queuing model, but unlike Cedar, we will not use the traditional M/G/1 queuing model which is too restrictive. Instead, we will use the diagram below to define our system.



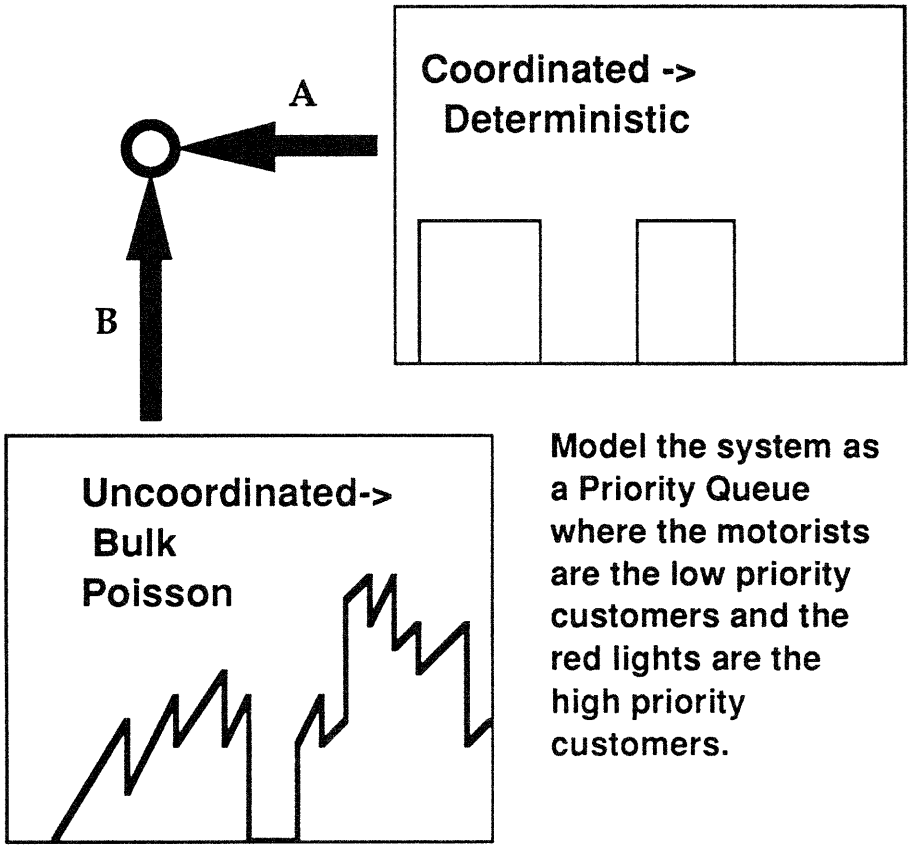
$(i \rightarrow j) \in S$ $\qquad\qquad$ $(k \rightarrow j) \notin S$

There are two approaches to the intersection. Arc (i->j) is part of the tree, and arc (k->j) is not

Figure 5.6

Assume that vehicles arrive at "j" along $(k \rightarrow j)$ in groups, and we know the number of vehicles in any particular group is distributed according to the P.M.F. **b(h)**, i.e. **b(h)=Prob.("h" vehicles in the group)**. [We actually only require a mean and a measure of dispersion/variance to make the calculation.] It seems reasonable that based on stragglers, previous right and left turns, curbside departures, etc. that the arrival times of the groups would be fairly autonomous. *(Note, if the arrivals in the non-S direction also occur only at discrete intervals then we have the same situation as when we have both directions with all arcs which are elements of S, and the cycle time and splits calculations become trivial.)* Assume that over a reasonably short period of time, the arrival rate of the groups is constant. Call this arrival rate λ . Further assume that the chance of two or more groups arriving at *exactly* the same time, coming from the same direction is zero. If we also reasonably expect the chance of a group arriving over a particular time interval to be proportional to the length of that interval and *non-overlapping* intervals to be independent (both of which pass first order "sanity" checks, i.e. the assumptions seem fairly reasonable) then we could conclude the *number of groups* arriving over some

fixed time frame are distributed as a Poisson random variable with mean λ. Define the average number of vehicles which arrive in direction **B** per unit time as $\lambda_1 \triangleq \sum_{i=1}^{\infty} i\lambda b(i)$. Figure 5.7 graphically displays the situation described in this paragraph. Even though the vehicles arrive in groups, they are serviced individually. That is, no more than one vehicle will cross the intersection at the same time. We also assume that vehicles are served in a first come, first served order.



There are two approaches to the intersection. Along the segment which is in the tree, the vehicles arrive in an orderly fashion, but the arrivals along the non-tree segment come in bulk arrivals with exponential headways.

Figure 5.7

As a reminder, our primary interest in this section is the uncoordinated side streets. Call the amount of green time in the A direction G_A and the amount of green time in the **B** direction to be G_B. We will model two customer classes, vehicles and red lights,

where red lights take priority over vehicles. We will refer to vehicles as type one customers and red lights as type two customers. Thus, any time we use the subscript one it refers to vehicle customer attributes, and any time we use the subscript two, it refers to red light attributes. For example, we know that $\bar{x}_1$ is the average amount of time it takes for a vehicle to cross the intersection; what would $\bar{x}_2$ be? Well, we know that it takes exactly G_A time units for a red light. So, we have $G_A = \bar{x}_2$. Red lights arrive deterministically at intervals of $G_A + G_B$ time units. The time between vehicle group arrivals is negatively exponentially distributed with mean λ (this flows directly from the Poisson distribution of group arrivals over time). We define $\rho_1 \triangleq \lambda_1 \bar{x}_1$. This represents the fraction of time the server, the intersection, is busy with vehicles. Let $\rho_2 \triangleq \frac{G_A}{G_A + G_B}$. For system stability, we must have $\rho_1 + \rho_2 < 1$ which we can write out directly as $\lambda_1 \bar{x}_1 + \frac{G_A}{G_A + G_B} < 1$. We notice that this can be rewritten as $\lambda_1 \bar{x}_1 (G_A + G_B) < G_B$ which tells us that on average the number of vehicles arriving during an entire cycle must have enough time to cross the intersection during the green light. Otherwise, the vehicles would stack up at the intersection indefinitely, and the average waiting time would go to infinity.

Originally when analyzing this problem, the author turned to Kleinrock's (1976) approach to priority queues. Afterall, a deterministic interarrival time is a subset of a general independent interarrival time distribution. Unfortunately, somewhere deep in moment generating space, two complications arose. One can show for example that *(after much simplification)* that

$$\text{average wait} = -\frac{d}{ds}[G_C^*(s)]\Big|_{s=0} = -G_0^{*'}(\lambda - \lambda)[-\lambda A'(1)]\hat{G}_0^{*'}(0)$$

Equation 5.4

which says the average wait is equal to the average amount of time a random incident customer in service takes times the average arrival rate for groups times the average busy period length. The problem is the average busy period length is extremely elusive in this case, because the busy period duration is **not** independent of time. Here is why. We know that a red light will occur after exactly G_B time units. Look at a sample timeline for the intersection.

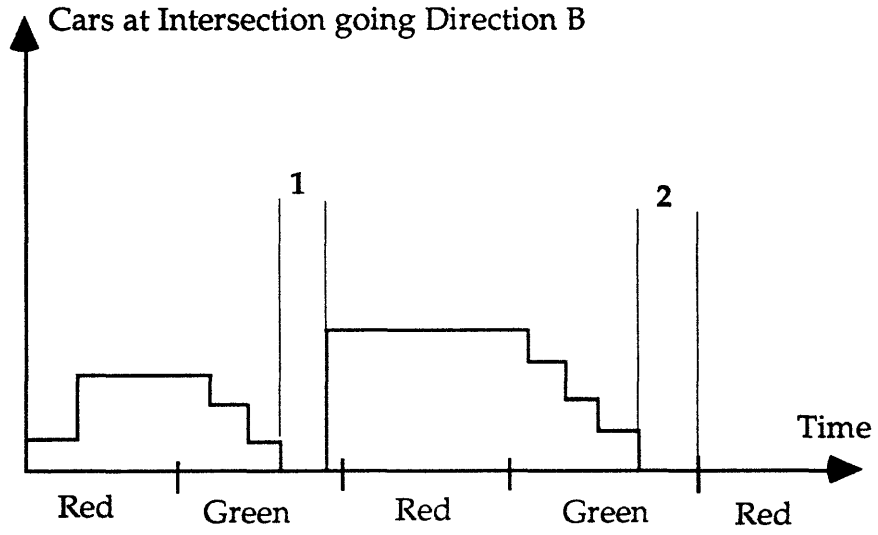


Diagram shows the vehicle queue at the intersection as a function of time.

Figure 5.8

We assumed above that the interarrival periods between platoons of vehicles was exponentially distributed. That means that theoretically the gap could be of a very large duration, and certainly the gaps could be more than one complete cycle length in duration, i.e. there could be several cycles where absolutely no vehicle arrive at the intersection even though the traffic flow is moderate. In figure 5.8 gap **1** represents the interarrival gap between successive platoon arrivals, but the distribution for gap **2** is something quite different. The second gap represents the amount of time between the last vehicle service and the start of the red light. Note, in many cases this quantity would be zero, i.e. vehicles will have to wait for the red light before passing through the

intersection. To characterize the second gap, one needs to be able to explicitly describe the probability of servicing "x" vehicles during a green light period given "y" vehicles present at the beginning of the green light. Then one is left in the unfortunate position of having to solve a recursion using this preliminary result to derive the time gap distribution. Specifically, we know a`priori the *fraction* of time during the green light when the system is empty is $(1-\rho_1)$, and we know that all the gaps which occur during a period are of exponential duration *except* for the last one. Additionally, we know the distribution for the number of groups which arrive during any given period. So, the first complication was the time dependent nature of the last gap (note: this gap might not exist at all on a given cycle if there are cars waiting to cross when the light turns red again). The second complication was even more insidious. The existence of bulk arrivals means that waiting time for customers is **not** independent of arrival time and bulk arrival size. The bigger the size of the bulk arrival, the greater is the chance that some or all of that arrival will have to wait for a subsequent red light. Again, the lack of independence means that we require a more explicit characterization of the system dynamics to get results from this path. But, do not despair. There is a technique which does reveal the quantity in question, mean wait time, and although not conceptually simple is computationally easy and elegant in its application of our earlier robust assumptions.

SECTION 5.2.2 Expected Mean Delay and Stops in Uncoordinated Direction

We know that the expected value of a sum of random variables is equal to the sum of the expected values of those same random variables always *even with dependence.*. Therefore, remembering that type one customers are vehicles and type two customers (the priority customers in this model) are red lights, one can say:

$$\overline{W}_S = \overline{W}_0 + \sum_{i=1}^{2} \overline{x}_i \, (\mathbf{Before}_i + \mathbf{After}_i), \qquad [5.5]$$

or the time average amount of time a random vehicle will wait at the traffic signal before it passes through the intersection is equal to the random incident average amount of time the vehicle must wait for the customer in service (either vehicle or red light) plus the mean service time for all the customers who will be served before our vehicle in question. We can write out eq. 5.5 explicitly in words as follows:

the average wait for a vehicle is equal to the average amount of time the arriving vehicle must wait for the customer in service (whether red light or vehicle) plus the average amount of time it will take for each customer type to be served times the average number of that type of customer expected to be served before a vehicle arriving at a random time.

$\mathbf{Before}_i$ refers to the expected number of all the type "i" customers who are in line *when our random vehicle arrives* which will be served before the random vehicle can pass through the intersection. Like wise. $\mathbf{After}_i$ refers to the expected number of all the type "i" customers who *arrive after our random vehicle arrives* which will be served before the random vehicle can pass through the intersection. Type 1 customers are vehicles. Let us assume that cars will pass through the intersection in the order they arrive (probably a safe assumption anywhere except Boston or Berkeley!), i.e. First Come, First Serve (FCFS). This implies that $\mathbf{After}_1$ equals zero. $\overline{W}_0$ is the average amount of time the random arriving customer must wait for the customer in service to complete service, e.g. for the car there to pass through the intersection or for the red light to finish, and $\overline{W}_S$ is the average wait for our vehicle, the quantity we are after. We can immediately do some simplifications and substitutions. Type 2 customers are red lights. Whenever a red light arrives, it moves immediately to the front of the line and starts being served, i.e. prevents vehicles from passing through the intersection. The $\overline{W}_0$ term is relatively straightforward to calculate. We arrive during a red cycle with probability $\left(\frac{G_A}{G_A + G_B}\right)$. We expect to see an average of $G_A/2$ units of red time remaining in

that light when we arrive. If we arrive during a green light and there is a vehicle passing through the intersection when we arrive then we would expect on the average for the vehicle to take $\bar{x}_1/2$ more time units to travel through the intersection. The probability the light is green *and* a vehicle is passing through the intersection when our vehicle arrives is: $\frac{G_B\rho_1}{G_A+G_B}=\frac{G_B\bar{x}_1\lambda_1}{G_A+G_B}$. Ergo,

$$\overline{W}_0=\frac{G_A^2+G_B\bar{x}_1^2\lambda_1}{2(G_A+G_B)}. \qquad [5.6]$$

The quantities $\mathbf{Before}_1$ and $(\mathbf{Before}_2+\mathbf{After}_2)$ are *strongly related.* We know that a maximum of $\left\lfloor\frac{G_B}{\bar{x}_1}\right\rfloor$ vehicles can pass through the intersection during a given green light. Therefore, if there are J vehicles at the intersection when the car in question arrives then the random vehicle will have to experience $\left\lfloor\frac{\bar{x}_1*J}{G_B}\right\rfloor$ or $\left\lfloor\frac{\bar{x}_1*J}{G_B}\right\rfloor$-1 red lights before it can pass through the intersection. If our random vehicle arrives during a red light it must wait for **exactly** $\left\lfloor\frac{\bar{x}_1*J}{G_B}\right\rfloor$ red lights; a randomly incident vehicle experiences a red light with probability $\frac{G_A}{G_A+G_B}$. Now, if the light is green when our vehicle arrives, which occurs with probability $\frac{G_B}{G_A+G_B}$, the expected number of red lights our vehicle must wait for is $\frac{\bar{x}_1*J}{G_B}$ {without the floor function}. This is more apparent after considering an example. Let $G_A=G_B=6$, $\bar{x}_1=2$ and J=7. During each green light, three vehicles can pass through the intersection. If our vehicle arrives during a red light, there will be two green cycles *before* the green cycle which allows our vehicle to pass through the intersection which is $\left\lfloor\frac{\bar{x}_1*J}{G_B}\right\rfloor$ **full** red lights, but if we arrive during a green cycle then a random number of cars will pass through the

intersection before the next red light. However, the number of cars passing through the intersection after our arrival and before the next red light is strictly greater than zero and less or equal to $\left\lfloor \frac{G_B}{\bar{x}_1} \right\rfloor$, three in this case. Specifically,, consider the table below:

Cars passing through on Arrival Green	Full Red Lights we must wait for
1	3
2	3
3	2

Table displays number of full red lights a vehicle must wait for depending on how many vehicles pass through the green light during the split when it arrives.

Table 5.1

Assuming each of these possibilities is equally likely gives us an average of $1/3*(3+3+2)=2.66667=\frac{\bar{x}_1 * J}{G_B}$ full red lights to wait for. Now, the number of vehicles before our randomly selected vehicle is not independent of whether the light is red or green, but we can place an *upper bound* on the number of vehicles in front of the randomly chosen vehicle by using the red light value. We would expect to have to wait for more red lights if we arrive during a red light than if we arrive during a green light for two reasons. First, there are likely to be more vehicles queued in front of us if we arrive during a red light, and two, the vehicles that are in front of us are not being serviced during the red light we arrive in. Thereby, an *upper bound* on the wait due to servicing the customers before our vehicle (excluding the vehicle in the intersection when we arrive is):

$$\bar{x}_1 * \text{Before}_1 + G_A\left(\frac{\bar{x}_1 * \text{Before}_1}{G_B}\right)$$ In total, we can say:

$$\overline{W}_S \le \frac{G_A^2 + G_B \bar{x}_1^2 \lambda_1}{2(G_A + G_B)} + \bar{x}_1 * \text{Before}_1 + G_A\left(\frac{\bar{x}_1 * \text{Before}_1}{G_B}\right)$$

Inequality 5.7

So, we are left with two unknown quantities, $\mathbf{Before}_1$ and $\overline{W}_S$. Luckily, these two quantities are related. How? We use the fact that the interarrival times of the groups are negative exponentially distributed. This means that, because Poisson Arrivals See Time Averages (PASTA), that the average lead vehicle in a group sees the time average as well. Using Little's Law then, we can say $\overline{W}_S\lambda_1 = \overline{N}_S$ where $\overline{N}_S$ is the system average number of vehicles on the street segment waiting to cross the intersection. What is the relationship between $\mathbf{Before}_1$ and $\overline{N}_S$? Actually, $\mathbf{Before}_1 \geq \overline{N}_S$ in this case. Why?

Imagine the case where the bulk arrival size is **always two**, and vehicle platoons still have exponential headways. The lead vehicle sees the system time average, because Poisson Arrivals See Time Averages (PASTA). But, the second vehicle sees the time average **plus one**. Thus, the average vehicle sees the time average plus one half. We might well ask the question are there ever cases where the average customer sees something **less** than the time averages? *Just as an example the other direction, it is possible to see less than the time average number of customers in the system when a random customer arrives. Consider a D/D/1 queue where the service time is less than the interarrival time. The customer arrives to find the server empty, but the time average number of customers in service is certainly greater than zero!*

In general, $\mathbf{Before}_1 = \overline{N}_S + \nu$ where ν is the average number of vehicles in front of a random vehicle approaching the intersection. What is the chance that our arriving vehicle finds itself in a platoon of size "i" vehicles as it reaches the intersection? There is a random incidence argument involved here as a randomly chosen vehicle is

far more likely to be part of a large group than a small one. Working through all the algebra gives $\nu = \sum_{i=1}^{\infty} \frac{(i-1)b(i)i}{2\bar{b}} = \sum_{i=1}^{\infty} \frac{1}{2\bar{b}}\left[i^2 - i\right]b(i) = \frac{\text{Variance(b)}}{2\bar{b}}$. Therefore,

$$\text{Before}_1 = \bar{W}_S\lambda_1 + \frac{\text{Variance(b)}}{2\bar{b}}. \qquad [5.8]$$

Now we can transform [5.7] into an inequality with one unknown:

$$\bar{W}_S \le \frac{G_A^2 + G_B\bar{x}_1^2\lambda_1}{2(G_A + G_B)} + \bar{x}_1*\left[\bar{W}_S\lambda_1 + \frac{\text{Variance(b)}}{2\bar{b}}\right] + \frac{G_A}{G_B}\left(\bar{x}_1*\left[\bar{W}_S\lambda_1 + \frac{\text{Variance(b)}}{2\bar{b}}\right]\right)$$

Inequality 5.9

This gives us a another traffic parameter for the traffic manager to use for determining the best TSC for his or here area of interest. Specifically, the traffic manager could say "I want the *average* wait for a traveler to wait no more than 30 seconds at a traffic light due to traffic signal controls." Since we have an upper bound on the average waiting time, we could select values for the splits to accommodate this request; this is exactly the approach we will take in chapter six.

Before moving on to a new section, it would be useful at this point if we did a "sanity check" on equation 5.9. To do this, first scale all the quantities so that the cycle time, (G_A+G_B), equals one (w.l.o.g.). Then inequality 5.9 becomes:

$W_s \le \dfrac{G_A^2G_B + G_B^2\bar{x}_1^2\lambda_1 + 2\bar{x}_1 \text{Variance(b)}/2\bar{b}}{2(G_B - \bar{x}_1\lambda_1)}$. This relation makes a lot of intuitive sense. As $G_B \leftarrow \bar{x}_1\lambda_1$, i.e. the time average fraction that the green light is occupied by vehicles passing through the intersection approaches one, the upper bound on the waiting time goes to infinity. As the red light time goes to zero, the waiting time only

becomes a function of the number of cars we would expect to see in front of a random vehicle when it arrives.

Since inequality 5.9 passes the sanity check with flying colors, let us try an example. Say that the green light in the priority direction has a duration of 25 seconds, and the green light in the non-coordinated direction has a duration of 15 seconds. We will assume that platoons on the non-tree arcs arrive in a bulk Poisson process as described above with a mean interplatoon arrival time of two minutes, and platoons come in two sizes. Platoons are equally likely to be of size one or size three. Assume that the average vehicle takes three seconds to safely cross the intersection. In equation form, we have (in minutes):

$G_A = 25/60 \quad G_B = 15/60 \quad \lambda_1 = 1 \quad \bar{x}_1 = 3/60$

$\text{Variance}(b)/2\bar{b} = 1/4$. Then inequality 5.9 tells us that the average wait for a vehicle traveling along the non-coordinated street segment would expect a delay of less than 8.25 seconds or .1373 minutes at this intersection.

Using the same type of reasoning we used to derived inequality 5.9, we can calculate the expected number of stops experienced by a vehicle to be bounded by the following inequality:

$$E(\text{Stops}) \leq \frac{G_A + G_B \bar{x}_1 \lambda_1}{(G_A + G_B)} + \left[\bar{W}_S \lambda_1 + \frac{\text{Variance}(b)}{2\bar{b}} \right]. \qquad [5.10]$$

So, we can predict the number of stops and average wait for the vehicles in our network, but there may be times when it is desirable to approximate more than just the average amount of time a vehicle will have to wait per intersection. We may want to

know some characteristics of the queuing system in more detail. We present a transform approximation for the time average queue length in the next section.

SECTION 5.2.3 Approximation for Z-Transform of Time Average Queue Length

The following equation represents an approximation to the true steady-state z-transform. Specifically, the transform was derived using the assumption that the steady

$$\frac{Q(z)}{e^{\lambda\bar{x}_1(B(z)-1)}} = \left(\frac{R}{R+G}\right)Q(z) + \left(\frac{G}{R+G}\right)\left[1 - \lambda_1\bar{x}_1 + \frac{Q(z) + \lambda_1\bar{x}_1 - 1}{z}\right],$$

where **Q(z)** represents the z-transform of the p.m.f. for the time average number of cars in the queue.

The Z-transform approximation passes several "sanity" checks. For example, if one eliminates red time it becomes the transform for an M/D/1 queuing system. If one eliminates the green time then no steady state solution exists.

In the past two sections we examined the measures of effectiveness under both the light traffic and moderate traffic model settings. But, what are the theoretical upper bounds we are constrained to? In thermodynamics, we have the famous Carnot heat engine results. No engine produced can exceed the efficiency of the theoretical Carnot engine because of the second law of thermodynamics. Is there a comparable upper limit in the realm of traffic signal control? Indeed there is.

SECTION 5.3 Network Multi-Commodity Flow Approximation for Unconstrained Travel Times

The best possible situation in a traffic flow environment is if **no one** encounters **any red** lights and passes directly through the network. We can mathematically approximate this by allowing every vehicle to shrink to an infinitesimal point and *turn all of the traffic lights green.* How would we possibly model such a situation? Well, really it turns into a network multi-commodity flow problem. We could view every vehicle *flowing* from a particular origin in **G(N,A)** to a particular destination to be composed of a specific fluid. We will have a source, the route's origin, and a sink, the route's destination, corresponding to each route through the network. We retain the capacity limits on the arcs in terms of vehicles per unit time; this is a function of the speed limit and the number of lanes on the street segment. As a cost per arc, we will use the amount of time required to traverse it at the speed limit.

The procedure defined above constitutes a minimum cost, constrained, multi-commodity flow problem. The output from this model would be the **absolute lower bound** on the amount of time it would take to traverse the network, assuming no one stopped for a traffic light anywhere in the system and no one broke the speed limit. Using our definitions from chapter three, this model can be formulated as shown below:

$$\text{Min} \sum_{h=1}^{\|R\|} \sum_{(i,j)\in A} x_{ijh} \hat{t}_{ij}$$

subject to

$$\sum_{h=1}^{\|R\|} x_{ijh} \leq \rho_{ij} \ \forall \ (i,j) \in A$$

$$\lambda_h O(j,h) + \sum_{i=1}^{\|N\|} x_{ijh} = \sum_{k=1}^{\|N\|} x_{jkh} + \lambda_h D(j,h) \ \forall \ j,h$$

$$x_{ijh} \geq 0 \ \forall \ i,j,h$$

where **D(j,h)** is one if node **j** is a destination for route **h**, and **O(j,h)** is one if node **j** is an origin node for route **h**.

We determined how to select the most important street segments *and* synchronize them in chapter four. In this chapter, we examined several ways of determining the MOE's given specific traffic signal control settings. In the next chapter, we will describe a method, Predictive Routing Information Signal Timing INtEgration (PRISTINE), of setting the splits, offsets and cycle time for **G(N,A)** using our work in the previous chapters.

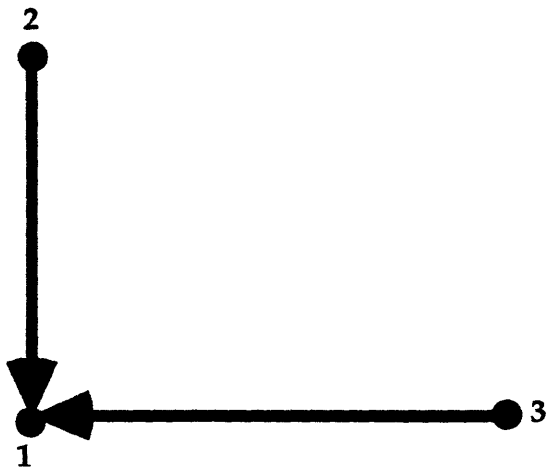
Chapter Six

Setting Split and Cycle Times

SECTION 6.1 Traditional Traffic Engineering Approaches to Setting Splits and Cycle Time

The methodology for setting off-sets can be independent of both splits and the cycle times, but once one crosses the boundary into the realm of split and cycle time setting, one must approach the problem from a more holistic standpoint (Hadi and Wallace, 1991; Hawat 1992; Hobbs 1979; et.al.). In this chapter, we will explore the issues of split and cycle-time setting. We will begin by getting an intuition for traffic signal control strategies through examination of the traffic lite or non-queuing model, and then we will move onto look at the Queue Effects Model (QEM) where queuing effects become significant. The QEM will form the basis of setting splits in the Predictive Routing Information Signal Timing INtEgration (PRISTINE) traffic signal setting method.

The most logical point to start the examination of splits and cycle times is to consider an isolated intersection. At first, imagine that the traffic is of very light intensity (e.g. the average time between successive arrivals to the intersection is on the order of ten minutes). Consider the intersection shown below:



Sample intersection with arrivals from two directions

Figure 6.1

Let the following relationships hold.

Direction	Average Arrival Rate	Green Time
$2 \rightarrow 1$	λ_A	G_A
$3 \rightarrow 1$	λ_B	G_B

Information about intersection shown in figure 6.1 (above)

Table 6.1

Assume that successive arrivals to the intersection are independent, i.e. assume the interarrival times are independent, and non-periodic, i.e. the arrivals occur at non-deterministic and do not always occur at times of the form **nV** where **n** is a positive integer and **V** is some real, positive number. Additionally assume that the motorists who arrive at the intersection and are stopped by a red light will be able to depart on the subsequent green cycle. Now, renewal theory tells us that the average number of motorists who will have to stop during a cycle

is: $\lambda_A G_B + \lambda_B G_A$. Thus, over a long period of time the fraction of motorists who will be forced to stop is

$$\frac{\lambda_A G_B + \lambda_B G_A}{(\lambda_A + \lambda_B)(G_A + G_B)}. \qquad [6.1]$$

Under these same conditions, the average amount of time a motorist would expect to spend waiting for a red light is:

$$\frac{\lambda_A G_B^2 + \lambda_B G_A^2}{2(\lambda_A + \lambda_B)(G_A + G_B)}. \qquad [6.2]$$

Given formula 6.2, how would one change the traffic signal settings to decrease the average wait per vehicle? One would want to diminish G_A or G_B to decrease the expected waiting time. In other words, the smaller one makes the cycle time the less amount of time the average customer would have to spend at the intersection due to traffic signal control. (Of course, there is some *practical* limit to this approach. One would certainly not want to set the split for any given direction below the amount of time required *for at least one vehicle* to clear the intersection. So, a cycle time of 1/100th of a second looks great on paper, but it is of little practical significance.) As we decrease the green splits we must ensure that we do not decrease the green time in the more heavily traveled direction more quickly than we decrease the green time for the lesser traveled direction. Note also that *the probability* the car will be required to stop will not change if G_A and G_B are decrease proportionally. This is easiest to see if you consider a lone car approaching the intersection. The chance the car will have to stop is the fraction of red-time the intersection experiences in the driver's

direction of travel. Likewise, the driver this vehicle expects to wait one-half of the red time in his direction if required to stop.

An important issue was realized by the simple example in figure 6.1, one which we will revisit in the latter portion of this chapter. Namely, what is the practical minimum, one desires for the effective green time in a given direction?

Effective green time is the critical issue as the traffic intensity (vehicles per unit time) increase at an intersection. Each time one changes the signals, red to green, green to red, at an intersection, some amount of usable intersection crossing time is lost due to accelerations/deceleration's and safety considerations to allow the intersection to clear. Thereby, in moderate to heavy traffic, one wants to make effective green times as long as possible in busy directions (i.e. there are certain *practical* restrictions to how long we can hold traffic in the non-priority direction) to maximize the fraction of useful time and the flow rate (vehicles passing through the intersection per unit time). But, there is a practical upper limit to the cycle-time and splits. A green light in one direction necessitates a red light in the other. The longer a light is red, the longer the queue tends to get in that direction, and if the light is red too long then the cars eventually spill back into other intersections, causing "gridlock." Additionally, experience has shown that between 120-150 seconds is the largest practical cycle-time that the average motorist will endure (Homburger, 1988). After that time limit, motorists in the US tend to do more risky maneuvers (e.g. cutting across on a red, causing gridlock, etc.).

A large amount of research has gone into various "practical" formulas for determining the best time for splits and cycle times. Much of the work has been

done looking at isolated intersections. To get a more intuitive feel for the effects of the various splits, again use the situation depicted in fig. 6.1.

Let us now look at a specific case where the interarrival times are independent and non-periodic. Our approach here will be to get some specific mathematical insights into the system behavior which we will apply later in the max split version of PRISTINE. Assume the arrivals in **both** directions are Poisson. First, let us calculate the expected number of vehicles which will stop during any particular cycle. This is evidently $\lambda_A G_B + \lambda_B G_A$. Let $C=G_A+G_B$. Now, the probability that any particular car must stop is given by:

$P_s \equiv$ **Probability of a random car stopping**

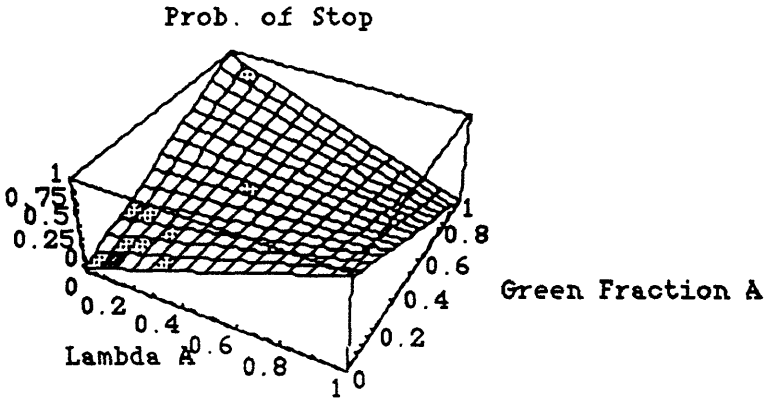
$= P(\text{Stopped}) = P(\text{Stopped}|A)P(A) + P(\text{Stopped}|B)P(B)$ [Note: $A \cap B = \emptyset$]

$$= \left(\frac{G_B}{C}\right)\left(\frac{\lambda_A}{\lambda_A+\lambda_B}\right)+\left(\frac{G_A}{C}\right)\left(\frac{\lambda_B}{\lambda_A+\lambda_B}\right)=\frac{G_B\lambda_A+G_A\lambda_B}{C(\lambda_A+\lambda_B)}. \qquad [6.3]$$

Note, this agrees with our earlier renewal theory result (which is always reassuring!). Now, in the following example let **V** be a random variable, the number of vehicles passing through the intersection during a cycle, and let **S** be a random variable, the number of vehicles stopped due to the traffic signal during the cycle. Thereby,

$$E\left(\frac{S}{V}\right)=\sum_{v=1}^{\infty}\left(\frac{1}{V}\right)\left(\frac{[\lambda_A+\lambda_B]^V C^V e^{C(\lambda_A+\lambda_B)}}{V!}\right)\sum_{S=0}^{V} S\binom{V}{S}P_S^S(1-P_s)^{V-S}. \qquad [6.4]$$

Now, $E\left(\frac{S}{V}\right)$ is a monotonically increasing function of P_s. Thus, if one were only concerned with minimizing the number of stops during a cycle, he would minimize P_s, but this is the same as setting the light to be green *always* in the direction with the greatest flow rate. Whether a vehicle is stopped for thirty seconds or five years, it still only counts as one stop. This insight is easier to visualize in fig. 6.2 below.



The figure above shows the probability of a vehicle stopping at the intersection shown in figure 6.1 as a function of the fraction of green time alloted to direction "A" and the relative flow rate from direction "A". Notice if our objective is to minimize the expected fraction of vehicles which stop at the intersection, we would always set the light red for the lesser travelled direction.

Figure 6.2

allocated to green time in the A direction. Notice that if the fraction of flow coming from the A direction is less than 50%, the smallest probability of stopping is derived by setting the fraction of green time in the A direction to zero. This seems extremely naive; so, let us consider how to minimize the expected amount of time a vehicle must wait. Using the same assumptions, the result is given by:

$$E\left(\frac{\text{Waiting Time}}{V}\right)=\sum_{v=1}^{\infty}\left(\frac{1}{V}\right)\left(\frac{[\lambda_A+\lambda_B]^V C^V e^{C(\lambda_A+\lambda_B)}}{V!}\right)\sum_{S=0}^{V}\sum_{i=0}^{S}\binom{S}{i}\frac{\lambda_A^i\lambda_B^{S-i}}{(\lambda_A+\lambda_B)^S}\left[\frac{G_B i+G_A(S-i)}{2}\right]$$

Equation 6.5

The minimum in this case occurs when $\frac{\lambda_A}{\lambda_A+\lambda_B}=\frac{G_A}{C}$. [6.6]

In the literature, this method is called *proportionality*. (Later in this chapter, we will use proportionality to bring candidate splits up to the system cycle time.) This is not completely orthogonal to our first result with respect to average number of stops as higher values of traffic flow in a particular direction tend to get more green time in that direction. Now that we have a better understanding of the relationships between cycle time, splits and our measures of effectiveness, it would be a good idea for us to look at some currently used methods of determining the splits and cycle time used in practice.

SECTION 6.1.1 Methods currently used in Practice to set Splits and Cycle Time

Perhaps the most widely used methods were designed by Webster. Webster examined an isolated intersection with Poisson arrivals and obtained his famous result stated as equation 2.2 in this thesis. Namely,

$$C_0 = \frac{1.5L + 5}{1 - Y}$$

C_0 = Optimal cycle time to minimize average delay
L = Lost time (e.g. amber time + acceleration / deceleration time) [6.7]
Y = Practical correction factor =.9-.0075L .

There are several methods in practice for determining split times at specific intersections. First, proportionality, equation 6.6, is used, and its basis is in an isolated intersection with Poisson arrivals in both directions. The second method is based on the same model, but the objective is to give both directions an equal probability of clearing out all vehicle waiting at the intersection.

Note in all these cases, the network nature of the model is essentially ignored, and this is most especially the case with the cycle time determination.

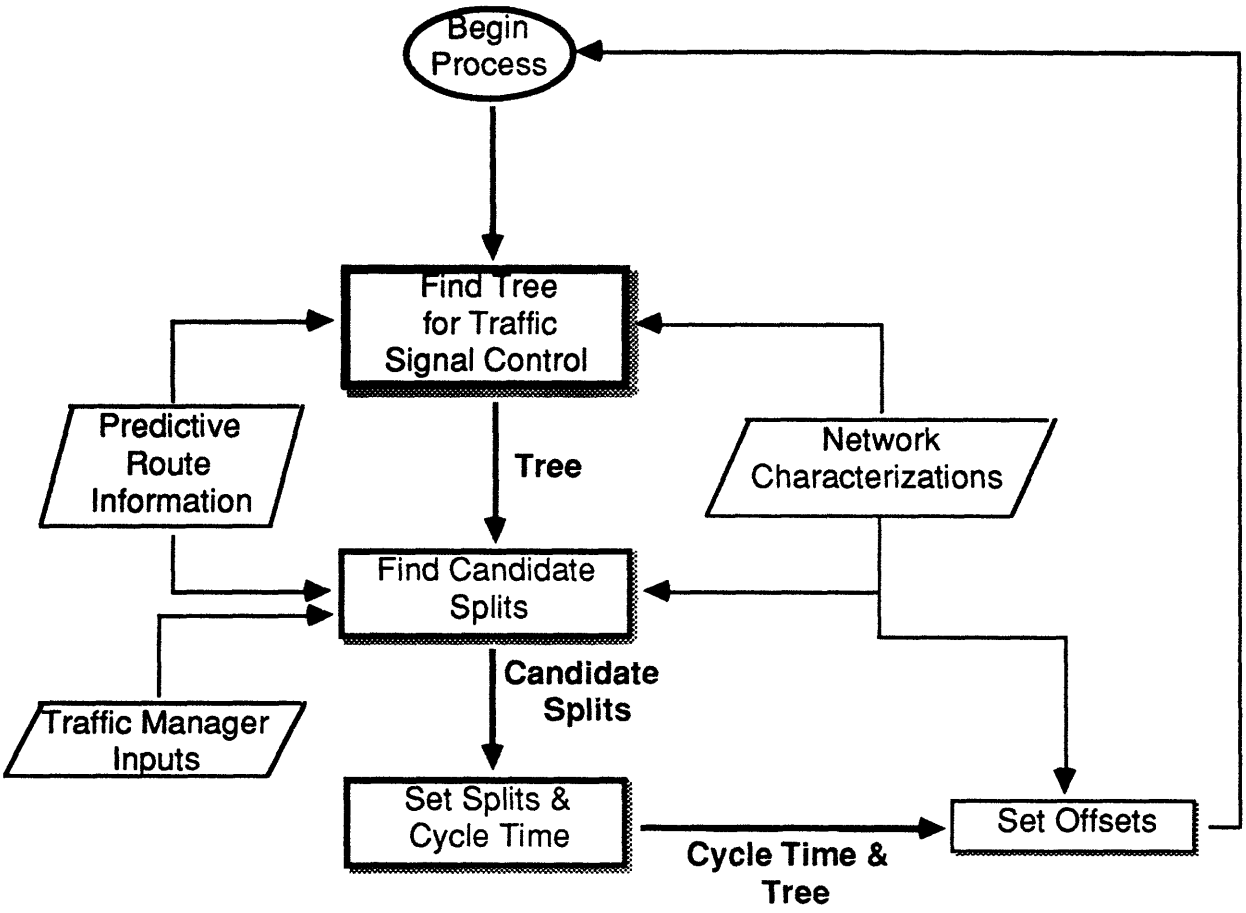
There have been many methods which look at the entire network in determining the settings for individual intersections. Most of these methods

were dealt with extensively in chapter two, but as a refresher consider the following example from the literature. In 1966 Little looked at maximizing the bandwidth for a network (Little, 1966). The next significant traffic signal control which did not use bandwidth as an optimization criteria was TRANSYT, developed by the Transport and Road Research Laboratory, United Kingdom (Robertson, 1969). TRANSYT based its traffic control settings for the network on minimizing a "disutility" function. In particular, Robertson forwarded the notion of minimizing a weighted linear function of stops across the network as well as total delay. The Texas Transportation Institute heralded the next milestone in traffic signal control in 1980 with PASSER, and PASSER was updated in 1984 to become PASSER-II (Texas Transportation Institute, 1984). PASSER-II uses a user supplied set of routes and maximizes the progression across these routes. This is more closely related to MAXBAND than TRANSYT as maximizing bandwidths implicitly improves progression ratings for the same routes. PASSER-II had several advantages over predecessors in that it directly considered: street segment capacities, turning movements, distances, nominal speeds and queue clearance intervals. We saw the first microcomputer application of real-time control with the advent of SCOOT (Robertson, 1991). SCOOT uses a heuristic search method to minimize a disutility function as with TRANSYT. SCOOT samples traffic sensors every four seconds and updates traffic signal control every five minutes. Several recently proposed methods for utilizing real-time control for traffic signal control employ dynamic programming, and the most widely used of these is OPAC, developed by Nathan Gartner (1983), University of Massachusetts, Lowell. Lan, Messer, Chaudary and Chang (1992) presented a methodology called COMBAND which was a compromise approach to setting traffic signal controls. COMBAND combined the notion of maximizing bandwidth with minimizing delay. The method is a

multiobjective linear programming technique. Linear programming has been in use in the area of traffic signal control since the mid-sixties, but the study of graph theory has only recently emerged as a driving force in the field. Two recent articles have addressed the use of graph theory in traffic management directly. The first of these, "Graph Theory and Traffic Management: A Review of Recent Progress and some Potential Applications" (Wright, et.al., 1989) examined the use of graph theory to design networks to cut down on the number of traffic crossing movements which occurred. Research (Holroyd and Miller, 1966) has shown a direct correlation between the number of traffic crossings and accident rates in urban areas. A recent US survey placed the total percentage of vehicle miles driven in the US at speeds lower than 35 M.P.H. to be over 65% (IVHS, 93). That is quite surprising when one considers the millions of intercity motor vehicle miles accumulated on an annual basis. This survey would tend to lend credence to Holroyd and Miller's work on designing *urban* networks. The other work, "Maximal Direct Covering Tree Problems" (Hutson and ReVelle, 1989), examined the question of how to select maximal trees given a cost function that did not require all nodes in a structure to be connected. Traditional approaches to maximal and minimal tree problems in graph theory literature assume 100% coverage of nodes in the graph. Hutson and ReVelle dismiss this constraint and formulate several models for sub-graph coverage. The applications to traffic signal control would be direct if suitable objectives could be derived. Hutson and ReVelle make no pretenses about the solubility of their approaches. In due course, this discussion has led us to methods for setting splits and cycle times using origin/destination and predictive routing information.

SECTION 6.1.2 Current Methods which Integrate Predictive Route Information

We assume that the models presented in this thesis receive predictive routing information from ATMS and ATIS. The information could be extracted from a model such as the one presented in "Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highway Systems" (Kaufman and Smith, 1993). Previous approaches to traffic signal control did not have this information available to them. This thesis assumes that this vital information is available from the start of the process. Recall figure 1.2.



Information flow scheme for PRISTINE

Figure 6.3

The only work in the literature which considers the use of predictive routing information in traffic signal control is MOTION (Ploss, G., Phillips, P., et.al., 1990) which was presented in a conceptual form at the transportation conference at Yokohama, Japan in 1990. Those earlier concepts were released with more details by Busch (1993). Much like this thesis' approach, MOTION sets the splits and cycle time independently of the offsets. In MOTION the offsets are based on traffic manager's input as to which routes should be synchronized. The splits are set based on the global cycle time and a unpublished queuing network approximation which is heuristic in nature.

Now that we have had the opportunity to examine the traditional and current approaches for traffic signal control, we can forge ahead and more fully understand the methodologies described in the remainder of this chapter.

SECTION 6.2 Predictive Routing Information Signal Timing INtEgration (PRISTINE) Method for setting Traffic Signal Controls

In this section, we will describe methods for setting the splits, cycle time and offsets for the network. Essentially we will calculate those quantities using the following inputs: street or network geometry, the nominal speeds or speed limits on each of the street segments, the predictive routing information, the minimum acceptable green time for any street, the minimum acceptable green time for a major street segment or arterial, the maximum acceptable average wait for a motorist at a given intersection, and the maximum acceptable cycle length.

First, we should understand the sources of these inputs. The street geometry is a characteristic of the physical layout of the network. Likewise, we assume that speed limits are unchanging characteristics of the network (or at least external to signal setting process). The predictive routing information which includes both routings and projected usage's of those routings is an output of the interaction between the ATMS and ATIS subsystems of IVHS. The maximum acceptable average wait for a vehicle at an intersection and the total

maximum allowable cycle time are cultural or regional quantities. As mentioned in the first chapter of this document, the maximum acceptable cycle time in the US and Canada is widely regarded as being between 120 and 150 seconds whereas in Southeast Asia, cycle times of over five minutes are not uncommon. The maximum wait is clearly in the realm of the traffic manager. The minimum amount of green time for the two types of street segments entering an intersection, priority and non-priority, could come from various sources. The traffic manager is one possible source, but if there is a sufficiently sensitive sensor array at work in the network, the information could be derived in real time from existing traffic conditions.

The idea behind the model is to first determine which of the street segments are *priority* street segments based on the predictive routing information. This information will be passed to the remainder of the signal setting procedure in the from of the spanning tree, S. Then based on the physical characteristics of the network, S and the various traffic manager inputs, the splits and cycle time are generated. Last, the offsets are calculated based on the cycle time, S and the physical characteristics of the network.

Now that we have the required inputs, we can describe the methods for setting the offsets, cycle time and splits for **G(N,A)**. We will present two methodologies in this document. The first will be a non-linear, multi-objective program which will solve for cycle time and splits and find an optimal solution for our model. The second model will exploit certain characteristics of the model and provide a heuristic technique which can be solved in a small fraction of the

time it takes to solve the non-linear program. Both method use the same procedure to set the offsets.

SECTION 6.2.1 Setting the Offsets

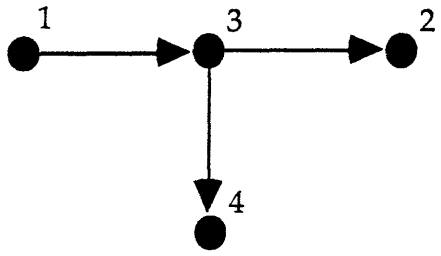
As mentioned above, the first step in this process will be to find the set of priority street segments or the set of street segments which will be coordinated. In general, it is not possible to synchronize all of the traffic signals for every direction or set of streets in the network. So, we must select the most vital street segments to coordinate based on the predictive routing information we receive from IVHS. To do this, we use one of the three heuristics described in chapter four: Route Augmentation (RA), Maximal Spanning Tree (MST) or Potential Function (PF). These heuristics received the **R** and Λ matrices from the Advanced Traffic Management System (ATMS) in coordination with the Advanced Traveler Information System (ATIS). Recall from chapter three that **R** represents the set of routes through **G(N,A)**, and specifically,

$$\mathbf{R}_i(\mathbf{j}) \equiv \text{the } j^{th} \text{ node on route "i".}$$

The desired usage's (vehicles/time) of the various routes are contained in Λ where

$$\Lambda \ni \lambda_i \equiv \text{desired usage of route } \mathbf{R}_i \in \mathbf{R}.$$

These matrices are the inputs to the heuristics in chapter four along with **D** which is a matrix whose elements correspond to the distances between nodes. Consider the simple urban street complex shown below.



A={(1->2),(2->3),(2->4)}
N={1,2,3,4}
m=3
n=4

A simple network

Figure 6.4

Let **V** be the matrix whose elements $\mathbf{V_{ij}}$ represent the speeds at which vehicles traverse the arcs $(\mathbf{i} \rightarrow \mathbf{j}) \in \mathbf{A}$. Thereby, the time it takes a vehicle to go from node **i** to node **j** along arc $(\mathbf{i} \rightarrow \mathbf{j})$ is given by:

$$\hat{t}_{ij} \equiv \text{time from "i" to "j" along arc } (i \rightarrow j) = \left.\frac{d_{ij}}{v_{ij}}\right| \begin{matrix} (i \rightarrow j) \in A \\ v_{ij} \neq \emptyset \end{matrix} .$$

The heuristics produce **S**, a tree containing the set of one-way arcs which will be coordinated by setting the offsets to allow progression along this directed, spanning tree. Recall from chapter four that we will use **S** as our arterial for purposes of computing progression as a measure of effectiveness, and our use of **S** to coordinate the offsets ensures that the traffic signal control plan is fundamentally grounded in a progression based system. We can arbitrarily select a leaf node, a node with only one arc connecting it to the rest of the tree, from **S** and set the offsets by "climbing" the tree. Consider once again fig. 6.3. Assume the cycle time is **C** and all the splits are **C/2** in length. Let the speed limits through **G(N,A)** be uniformly equal to one for this example. Let **S={(1->2), (2->3), (2->4)}**. Select node one as the root node in this case. Now, we have the following relations:

$$\theta_{21} = 0$$
$$\theta_{12} = \theta_{21} + \hat{t}_{12}$$
$$\theta_{23} = \theta_{12} + \hat{t}_{23}$$
$$\theta_{24} = \theta_{12} + \hat{t}_{24}.$$

Now, if arc **(2->4)** had been replaced with arc **(4->2)**, we would have just done a sign reversal and obtained: $\theta_{24} = \theta_{42} - \hat{t}_{24} = \theta_{12} + C/2 - \hat{t}_{24}$. Note, we actually need to know the exact splits and cycle-time to set the offsets. So, we really calculate **S** first and then come back and figure out the offsets after calculating the splits and cycle time.

SECTION 6.2.2 Setting the Splits and Cycle Time

In this section, we will examine two methods of setting the splits and cycle time. One method, a math program, solves to optimality, and the second method, a heuristic exploits certain characteristics of the system and solves for the splits and cycle time in a fraction of the time required for the non-linear, multi-objective program. The best place to start in determining the splits and cycle time is to examine our inputs.

SECTION 6.2.2.1 Inputs for Split and Cycle Time Determination

The inputs to this section of the model are as follows. **C** is our system cycle time which will be set by the process either math program or heuristic. So, it is a *decision variable* at this point. C_{max} is the maximum allowable cycle time; this is an input to the model by the traffic manager. This prevents system "optimal" cycle times of 2 hours for example. **b(x)** is the probability of having **x** vehicles in a platoon in **G(N,A)**; it is an input to the model. It could be gathered in real time, or it could be surmised based on historical data. β is the weighting of stops per vehicle in the objective function, and γ is the weighting of the wait per vehicle in the objective function. Note: $\gamma + \beta \leq 1$ **and** $\beta, \gamma > 0$. The street segment going one-way originating at intersection "**i**" and ending at intersection "**j**" is called **(i->j)**. s_{ij} is the split (green time) in seconds for street segment

(i->j) at intersection "j". The total flow from all routes along **(i->j)** is λ_{ij}. **S** is the spanning tree generated by one of the techniques from chapter four. $\mathbf{Arterial_{min}}$ is the minimum allowable green split for an arterial, and $\mathbf{Network_{min}}$ is the minimum allowable green split across the entire network. Both of these are inputs to the model, set by the traffic manager. The *psi* function is a Boolean measure of alignment in the network.

$$\Psi[(i \rightarrow j),(k \rightarrow j)] = \begin{cases} \textbf{.true. if } (i \rightarrow j) \textbf{ and } (k \rightarrow j) \textbf{ share a common green cycle} \\ \textbf{.false. else ...} \end{cases}$$

$\bar{x}_1$ is the average amount of time it takes for a vehicle to safely cross an intersection.

SECTION 6.2.2.2 Non-Linear Program (NLP) for Determining Splits/Cycle Time

In this section, we will present a multi-objective, non-linear math program to solve for the optimal splits and cycle time. The traffic manager will need to specify the weighting appropriate to stops in the network and average vehicle wait. As stated earlier, progression is automatically considered in the traffic signal timing plan through our use of the spanning tree to set the offsets. Before we launch directly into the description of the math program, it would be useful to make some additional computations and definitions.

We will begin with the relation $X(s_{ij}) \hat{=} \left\lfloor \frac{s_{ij}}{\bar{x}_1} \right\rfloor$ is a measure of how many vehicles can be safely accommodated during split s_{ij}. $W(s_{ij},y)$ is the probability of waiting for "y" red lights on an arterial at intersection "j" with split s_{ij}. We will a series of measurement variables. The first of these is the average wait at intersection "j" for priority street segment (i->j) where s_{kj} is the red light period which is defined as $W_{ij}^* \hat{=} \sum_{y=0}^{\infty} j \bullet s_{kj} \bullet W(s_{ij},y)$. Then, the weighted sum of these which is directly proportional to the expected average wait per unit time is equal to: $W_j^* = \lambda_{ij} W_{ij}^* + \lambda_{kj} W_{kj}^*$ where $(i \rightarrow j),(k \rightarrow j) \in S$. In general, we do not need to know everything about b(x) for the non-priority streets in the QEM; in fact, we need to know two items the variance of b and the mean value of b. We define a special variable to store the important quantity, the variance of b divided by double the mean value of b, which we will call B. The expected wait per vehicle along non-priority roads is taken directly from the QEM, and we have:

$$\overline{W}_{kj} \hat{=} \frac{s_{ij}^2 s_{kj} + s_{kj}^2 \lambda_{kj} \bar{x}_1^2}{2C(s_{kj} - C\bar{x}_1 \lambda_{kj})} + \frac{B \bullet C \bullet \bar{x}_1}{1 - \bar{x}_1 \lambda_{kj}} \text{ where}$$

$\Psi[(i \rightarrow j),(k \rightarrow j)] =$ .false., $(i \rightarrow j) \in S$ and $(k \rightarrow j) \notin S$. This leads us to our total for the non-priority street segments at intersection "j", or $\overline{W}_j = \sum_{\forall k \text{ s.t. } (k \rightarrow j) \notin S} \lambda_{kj} \overline{W}_{kj}$.

Thus, our total expected delay per unit time at j would be characterized as $\hat{W}_j = W_j^* + \overline{W}_j$. Now, moving onto the number of expected stops in the network, we begin by using the QEM result and state:

$$\overline{T}_{kj} \hat{=} \frac{s_{ij}^2 s_{kj} \lambda_{kj} + s_{kj}^2 \lambda_{kj}^2 \overline{x}_1^2 + B \bullet C \bullet s_{kj} + C^2 B \overline{x}_1 \lambda_{kj}}{2C \bullet s_{kj}(s_{kj} - C\overline{x}_1 \lambda_{kj})} \text{ where as before}$$

$\Psi[(i \rightarrow j),(k \rightarrow j)] =.\text{false.}$, $(i \rightarrow j) \in S$ and $(k \rightarrow j) \notin S$. For the priority routes, we have a more straightforward expression for the expected number of stops which is: . $T_{ij}^* = \sum_{y=0}^{\infty} yW(s_{ij}, y)$. Then we have similar definitions for stops as we had for average wait; specifically we have:

$T_j^* = \lambda_{ij} T_{ij}^* + \lambda_{kj} T_{kj}^*$ where $(i \rightarrow j),(k \rightarrow j) \in S$, $\overline{T}_j = \sum_{\forall k \text{ s.t. } (k \rightarrow j) \notin S} \lambda_{kj} \overline{T}_{kj}$, and $\hat{T}_j = T_j^* + \overline{T}_j$.

Now, we can write our entire math program in a notationally compact form as:

$$\textbf{Minimize} \quad \beta \sum_{j=1}^{n} \hat{T}_j + \gamma \sum_{j=1}^{n} \hat{W}_j$$

subject to

$s_{ij} + s_{kj} = C \ \forall \ i,j,k$ where $\Psi[(i \rightarrow j),(k \rightarrow j)] =.\text{false.}$

$s_{ij} > C\overline{x}_1 \lambda_{ij} \ \forall \ (i \rightarrow j)$

$s_{ij} > \text{Network}_{min} \ \forall \ (i \rightarrow j)$

$s_{ij} > \text{Arterial}_{min} \ \forall \ (i \rightarrow j) \in S$

$0 < C \leq C_{max}$

$s_{ij} \geq 0 \ \forall \ (i \rightarrow j)$

Math Program 6.1

Math program 6.1 solves optimally for the various splits and the cycle time, but it could be a time consuming process. The dimensionality of the decision variables is **2n+1** where **n** is the number of intersections in the network. Is there a way of taking advantage of the *structure* of the network and derive a solution without having to resort to a large, non-linear math program? This was the motivation for developing the heuristic presented in the next section.

SECTION 6.2.2.3 Split setting Heuristic (SH)

Recall that the time inputs to our system were the minimum acceptable green split for arterials, i.e. elements of S, minimum acceptable green split across the network, and maximum acceptable cycle time. Let us develop an intuition for these limits before going into the mechanics of setting the splits and cycle time in the heuristic. As stated in chapters one and three, the maximum acceptable average wait at a traffic light is a social factor far more than a characterization of the network or the vehicles and their routings. For example, it may be far more acceptable for "social utility" to have everyone in the network spend an extra minute traveling through the city and limit all the traffic signals to a maximum cycle of 120 seconds than to have a cycle length of five minutes. So, to some extent we need to consider the impact of each traffic light as well as aggregate totals or averages. It may be totally reasonable to spend an average delay of 45 seconds at a particular light, but no matter how short the remainder of the trip takes, it would not be reasonable to expect the average motorist in New York City to spend five minutes at a particular light. The concept behind the minimum acceptable green splits is straight forward. The global minimum acceptable green split across the network is the minimum time that is reasonable

for a green period to last. This could be a time varying quantity. For example, one would never want this quantity to fall below the average amount of time it takes for one vehicle to safely accelerate and cross the intersection, but at 3 AM under conditions of extremely light traffic, the value might reasonably be very close to this value. On the other hand, if the roads are at half capacity or more, one might specify the value to be fifteen seconds, e.g. long enough for five average vehicles to pass through the intersection for example. This quantity, which is more or less a safety and convenience factor, is different than the minimum acceptable green split on the arterials or coordinated arcs. Here the issue is one of accommodating the platoons in **G(N,A)**.

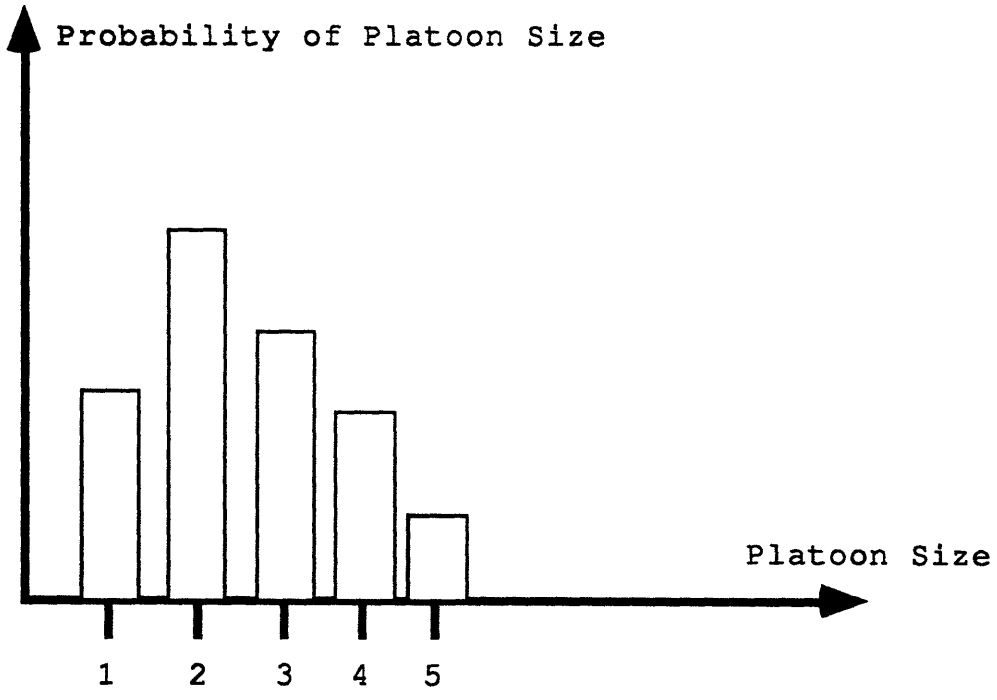
The concept of designing a traffic signal control plan around the idea of getting the majority of the platoon to clear the intersection underlies many of the models in use today. There are good reasons for this. Every model is an approximation to the real world. Drivers accelerate and decelerate at differing rates. Likewise, they maintain differing following intervals, and all these factors change base on the lighting, the road conditions and the level of congestion. By seeking to accommodate a number of vehicles during one green light, one implicitly allows for the nuances of human behavior which could never adequately be described in a model. All progression based systems implicitly use this concept whether explicitly maximizing progression or more subtly using bandwidth maximization. In Matson, Smith and Hurd's (1955) classic Traffic Engineering text, the method prescribed for calculating splits and cycle time is in its purest form *given a traffic density of X, what are the splits and cycle time required at a particular intersection to accommodate a minimum of Y percentage of the vehicles*

which arrive during a given cycle? Traffic Engineering listed two formulae based on empirical research relating offsets, green splits and cycle time:

$$\text{Progression Speed(M.P.H.)} = \frac{.68 * \text{Distance between Intersections (feet)}}{\text{Fraction of Platoon to Accommodate}}$$

$$\text{Volume(vehicles / hour)} = \frac{3{,}600 * \text{Bandwidth(seconds)}}{(\text{Cycle Time[seconds]})(\text{Average Headway between Vehicles [feet]})}$$

We specified that this minimum green split time for the arterials may come from the traffic manager, presumably based on knowledge of the system, but if we use information gathered from the network, our overall intuition is the same as Matson, Smith and Hurd's. Imagine that we knew the distribution of platoon sizes for the network, based on historical or observed data to be as shown below.



The probability mass function for platoon size in the network

Figure 6.5

We might set our criteria for the smallest acceptable green split for the arterial to be sufficient to allow 80% of platoons *reaching the intersection as the light turns green* in that direction to pass through an intersection unhindered. In the case shown in fig. 6.4, this might require us to allow 4 vehicles to be able to pass through which would put the lower bound at 3*4=12 seconds where 3 seconds represents the average amount of time it takes a vehicle to accelerate and cross the intersection. If we wanted the split to accommodate 95% of all platoons that approached the intersection, we would have to allow five vehicles to pass which would require 3*5=15 seconds as a minimum split time for an arterial. There are several items to consider at this point. Platoon cohesion is a very transient phenomenon at best, and the larger the platoon, the more likely it is to disperse. Baass and Lefebvre (1990) considered this process in detail in their paper "Analysis of Platoon Dispersion with Respect to Traffic Volume". So that even if a longer split upstream might favor longer platoons, it is unlikely that such an event would propagate throughout the network. We must also recognize that there may be efficiency benefits accrued by breaking up large platoons as we will see in the next section.

Eddie (1967) pointed out in his paper with Bavarez that in the Holland Tunnel in New York City, the traffic authority purposely introduced breaks in large platoons which resulted in a better overall travel rate through the tunnel. Herman and Rothery (1967) explained why this would be true using a fluid dynamic approximation for traffic dynamics. The idea was that a small perturbation in a platoon could be augmented by other perturbations whereas a break in-between elements cause them to behave independently. Herman experimentally verified his model and demonstrated a case where a platoon of

eleven vehicles had to stop four times in a two mile interval while a group of three vehicles experienced no stops under nearly identical conditions. Thus, to save time overall, the goal should not be to accommodate every platoon but rather to accommodate some large fraction of the potential platoons passing through the intersection. Now we have lower bounds on the acceptable green splits and an upper bound on total cycle time. We are ready to set the splits and cycle time.

We will use the QEM model developed in chapter five to aid in setting splits. Recall inequality 5.9, shown below

$$\overline{W}_S \le \frac{G_A^2 + G_B \bar{x}_1^2 \lambda_1}{2(G_A + G_B)} + \bar{x}_1 * \left[\overline{W}_S \lambda_1 + \frac{\text{Variance(b)}}{2\bar{b}} \right] + \frac{G_A}{G_B} \left(\bar{x}_1 * \left[\overline{W}_S \lambda_1 + \frac{\text{Variance(b)}}{2\bar{b}} \right] \right).$$

This can be rewritten as a quadratic in terms of the green split in the uncoordinated direction. If we specify a maximum acceptable average wait in the non-coordinated direction, we can solve for the minimum acceptable green split for the uncoordinated direction as long as the consistency equation

$\overline{W}_S > \dfrac{\bar{x}_1 \left(\dfrac{\text{Variance(b)}}{\bar{b}} + \lambda_1 \bar{x}_1 \right)}{(1 - \lambda_1 \bar{x}_1)}$ is met. If this condition is not met it means that the intersection is over saturated. Under conditions of over saturation, the best approach is simply to increase the network flow capacity as much as possible and attempt to allow the excess traffic to bleed off as quickly as possible.

By definition **S** must touch every node; so, we can generate a candidate set of splits (and hence cycle times which are just the sum of the splits) for every intersection. Then we take the maximum at all the candidate splits as the cycle time for **G(N,A)**. Intersections with shorter candidate cycle times have their candidate splits increased proportional to the volume of traffic flowing into the intersection from that direction. This is essentially a restatement of the Pignataro method (Gerber and Hoel, 1988). The primary difference is that Pignataro used a measurement of vehicle density over the proceeding fifteen minute interval to determine the minimum green splits, and we use the QEM model to arrive at the minimum acceptable green splits. One might ask *why it is better to increase the splits rather than normalize to some intermediate value.* First, consider the coordinated direction. Every increase in the green time tends to allow more of the potential platoons to pass through the intersection unimpeded. Additionally, in the uncoordinated direction, *proportional* increases in the green splits for the coordinated and uncoordinated direction have no effect on the chance of stopping if the conditions below are met.

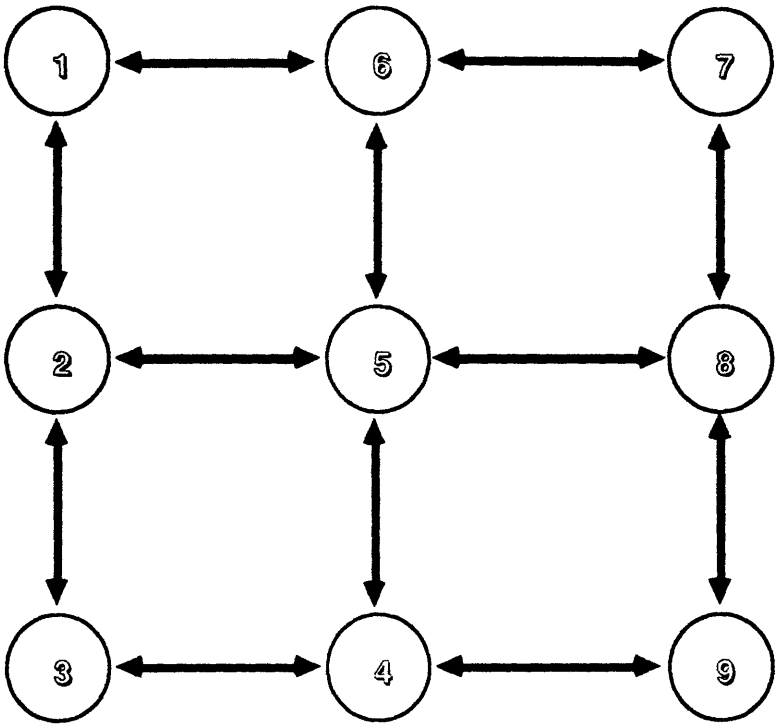
$$G_A > \bar{x}_1$$
$$G_B > \bar{x}_1$$
$$\lambda_A \bar{x}_1 (G_A + G_B) < G_A$$
$$\lambda_B \bar{x}_1 (G_A + G_B) < G_B$$

The first two conditions should be met by an appropriate choice of a lower bound for the network green split as the requirement is for the green split to exceed the average time it takes for one vehicle to accelerate and cross the intersection, hardly an unreasonable condition. The second two conditions are

system queue stability conditions which are true irrespective of the vehicle arrival process, and namely, they require the amount of green time per cycle in a specific direction to exceed the average amount of time it would take for the vehicles arriving during the cycle time to clear the intersection. Additionally, this method tends to give a high priority to the busiest intersection; the intersection with the highest candidate splits will be the intersection with the highest volume of traffic. There are two primary reasons for giving priority to the busiest intersection. First, the busiest intersection does not exist in isolation. The traffic arriving from this network must arrive from somewhere, and presumably this traffic would come from within the network. So, the intersections surrounding the busiest intersection would tend to have candidate splits nearly as long as the highest set. Second, by favoring the busiest intersections, we are favoring larger volumes of traffic. When there is little flow on the network, nearly any traffic strategy can perform well, but as congestion continues to increase, the real benefits are accrued from systems able to anticipate and adapt to these conditions.

We will now illustrate this technique using the same network we used in chapter four.

The base network is shown below.



An illustrative 3x3 network

Figure 6.6

For this example, we used the potential function (PF) technique to determine the tree for the offsets. The results were calculated by an automated version of the SH and then verified manually. The following information is available to the model: the network geometry (shown above), the predictive route information (shown below in table 1), the minimum acceptable green time at any intersection (15 secs), the minimum acceptable green time for a major roadway at an intersection (30 secs), the average amount of time it takes a vehicle to clear an intersection (2 secs), the acceptable wait for an average vehicle to wait at any given intersection (30 secs), and the coefficient of variation for the platoon size is 1. All street segments are 308 feet long, and the speed on all street segments is 30 MPH. The tree, S, consists of the following arcs:

Arc (In order of selection)	Value
9->4	4.9
4->3	5
3->2	7.5
2->1	10
8->9	12.5
7->8	15
4->5	4.8
5->6	9.2

The arcs are selected in the order shown from top to bottom using the Potential Function Heuristic. The value shown on the right is a relative measure for how beneficial the addition of the arc to the spanning tree will be.

Table 6.2

Now that the tree is determined, we can derive the candidate splits. These are displayed by intersection.

CANDIDATE SPLITS/CYCLE TIMES

Intersection	Arc	Green (secs)	Cycle Time(secs)
1	(6->1)	15	45
	(2->1)	30	
2	(5->2)	15	45
	(3->2)	30	
3	(4->3)	30	45
	(2->3)	15	
4	(3->4)	15	60
	(5->4)	30*	
	(9->4)	30*	
5	(4->5)	30	45
	(6->5)	30	
	(8->5)	15	
	(2->5)	15	
6	(5->6)	30	45
	(1->6)	15	
	(7->6)	15	
7	(6->7)	15	30
	(8->7)	15	
8	(5->8)	15*	45
	(7->8)	30*	
	(9->8)	15	
9	(8->9)	30	45
	(4->9)	15	

NOTE: because of the low vehicle occupancy rate the queuing model gave an amount of required green time that was less than the minimum specified by the traffic manager, but if the total rate along arc (7->6) had been as high as 6 vehicles per minute, it would have been 17.5 seconds, and if the rate had been 9 cars per minute it would have required over 29 seconds.

This table shows the candidate splits selected using the Split setting Heuristic (SH) method to set them.

Table 6.3

Now, the largest candidate cycle time was 60 seconds for intersection 4. Now, we will set the set the splits so that all the cycle times are equal to 60 seconds, and we get the following results.

ACTUAL SPLITS/CYCLE TIME

Intersection	Arc	Green (secs)	Cycle Time(secs)
1	(6->1)	22.2	60
	(2->1)	37.8	
2	(5->2)	15	60
	(3->2)	45	
3	(4->3)	45	60
	(2->3)	15	
4	(3->4)	30	60
	(5->4)	30	
	(9->4)	30	
5	(4->5)	40.3	60
	(6->5)	40.3	
	(8->5)	19.7	
	(2->5)	19.7	
6	(5->6)	39.8	60
	(1->6)	20.2	
	(7->6)	20.2	
7	(6->7)	30	60
	(8->7)	30	
8	(5->8)	15	60
	(7->8)	45	
	(9->8)	15	
9	(8->9)	45	60
	(4->9)	15	

This table shows the actual splits and cycle time selected after the second pass.

Table 6.4

The offsets would be selected so that the vehicles traveling along the tree shown in figure 1 (in green) would pass through green lights after entering the tree.

The cycle time is 60 seconds, and at 30 MPH it takes 7 seconds to go 308 feet. We can arbitrarily select any "leaf" on the tree as our base node. So, we will select node 7 as the base node. So, our offset for the green light going from starting at seven and heading toward 8 (or equivalently the green light one encounters going from eight to seven), Θ_{87}, is zero. Note that the offset for Θ_{67} is 30, based on the splits. Now Θ_{78} is 7 while Θ_{58} is 52. The compiled offsets for the network are shown below.

OFFSETS

Intersection	Arc	Green (secs)	Offset (secs)
1	(6->1)	22.2	19.8
	(2->1)	37.8	42
2	(5->2)	15	20
	(3->2)	45	35
3	(4->3)	45	28
	(2->3)	15	13
4	(3->4)	30	21
	(5->4)	30	51
	(9->4)	30	21
5	(4->5)	40.3	58
	(6->5)	40.3	58
	(8->5)	19.7	38.3
	(2->5)	19.7	38.3
6	(5->6)	39.8	5
	(1->6)	20.2	44.8
	(7->6)	20.2	44.8
7	(6->7)	30	30

	(8->7)	30	0
8	(5->8)	15	52
	(7->8)	45	7
	(9->8)	45	7
9	(8->9)	45	14
	(4->9)	15	59

This table shows the offsets selected using the PF method.

Table 6.5

The measures of effectiveness for the PF method are as follows. Note, the following table ignores the effects of the starting light. For example, a vehicle approaching intersection 7 from the north would have a 50% chance of stopping based on random incidence, and the expected delay then would be 15 seconds. We will assume the vehicle begins its voyage just as the light turns green in the vehicle's direction of entry for the net. We also assume light traffic conditions.

Route	Desired Usage	Expected Stops	Expected Delay
7->8->9->4->3->2->1	2.5	0	0
6->5->4	3	0	0
7->6->1	2.3	1	13.8
9->4->5->6	2.4	0	0
2->5->6	2	1	12.7

The MOE's for the Traffic Signal Control plan are displayed int he table above. The MOE's were calculated using an expected value approach.

Table 6.6

Thus, the expected number of stops across all vehicles for the network is .352, and the expected delay per vehicle across the network is 4.7 seconds.

SECTION 6.2.3 Strategy's Strengths and Weaknesses

This section will examine in a very general sense when the use of predictive routing information and the strategy outlined above would prove beneficial.

There are two primary cases when the PRISTINE traffic signal setting strategy would prove very effective. Both cases involve the edge that the strategy receives from the predictive routing information. If there is an accident or other incident that blocks a street, the ATMS and ATIS systems will provide that information to both vehicle operators and the traffic control systems. The PRISTINE strategy could free more green time along avenues leading away from

and around the accident scene as motorists are rerouted by IVHS. Current systems could not react to this change in the roadway pattern until congestion had occurred on the streets surrounding the accident. The second example is very similar. If there is a predictable surge in the traffic demand, by opening up, i.e. allowing more green time, the routes which will be hardest hit in advance, the PRISTINE strategy will prevent congestion later.

Just as there are times when the PRISTINE strategy will do very well, there are also situations where the effect will be negligible. In situations where there are no clearly defined flows of traffic, the PRISTINE strategy will not do any better than existing systems. The heart of the PRISTINE advantage is in being able to exploit patterns in traffic and synchronize the appropriate street segments accordingly.

Although we specifically eliminated the heavy traffic situations, i.e. situations where the congestion is no longer a function of traffic signal control so much as a capacity issue, the PRISTINE strategy could be quite useful in these situations as well. When a sports stadium lets out its glut into the surrounding roadways, there is a temporary lack of required capacity. By using the tree finding and offset determination heuristics from chapter four, we could identify routes of egress from the congestion center and give priority to these routes. Although there will still be a large traffic jam, the effects may be mitigated by such an approach. In these cases, we would set the cycle time to the allowable maximum and then break down the splits by straight proportion.

The entire PRISTINE strategy would be repeated every five to ten minutes ideally. A lot depends on how accurate the data bases which ultimately are used in IVHS are and how frequently they are updated. If the data bases are updated only every twenty minutes then it would be unreasonable to update the PRISTINE strategy any sooner than every twenty minutes. Now that we have designed a TSC strategy, how do we evaluate it? The answer lies in the next chapter.

Chapter Seven

Simulation and Heuristic Testing

SECTION 7.1 Simulation

In the previous chapters, we established the theoretical basis for the Predictive Routing Information Signal Timing INtEgration (PRISTINE) model. Chapter three established the basic mathematical description of the problem. Chapter four describes the use of spanning trees in setting offsets in a Traffic Signal Control (TSC) plan. Chapter five developed the Queue Effects Model (QEM) which is the heart of PRISTINE for setting the splits on non-tree street segments, and chapter six combined the models from the earlier chapters into one control package, PRISTINE. In this chapter we will test our concepts using a traffic simulation motivated by real world traffic data.

The Predictive Routing Information (PRI) generated under IVHS and required by PRISTINE, is not yet available in practice. By PRI we are referring to the desired usage rates described in chapter three. IVHS will provide us with the anticipated average arrival rate of vehicles, over a fixed period of time, into the network delineated by route. (In this thesis we used ten minutes to represent a reasonable "fixed" period of time.) (See appendix C for an example of predictive routing information used in the simulation.)

It is not feasible to test PRISTINE with real world traffic at this point; we will turn to simulation as a means of evaluating the performance of our traffic signal control strategies. We have two primary tasks in this chapter. Recall from chapter six that we developed *two* methodologies for setting the splits. One was a Split setting Heuristic (SH) technique, and the other involved the solution of a Non-Linear Program (NLP) to determine the splits. Our first goal is to use simulation techniques to determine which method, NLP or SH, performs better as the split and cycle time setting engine under PRISTINE.

Our second goal in this chapter is to use simulation techniques to test our hypotheses concerning the value of predictive routing information (as used in PRISTINE) in setting traffic signals. We have described our rationale for believing the use of PRI would offer advantages over existing traffic signal control methods in chapters 1, 2 and 6, but we should test PRISTINE against a methodology which actually uses *existing information*. In this chapter we test PRISTINE against an idealized Third Generation Control (3GC) system; we will further describe 3GC in section 8.1 and appendix A.

However, before we can make comparisons, we need to construct an appropriate traffic simulation model. We will take up this task in sections 7.1.1 through 7.1.3.

SECTION 7.1.1 Evaluating Traffic Simulation Models

In this section, we will address two issues. The first will be what aspects of the physical world must be incorporated into a simulation if it is to be useful for our tasks of comparing SH vs. NLP and PRISTINE vs. 3GC. For example it is important in our research for there to be a *dynamic interface* between the simulation and the traffic signal control strategy. Specifically, we are talking

about the ability of the simulation to both provide information to and receive traffic signal timing plans from the traffic control strategy while the simulation is running.

The second issue is how one would characterize a *good* traffic simulation. The two issues are related, and we will integrate them by developing our *criteria* for a good simulation as we discuss our *requirements* for the simulation. Our criteria are the standards by which we judge the value of a simulation while our requirements are those aspects that must included in the simulation for it to be useful to us.

Barlas (1989) described a method for evaluating complex simulations in an article for the European Journal of Operational Research. In his article Barlas pointed out that simulations must be evaluated in *two distinct areas,* structural validation and behavior validation. Structural validation is designed to check whether the structure of the simulation is an adequate representation of real system it models. Behavioral validation is designed to test whether the simulation produces an acceptable output behavior. For example if one were simulating the trajectories of soccer balls kicked during the World Cup playoffs then the structural tests would be things like ensuring: the Young's Moduli of the balls were correctly modeled and the equations for the force of gravity were correctly written. A behavioral test would be to ensure that the soccer balls fly farther when kicked harder. Barlas correctly asserts that the structural portion of the testing should be completed first as there is no point in determining behavioral characteristics of a simulation if the simulation is not structurally sound. Our requirements for the simulation correspond to Barlas' structural testing criteria.

It is important to understand the *tone* of Barlas' remarks as well as developing our own criteria based on his broad guidance. Barlas' techniques are designed to be applied to very complex systems (e.g. traffic simulators) where it would be literally impossible to test every conceivable situation that might arise in the course of the simulation. Barlas notes that frequently simulations are tested in non-systematic and wasteful ways, e.g. vary this parameter with the following one million settings, when the simulation could have been tested *systematically* in a much shorter and more productive period of time. In the next section, we will begin by developing our own structural criteria to test a traffic simulation.

SECTION 7.1.1.1 Structural Requirements

There are certain key aspects of the physical world we would like to be present to model traffic in the simulation. These are part of the structural requirements under Barlas' simulation evaluation scheme. A list of our structural requirements are as follows.

(1) **Reproducibility:** Vehicles should travel through the network along the street segments in a *logical* and *consistent* manner, e.g. given the *same* set of circumstances, a vehicle should take the *same* action every time the simulation is run. This is important to us, because we will be testing PRISTINE vs 3GC using the same traffic scenarios in section 8.2, and the more random interference we can filter out in the simulation design, the more accurately our simulation runs will reflect real differences in the quality of the traffic signal control plans rather than just random variation.

(2) **Platooning:** Vehicles tend to clump under real-world conditions, and these groups of vehicles are called platoons. It is critical to the logic in PRISTINE that vehicles move through the traffic network in platoons. Any simulation

designed to test the performance of SH vs NLP in PRISTINE should produce platooning.

(3) **Capacity:** The simulation should have sufficient capacity in terms of memory, addressing, etc. to model traffic flows as they begin approaching street segment capacities, because this is frequently when TSC plans reap their greatest benefits.

(4) **Congestion Effects:** Congestion must *have an impact* on the vehicles in the network. As congestion increases, traffic flow should degrade. Vehicles will be starting and stopping, and vehicle progression will become more erratic.

(5) **Reasonable Time to Run:** The simulation should run in a reasonable time frame. It would not be useful for the purpose of analysis to use a simulation that required several days of CPU time to simulate a couple of seconds of simulated time, no matter how well the simulation matched the physical properties of the system.

(6) **Vehicle Acceleration/Deceleration:** Vehicles should accelerate and decelerate in reasonable ways. It should require several seconds for a vehicle to go from a complete stop to 30 MPH.

(7) **Dynamic Interface:** Another absolutely essential feature of the simulation is that it must allow a dynamic interface with the TSC strategies it is testing. The 3GC strategy will require information about arrivals and departures on each street segment in real-time; the simulation must be able to provide this information *and* implement the generated TSC plan without stopping or restarting the simulation. Even the state of the art simulation systems do not allow this interface as we will see later in this section.

(8) **MOE Reporting:** The simulation must be able to monitor and record our Measures of Effectiveness (MOE), average stops per vehicle and average wait per vehicle.

SECTION 7.1.1.2 Behavior Validation

Behavior validation is divided into two categories, pattern prediction and structurally oriented behavior tests. Pattern prediction testing is based on the premise that since simulations are designed to test very complex stochastic

systems, it is extremely difficult if not outright impossible to accurately predict individual values of all the outputs for a given simulation run. If it were easy to do so, there would be no need to resort to simulation as a technique. On the other hand, it certainly **is possible** to predict **patterns** among several simulation runs based on changes in the parameters between the simulation runs. We will examine three such patterns.

(1) **Increase in Average Delay:** We would expect that as the traffic flow volume across the network increases for a *fixed* TSC plan, the average delay per vehicle would increase. (This is true to the point where heavy traffic conditions prevail. Once vehicles are required to wait through entire green light splits because the street segment ahead is clogged, traffic patterns tend to be very unstable.)

(2) **Variance in Individual Delay:** If we increase the variance for a stochastic arrival rate then we would expect to see a greater variance in the individual wait experienced by each car. For example, suppose we have one system where the vehicles always arrive in groups of two. Additionally, we have another system which is identical in all respects, but the vehicles arrive in groups of one or in groups of thirty-six. We can certainly set the occurrence of the two arrival sizes so that the mean is two, but in the latter case, the additional variance in the arrival rate would cause a much larger variance in the delay experienced by the individual vehicles due to congestion, interactions between vehicles, etc.

(3) **Increase in Average Transit Time:** As the network becomes more crowded we would expect to see an increase in the average transit time per vehicle.

Barlas identified trends, average values and variations as three areas to examine when testing the behavioral characteristics of simulations. *Trends* are the general relationships between changes to input parameters and simulation outputs. In particular, one is more interested in the form of the relationship rather than in a specific mathematical equation. If changes to the flow rate in the

the network caused seemingly random variations in the wait per vehicle for a fixed TSC plan, we would suspect that something untoward was going on with the simulation. The trends should be evaluated for *simple cases* where it is easy to predict the form of the relationship; then when it comes time to make decisions based on the more complex relationships generated by the simulation, the decision maker will have more confidence in the value of the information provided by the simulation.

SECTION 7.1.2 Existing Traffic Simulation Models

We described seven commonly applied traffic simulations in the literature review: ROGUS, CORFLU, UTSM, SATURN, SATCHMO, DYNASMART and TRAF-NETSIM. All of these simulations fail one or more of our structural criteria developed in section 7.1.1.1. Most of these simulations are unable to provide information to a traffic signal timing strategy and implement new traffic signal plans while the simulation ran. Therefore, they failed our criterion for dynamic interface [criterion 7.1.1.1(7)] Additionally, the remainder of the simulations were either unavailable to us or did not report the measures of effectiveness we wished to measure [criterion 7.1.1.1(8)]. Therefore, we elected to develop our own traffic simulation model (see section 7.1.3 and appendix B).

SECTION 7.1.3 The Lin-Sarkar-Staats Simulator (LS3)

Here we will briefly describe the development of the LS3 simulator and conduct the structural and behavioral tests described in section 7.1.1. (For a more technical description of LS3 see appendix B.) In section 7.1.3.1 we will trace the historical development of LS3 and see how it evolved into a form which met all of our structural criteria. We will go on to look more closely at the form of the simulation in section 7.1.3.2. Our outputs from the simulation, measures of effectiveness, are defined in section 7.1.3.3. In section 7.1.3.4, we will examine output from the simulation specifically looking for our behavioral validation patterns from section 7.1.1.2.

SECTION 7.1.3.1 Historical Development and Structural Testing

Lin (1992) set out with the objective of writing a computer simulation for traffic that would combine the best aspects of both microscopic and macroscopic traffic modeling. The important features included but were not limited to: tracking each vehicle's speed, position and turning movements, utilizing origin-destination pairs [i.e. routing], measuring overall congestion and supporting large network simulations. By May 1992, Lin had constructed a simulation which ran under Turbo-C on an MS-DOS, 386 platform which integrated criteria 7.1.1.1(1) through (6) in its design parameters. The simulation not only met but exceeded the requirement to run in a reasonable time [criterion 7.1.1.1(5)]. In fact, for a small network (e.g. 3x3 nodes), the system was able to simulate 155 seconds of time for every second of real time the simulation ran for. Although

the simulation was fast, it did have limitations. First, all of the street segments had the same speed limit, 30 MPH. Second, the simulation did not allow the signal timing plan to be modified once the simulation was running [criterion 7.1.1.1(7)]. Lin did not provide a means of either monitoring or displaying any conventional MOE; so, it also failed in the area of MOE reporting [criterion 7.1.1.1(8)].

Sarkar (1993) began working on upgrading the simulation in the Fall of 1992. Sarkar's objective was to accomplish three tasks with respect to the simulation. First, he was to allow for variable speed limits on the street segments. Second, the simulation had to be modified to allow street segments of varying lengths. Last, Sarkar needed to integrate a method of monitoring and displaying the MOE for the model, average stops per vehicle and average wait per vehicle (see appendix B for a description of these MOE's in the context of the simulation). By May 1993, Sarkar had accomplished his tasks. The simulation ran on a Macintosh(TM) Quadra 700 using Think C(TM) at a slightly slower pace, but it allowed for the representation of much more realistic street networks and traffic patterns to be simulated. But, the simulation still did not allow the traffic signal timing plan to be modified during the simulation [criterion 7.1.1.1(7)].

From Fall 1993 to Spring 1994, the simulation was transferred from C to FORTRAN 77 and installed on a DEC 5000/20 workstation. The following features were added. The simulation allows for dynamic changes in the TSC plan; the cycle length, split configuration and offsets are allowed to vary during program execution. (The network structure remains fixed throughout any

any specific simulation run.) The simulation is fully integrated with the TSC routines that support it, and information about congestion levels, vehicle position, speed, etc. can be fed directly to the TSC calculation programs. Such is required to accurately model the 3GC system. The simulation now allows for varying PRI to be entered into the system while the simulation continues to run. So, the simulation passes *all* of the structural tests described in section 7.1.1.1. The simulation can perform approximately 1100 iterations per minute for a moderately congested (i.e. approximately 850 vehicles active in the simulation) network with 28 nodes. The simulation works in discrete time, and each iteration is approximately 1.007 seconds. This is long enough for a vehicle to pull out into an intersection, come to a stop from 5 MPH, etc., but it is short enough time that most traffic maneuvers can be completed at a microscopic level. Additionally, the simulation has the capability to have vehicles enter the network and leave the network from *any pair* of nodes; this is an improvement on earlier versions of LS3 and on NET-TRAFSIM as well.

This completes our formal structural evaluation of LS3. We will test the behavioral characteristics in section 7.1.3.4.

SECTION 7.1.3.2 Structure of LS3

The structure of LS3 is further described in appendix B. The simulation accomplishes its tasks in four major subprograms. SIGNAL changes the traffic lights from iteration to iteration according to the TSC plan and ensures that no safety conflicts occur such as both directions being given green lights at the same

time. MOVE generates new vehicles and moves existing vehicles through the network. PRICAL reads in the real-time PRI information and calls the TSC setting routines. Last, STATS monitors and prints the MOE for the simulation. The SH, NLP and 3GC modules are integrated into PRICAL.

The LS3 has seven primary data structures for effectively administering the simulation. Structure "TSC" holds the information on the offsets, cycle time and splits for the signal lights. Structure "NET" maintains information on the physical layout of the network. In particular, NET holds the index ORDER which allows the simulation to determine whether a vehicle is turning left or right, going straight ahead, or even leaving the network as it comes to the end of a street segment. Structure "ROAD" contains the information concerning which vehicles are on which street segments and which positions on those street segments they occupy. Conversely, structure "AUTO" holds the information from a vehicle's perspective such as route choice, speed, current arc, status and platoon information (e.g. which vehicle it is following, which vehicles may be following it, etc.). Structure "MOE" contains exactly that, information on the measures of effectiveness. Structure "PRI" holds the predictive routing information, and last, structure "MGR" has the traffic manager inputs such as maximum cycle length, minimum green split, etc.

SECTION 7.1.3.3 Measures of Effectiveness

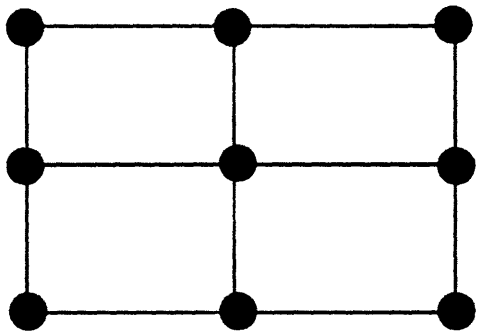
Every traffic simulation and traffic signal control strategy uses its own unique measures of effectiveness (e.g. some use average queue length, others use

average stops only, still others measure the average speed of vehicles, etc.). In chapter three, we emphasized stops per vehicle and delay per vehicle. In the LS3 simulator these measures of effectiveness are defined differently than they were in chapter three. In the LS3 simulation, we define the total amount of time a vehicle is traveling at less than 5 MPH as *wait* or *delay* . This measurement includes both full stops and significant braking operations. Delay can be viewed as the time of greatest driver frustration, i.e. when the vehicle is fully stopped or just creeping forward. Additionally, as we note in the diagram 7.5 above, the average transit time and delay per vehicle are positively correlated as well. Stops per vehicle are measured as one would expect. Every time the vehicle reaches 0 MPH a stop is recorded. Additionally, LS3 is capable of measuring the average transit time per vehicle as a measure of effectiveness. Several authors have argued that the average stops per vehicle in the network are the best measure of emitted pollutants. The average delay per vehicle is a measure both of lost time and driver frustration, and average transit time per vehicle is the easiest to visualize. We will focus on delay as our *primary* measure of effectiveness, but we will also display the average stops experienced per vehicle for those who prefer this measure. For the our supplemental runs with the surge scenario in section 8.2, we will also display the average transit time.

It is important to point out that the simulation is an approximation to real traffic behavior. For example drivers in the simulation are "timid"; they will only make a left turn or proceed onto a street if they have sufficient clearance to do so. In some cases this leads to drivers waiting five minutes or more during congested conditions to make a left turn. This delays not only the vehicle making the turn but the vehicles in back of it as well.

SECTION 7.1.3.4 Behavioral Testing

We tested the simulation's behavior using a 3x3 network called "bar". See the network in figure 7.1 below. Each street segment is two way. There are twenty-four one-way street segments in the network. The average flow rate on each of these 24 arcs is the same. For example, in the scenario where the average system flow rate is 72 vehicles per minute, each arc has an average flow rate of 3 vehicles per minute.

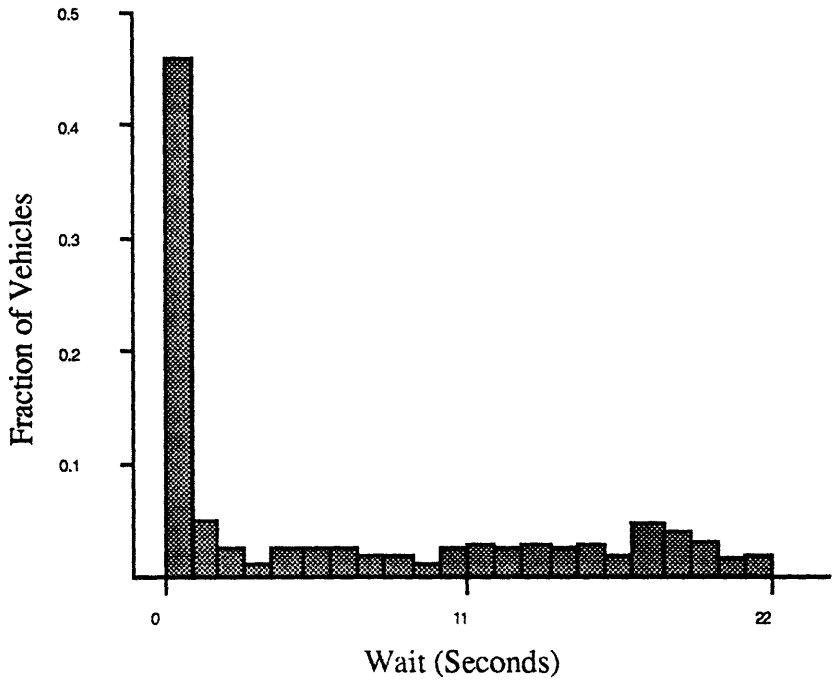


3x3 Network called "BAR"

Figure 7.1

The splits were uniformly set to 20 seconds of red and 20 seconds of green time for each intersection. The average wait per vehicle in "bar" with a network flow rate of 36 vehicles per minute was 6.04 seconds and had a standard deviation of 7.24 seconds. Under light traffic conditions, if we let the simulation run for an infinite time period then we would expect the average vehicle to wait for 5 seconds (which is just the probability of stopping for a random vehicle, one-half,

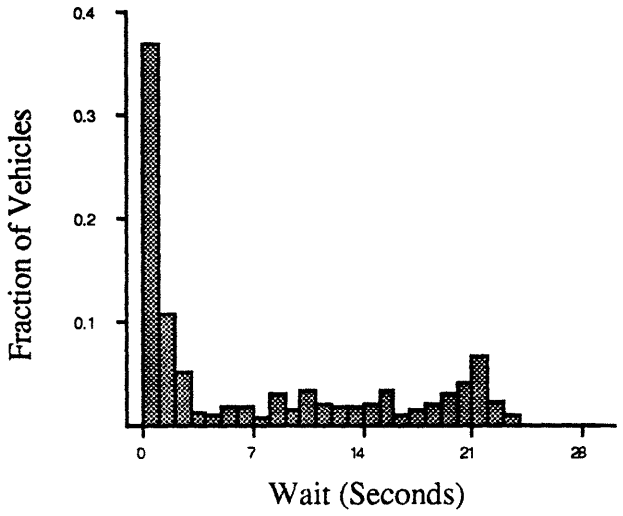
times the average amount of time we expect a vehicle to have to wait for if it has to stop, ten seconds), but as we see in figure 7.2 below at least some of the vehicles waited for up to 22 seconds which means there were at least *some* queuing effects experienced during the simulation run. This is confirmed by the fact that only 46% of the vehicles were able to pass through the network without stopping as opposed to the 50% we would expect to see under light traffic conditions. The 6.04 seconds seems like a reasonable estimate of the actual waiting time we would expect an average vehicle to experience in this network given these conditions. The actual wait time distribution is displayed in figure 7.2 below. (Note that some small fraction of the vehicles stopped more than once due to congestion effects.)



Histogram of Waits experienced by each vehicle exiting BAR for flow of 36 vehicles/minute.

Figure 7.2

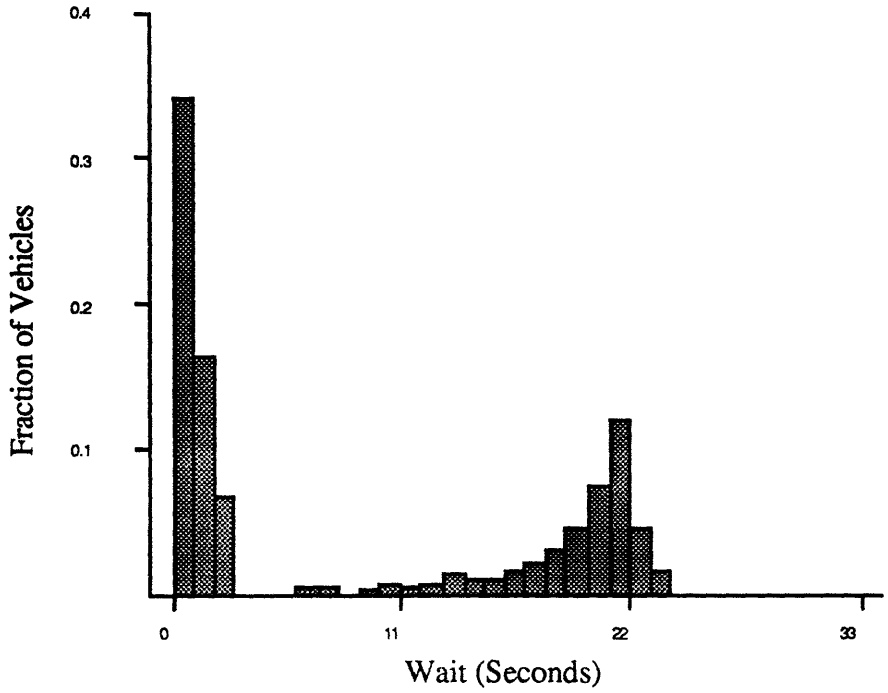
When the flow rate was increased in the network to 72 vehicles per minute in the network, the average wait increased to 7.05 seconds with a standard deviation of 8.11 seconds. The waiting time distribution is displayed below in figure 7.3.



Histogram of Waits experienced by each vehicle exiting BAR for flow of 72 vehicles/minute.

Figure 7.3

When the vehicle flow rate was further increased to 144 cars per minute, the average wait per vehicle went to 8.40 seconds with a standard deviation of 9.38 seconds. The waiting time distribution is displayed in figure 7.4 below.

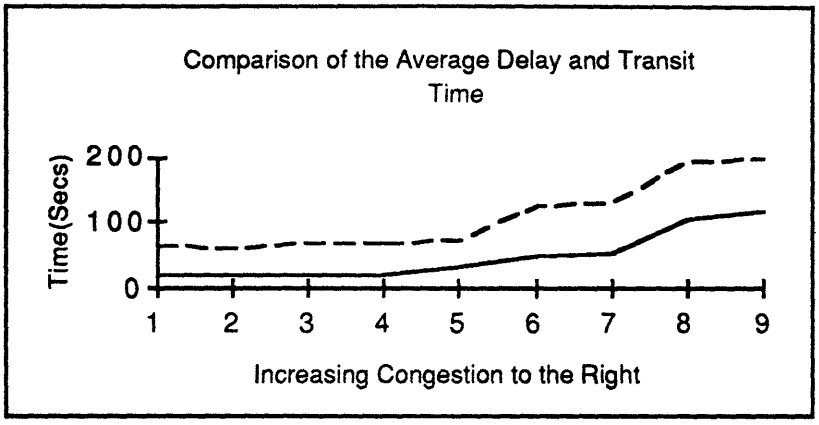


Histogram of Waits experienced by each vehicle exiting BAR for flow of 144 vehicles/minute.

Figure 7.4

As we increased the flow rate in the network, the average delay per vehicle increased [pattern 7.1.1.2(1)]. Also, the standard deviation of the wait increased; the variance in the size of the platoons becomes more pronounced as the system flow rate increases [pattern 7.1.1.2(2)]. We also note that congestion begins to play a role in the delay factor as the system flow rate was incremented. All of these are expected trends. The mean and standard deviation data reinforce our expected behavior for the simulation.

In the figure below, we see a plot of delay and transit time displayed as the level of traffic flow. Again, we see expected trends.



The top line represents the average transit time per vehicle, and the bottom line represents average delay per vehicle *as measured by the simulation.* Transit time and delay are closely related. The difference between the top and bottom lines can be interpreted as the time the vehicle is moving freely in the network. The widening gap indicates that the free movement is at a decreasing average speed. Notice that travel time increases for the more congested runs on the right side of the graph. This data was taken from eight runs of the surge scenario presented in section 8.2.

Figure 7.5

These figures were extracted from eight runs of the simulation using the surge scenario presented in section 8.2 below. The more congested runs are to the right side of the graph. It takes longer for vehicles to travel through the network under congested conditions [pattern 7.1.1.2(3)]. Table 7.1 shows the average delay and transit time by run.

Wait or Delay	Transit Time
21.151	69.553
21.926	69.72
23.278	73.915
25.835	79.277
53.389	130.988
89.079	169.515
106.306	193.86
117.409	199.917

This table lists the wait and transit time for each data run displayed in figure 7.5 (above).

Table 7.1

Later in the thesis we will refer to examples which will further bolster our confidence in the simulation results, but this concludes our formal testing of LS3.

SECTION 7.2. Testing the Non-Linear Program (NLP) vs the Split setting Heuristic (SH)

In this section we will compare the NLP and SH methods of setting splits and cycle time for PRISTINE. In chapter six we formulated the NLP in theoretical terms, but the NLP is not trivial to solve. In the next section will describe our method for solving the NLP. (We were able to code the SH directly as described in section 6.2.2.3, and we will not devote a separate section to describing this operation.)

SECTION 7.2.1 Implementing the Non-Linear Program (NLP)

In this section we will describe the methods used to implement the NLP described in section 6.2.2.2 previously. Recall the NLP took the form:

$$\textbf{Minimize} \quad \beta \sum_{j=1}^{n} \hat{T}_j + \gamma \sum_{j=1}^{n} \hat{W}_j$$

subject to

$$s_{ij} + s_{kj} = C \;\; \forall \;\; i,j,k \;\; \textbf{where} \;\; \Psi[(i \rightarrow j),(k \rightarrow j)] = .\textbf{false.}$$

$$s_{ij} > C\bar{x}_1\lambda_{ij} \;\; \forall \;\; (i \rightarrow j)$$

$$s_{ij} > \textbf{Network}_{min} \;\; \forall \;\; (i \rightarrow j)$$

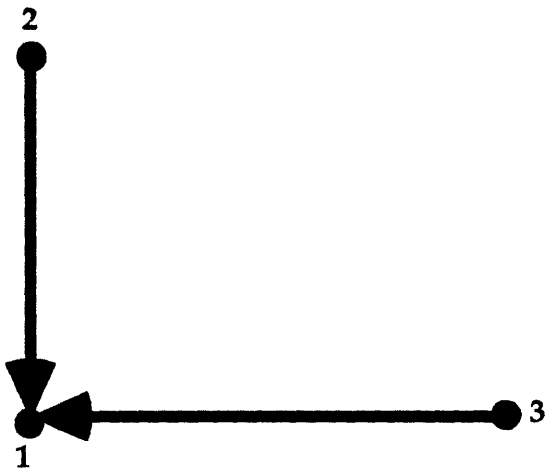
$$s_{ij} > \textbf{Arterial}_{min} \;\; \forall \;\; (i \rightarrow j) \in S$$

$$0 < C \leq C_{max}$$

$$s_{ij} \geq 0 \;\; \forall \;\; (i \rightarrow j)$$

Math Program 6.1

The nomenclature for math program 6.1 is contained in section 6.2.2.2. Imagine we have the sample intersection shown below in figure 7.6.



Sample Network

Figure 7.6

We know the following information about the intersection shown.

Direction	Average Arrival Rate	Green Time
$2 \rightarrow 1$	λ_A	G_A
$3 \rightarrow 1$	λ_B	G_B

Descriptive flow characteristics of network shown in figure 7.6, above.

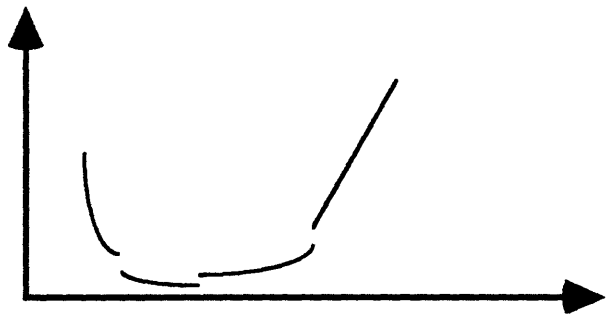
Table 7.2

For our objective function to be piece wise convex for this intersection, the following conditions must hold:

$$G_A > \bar{x}_1$$
$$G_B > \bar{x}_1$$
$$\lambda_A \bar{x}_1 (G_A + G_B) < G_A$$
$$\lambda_B \bar{x}_1 (G_A + G_B) < G_B$$

The first two conditions require the green split never falls below the average amount of time it takes for a single vehicle to safely cross the intersection, and the second set of conditions are system stability requirements. Essentially the average amount of green time in a direction must be sufficient to allow the average arriving vehicles to safely traverse the intersection. Extending these constraints to the network as a whole, we require that these constraints are met at every intersection. If the $\mathbf{NETWORK}_{min}$ green split is selected to be greater than the amount of time it takes for one vehicle to cross safely, this condition will be met automatically. The NLP assumes that the arrivals to the intersection occur in platoons separated by exponential headways while the platoon size is geometrically distributed. Math program 6.1 already contains the stability condition as one of its constraints.

Thus, we have a piece wise convex function being optimized over a convex set (i.e. all of the constraints are linear and hence convex). Unfortunately, there are complications. First, the NLP will be called repeatedly during the course of the simulation, and the street segments which are members of the tree will change from call to call. Second, the objective function is not convex, it is *piece wise convex*. Consider the function shown in figure 7.7 below.



An example of a piece wise convex, discontinuous function

Figure 7.7

This function is piece wise convex as well. It is non-differentiable to even first order across its entire domain. Fortunately, the *structure* of the problem aids us immensely. Although there are **2n+1** decision variables, two for each intersection (i.e. the green split in seconds each direction) and one for the cycle time, the problem is really much simpler than that. The objective function is comprised of a sum of the results for independent intersections, and the cycle time is equal to the sum of the green splits *at each intersection*. Therefore, we can use the chain rule to derive the following relationship:

$$\frac{\partial \mathbf{Objective}}{\partial C} = \sum_{i=1}^{2n} \frac{\partial \mathbf{Objective}}{\partial s_i} \bullet \frac{\partial s_i}{\partial C} \qquad [7.1]$$

where s_i is a green split at each intersection. But, to generate the partial derivative of C, the cycle time, we need to generate the partial derivatives for each unique split; there are a maximum of two of these per intersection, one for each direction which does not share green time. We can retain these results and apply a projection from the total gradient of the objective function in the split space, $\nabla_s \mathbf{Objective} \in \Re^{2n}$, onto the two dimension split space using the projection matrix: $\nabla_s \mathbf{Objective}(\nabla_s \mathbf{Objective}^T \nabla_s \mathbf{Objective})^{-1} \nabla_s \mathbf{Objective}^T$ (Strang, 1986).

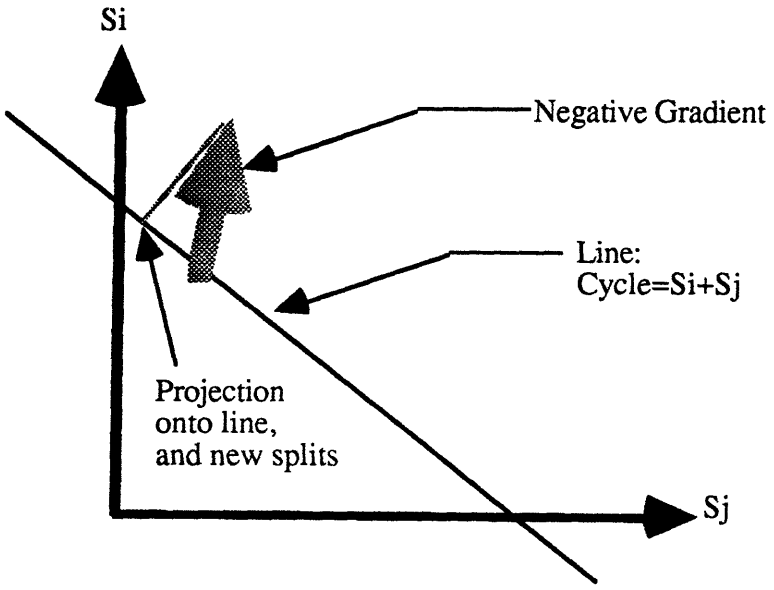
The problem remains that the objective is discontinuous and piece wise convex. Under these conditions, it is theoretically possible for NLP to select a local minimum which is not the global minimum. To address this problem, we did numerical approximation for pseudo-gradients, using a sufficiently large epsilon, about .1 seconds, to ensure that the discontinuities did not cause overflow errors (Bertsekas, 1992). Then, we took the steepest descent route and used a limited line search to find the next candidate minimum point. The line search technique ensured that the NLP looked for its next iteration point beyond

the continuous area the current point was in, i.e. the line searched "jumped over" discontinuities to search for the lowest point along the descent direction. When the line search selected a point either below the minimum cycle time or larger than the maximum cycle time, we projected back along the negative gradient until we intersected the appropriate boundary.

The objective function is an aggregate of two functions, one is a continuous convex function and the other is a piece wise linear function. The piece wise linear function is the portion which produces the discontinuities. The effect of the piece wise linear function *decreases* as the split for the priority direction *increases*. Beyond about 32 seconds, the multiplier for the linear function drops off dramatically. For example, if the split for the priority direction is 32 seconds the multiplier is about .007, and it gets smaller after that. Since we used a lower cut off for priority splits of 30 seconds, the continuous convex function dominates, and the NLP is likely to select the local minimum within the constrained area.

For example, say that the traffic flow is approximately seven vehicles per minute along both the priority and non-priority arcs entering the intersection, and the current splits are set at 32 seconds each. We will use weightings of one for both stops and delay in the objective function. The contribution from the continuous convex function is on the order of 2.53 while the contribution from the linear function is approximately .273. The combination of using the line search, which cuts across the discontinuities, and the small effect of the linear function in the region where the priority split is greater than 30 seconds makes it improbable that NLP would select a sub-optimal point.

Why did we use the negative gradient? The gradient gives the direction of greatest increase, and we are trying to minimize a weighted sum of the average stops per vehicle and wait per vehicle experienced in the network. So, we want to go in the direction of greatest decrease. In the $\Re^2$ intersection split space, the solution method is displayed in figure 7.8 below.



Gradient projection method used by NLP to improve splits at each iteration

Figure 7.8

We solved the issue of the changing problem structure by using Boolean variables, either 1 or 0, which were tied to the condition "this street segment is part of the tree." Essentially, the math program is run through a pre-processor step where the Boolean variables are assigned. Once the Boolean variables are set, the program is passed onto the NLP solver.

One last issue arose. Initially the NLP took a relatively long time to converge. It tended to oscillate around the optimal solution, sometimes

requiring up to 10,000 iterations or more to converge. Most of these steps took the solution outside the feasible area. To counter this, we introduced barrier functions that severely penalized the objective for points outside the feasible region. This brought the number of iterations down to less than 1,000 for most simulation runs.

SECTION 7.2.1.1 Testing the Non-Linear Program (NLP)

In the end, there was one last question though. Did the NLP actually find the optimal solution as defined by the theoretical work in chapters three through six? The NLP portion of the program sets the splits and cycle times. There is a heuristic portion that sets the offsets according to the spanning tree. We used the Maximal Spanning Tree (MST) method for the NLP module. The MST was extensively error tested, and it each case the MST routine selected the arcs it should have according to the procedures developed in chapter four. There is no generalized solution for how to optimally set the cycle and splits for a traffic network, but in a very simple case, we can test the output from the simulation/NLP to see if it is close to an optimal solution. The most evident case is the light traffic example. Recall the situation described in figure 7.6 and table 7.2, the optimal setting for the light traffic example is to set the split in proportion to the arrival rate in each direction. To test the simulation/NLP performance, we: set the minimum green split to 30 seconds, set the platoon size to a constant of one, set the flow rates equal in both directions and let the simulation/NLP run for six hours of objective time. Theoretically, the average vehicle should have stopped .5 times, and the average vehicle should have waited for 7.5 seconds.

The actual simulation results revealed a wait of 7.505 per vehicle, and the average vehicle stopped .5 fraction of the time. Notice that throughout the last paragraph we mentioned testing both the simulation and the NLP split setting method. To get the results we did, presumably both portions, the NLP and the simulation, had to be working. The level of agreement between the theoretical result and the result seen in practice is satisfactory.

The Split setting Heuristic (SH) was tested for the Potential Function (PF) case shown at the end of section 6.2.2.3. It calculated the results as shown in tables 6.3 through 6.5. These figures are completely accurate. We will not devote a separate section to this result.

SECTION 7.2.2 Description and Results of the Simulation Runs

In this section we will explore the simulated performance of the two competing methods for the Predictive Routing Information Signal Timing INtEgration (PRISTINE) method. The Non-Linear Program (NLP) for split setting and Split setting Heuristic (SH) approaches for PRISTINE were compared using data runs on the Lin-Sarkar-Staats simulation model. We would *expect* the NLP to dominate the SH, because the NLP solves to optimality while the SH gives an approximation of the optimal solution. However, recall that the assumptions that the NLP and SH are based on do not completely reflect "reality" as it exists in the simulation. So, even though the NLP solves the traffic signal problem to optimality according to its assumptions, this may not represent

the best settings for the simulation. The NLP begins with the SH solution as its initial solution.

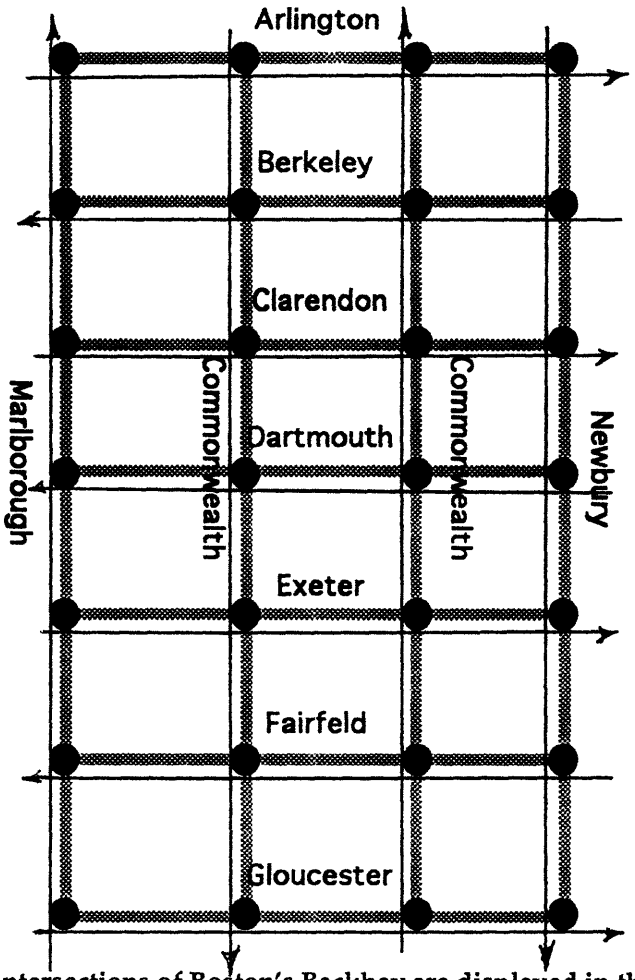
We would expect one of three outcomes from our simulation trials. The first possibility is that the SH produces MOE's that are very close or even superior to those produced by the NLP. Thus, the SH would offer a much faster alternative to the NLP while not suffering a substantial penalty in the way of performance as measured by the MOE's. A second possibility would be that the NLP would offer substantial savings over the MOE's generated when using the SH method, but the NLP would take substantially longer than the SH to find the optimal solution. The third and last possible outcome would be that the NLP would offer substantial savings over the MOE's generated using the SH, and the NLP would produce its timing plan after a relatively short period of time. The data runs revealed the third option to be the most accurate depiction of the relationship between the two models.

The model was run on a DEC Station 5000 using the following parameters: maximum cycle length=2 minutes, minimum green split=10 seconds, minimum priority green split=30 seconds, maximum acceptable average delay per intersection for non-priority routes=30 seconds. For consistency, these parameters will be used for all of our simulation runs in chapter seven.

We discovered in the course of our research that the Measures of Effectiveness for the simulation were *relatively insensitive* to the choices of the

weightings for stops and wait in the objective functions of PRISTINE and 3GC. We did simulation runs with weightings on the average delay per vehicle and average stops per vehicle ranging from total contribution from stops to total contribution from waits. None of the runs showed a significant change to the MOE's based on these changes of parameters. We did note that the average stops per vehicle, the average delay per vehicle and the average transit time per vehicle are correlated. If one MOE was low then all the MOE's tended to be low. As we point out in sections 6.1 and 8.1, it is not in the best interest of the traffic manager to use the average stops per vehicle or the total stops throughout the network for a given time frame as the primary basis of judging a TSC plan. So, we selected the average wait per vehicle as the dominant MOE.

We used the Boston Backbay area as our idealized network (see figure 7.9 below).



The streets and intersections of Boston's Backbay are displayed in the figure above.

Figure 7.9

Physically the network consists of a moderate number of nodes and street segments, 28 and 45 respectively. The arcs are one-way street segments, and the maximum speed on each of the segments is 30 MPH.

Five specific scenarios were selected to compare the NLP and SH performance and included:

Name	Time Duration	Vehicles/Hour
Light Traffic	2 AM-5 AM	370
Morning Start-Up	6 AM-7:30 AM	1434
Morning Rush	7:30 AM-8:30 AM	2671
Mixed Traffic	9 AM-4 PM	2448
Evening Rush	4:30-5:30 PM	2748.

The table above displays the key information about each of the scenarios used to test the SH and NLP. In the table, vehicles per hour represents the average number of vehicles exiting the simulation per hour of simulated time.

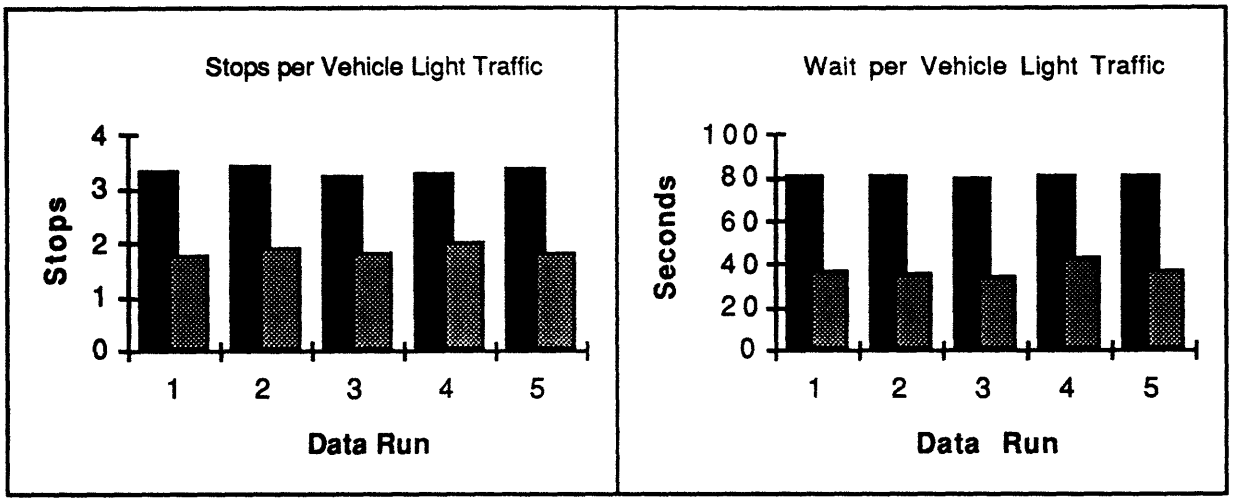
Table 7.3

A sample set of PRI is displayed in appendix C.

Recall that our goal was to get a feel for the first order effects; that is we are looking for the major trends rather than extremely detailed relationships between the inputs to the traffic signal control systems and the outputs from the simulation. Each of the possible combinations of SH or NLP and choice of scenarios was run five separate times, using like seed values for the random number generator for the runs of the NLP and SH. For example, the number 16838 was always used as the random seed for the first run in a series whether it was the NLP or SH that was being tested, and 5758 was used as the second seed, etc. We selected five runs as the number of trials, because we wanted to get a feel for the variance of the MOE's using the NLP and SH models. If one method dominated for a particular scenario then five runs was enough to show this trend. If the methods produced MOE's which were sufficient for suggesting the "tie." (A much larger set of runs to find out which model was slightly ahead did not seem worthwhile.) This was borne out by our simulation data runs in the remainder of the section.

The SH ran faster than the NLP. In fact, the SH solution was used as the starting point for the NLP. Both the NLP and SH were designed to solve the model described in chapter three. The simulation is by nature stochastic. Neither the NLP or SH can tell exactly what will happen in the network once the simulation begins running. The predictive routing information used by both is just an estimation. It is entirely possible that the SH could derive a solution that works better for a particular simulation run than the solution produced by the NLP. The NLP typically took between 500 to 800 recursions (i.e. the NLP examined 500 to 800 different traffic signal settings) to converge to a constrained optimal solution.

Specific results by run and analysis are as follows(the light bar is always the SH result while the solid bar is for the NLP). The light traffic scenario is designed to capture late night traffic flows in the network. There are no discernible patterns, and the traffic flow is extremely light, on the order of one car per minute on each street segment. The vehicles essential are randomly incident on each traffic light in the network. Several interesting points emerge as we analyze figure 7.10 below.



The figure above shows the results from the light traffic scenario. The dark bars represent the data from the NLP while the light bars are from the SH. In this case, we are measuring average Stops per Vehicle and average Wait per Vehicle, and smaller is better. The SH performs better for this scenario.

Figure 7.10

The numerical results are displayed in table 7.4 below.

NLP WAIT	SH WAIT	*NLP STOPS*	*SH STOPS*
80.497	36.231	*3.32*	*1.79*
80.588	36.152	*3.44*	*1.9*
80.315	34.129	*3.26*	*1.81*
80.748	42.768	*3.29*	*2*
81.094	37.043	*3.38*	*1.8*

This table displays the average stops per vehicle and wait for vehicle experienced by the SH and NLP methods for each of the extended simulation runs in the light traffic scenario. Each row is the result of one data run for each model. Since smaller totals are better, the SH performed better than the NLP for this scenario.

Table 7.4

For the light traffic scenario, the NLP produced a mean value of 3.338 stops per vehicle with a standard deviation of .072 across the simulation runs, and the NLP simulation runs had a mean waiting time of 80.648 seconds per

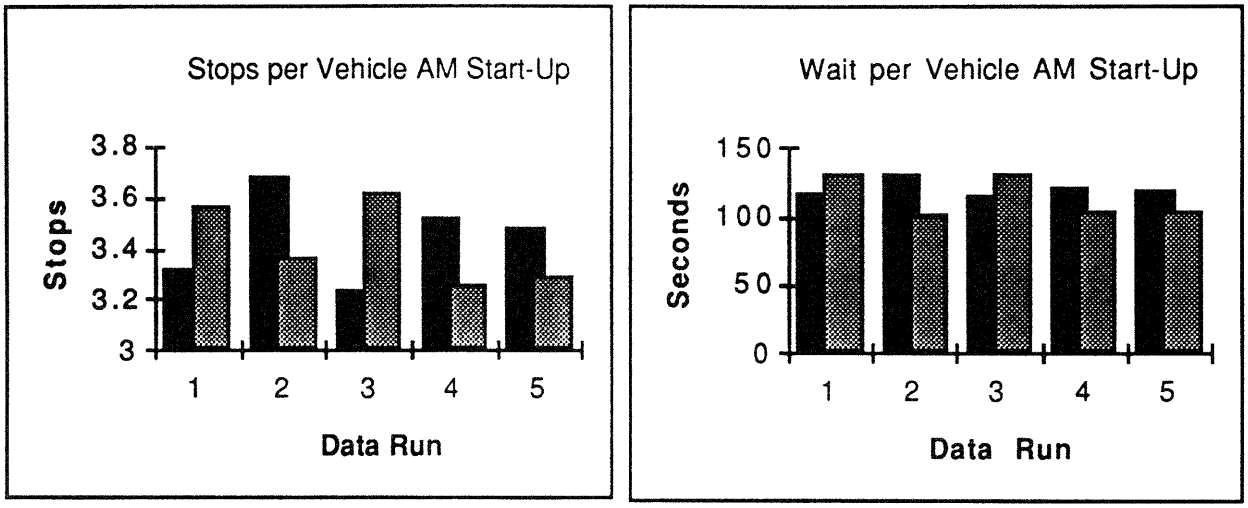
vehicle with a standard deviation of .294 seconds. The SH produced a mean value of 1.86 stops per vehicle with a standard deviation of .09 across the simulation runs, and the SH simulation runs had a mean waiting time of 37.265 seconds per vehicle with a standard deviation of 3.259 seconds. The SH produced a savings of 53.79% for waiting time in this scenario.

The SH method actually does better than the NLP method for this scenario. Why is this the case? Recall equation 6.6, $\frac{\lambda_A}{\lambda_A + \lambda_B} = \frac{G_A}{C}$. This describes the optimal solution for the light traffic model. Under the conditions described for the light traffic scenario, the assumptions for the light traffic model are met. Recall from section 6.2.2.3 the SH determines the minimum amount of time required at each intersection using the Queue Effects Model (QEM) from chapter 5 to ensure the average vehicle is detained for less than the maximum acceptable average delay per intersection. Typically in the light traffic case this was between 1 to 5 seconds. Then, the SH method ensures that the minimum green splits across the network are applied. In our case, the minimum green split for a priority street segment is 30 seconds. So, the minimum practical setting for the network cycle was 60 seconds. (The spanning tree touches every node; therefore, if the network has more than two intersections, there will be at least one node with at least two tree arcs attached to it. It would be extremely unlikely that every such node would have only one or zero *incoming* tree arcs.) The SH increases the splits in each direction in proportion to the vehicle flow rate in that direction. Where the minimum QEM requirement is very small, the final solution is a very close approximation to equation 6.6.

The NLP on the other hand, always considers potential queuing effects before finally setting the splits; in light traffic, it considers them too much.

Imbedded in the NLP model is a preset parameter **B** which represents the ratio of the standard deviation of the platoon size in the network divided by two times the average platoon size. When the parameters were set for the NLP, we selected a value for **B** based on the assumption that congestion *would* have a significant effect on traffic flow in the network. We focused on heavier traffic, because this is where the potential savings possible under traffic signal control are the greatest. So, the NLP set the cycle time to 60 seconds in the light traffic case too, but it also *added a bit extra* to the green split for the non-priority direction. We used a shifted geometric distribution to furnish platoon sizes in the simulation; thus, it was *possible* to have platoons of large size. By and large, such platoons did not arise in the light traffic case, and this is why the SH tended to do better under these conditions. It would be possible to design a system where the **B** values could be assigned by intersection and measured in real time as traffic approached the intersection; this would allow the NLP to perform better under light traffic conditions. As the traffic volume increased, the NLP began doing better than the SH.

The morning start-up scenario is a transitional scenario from the light traffic situation prevalent in the night to the morning rush hour. The traffic initially is disorganized with small numbers of vehicles traversing the network more or less uniformly, but as time goes on a definite pattern of traffic moving from West to East emerges. The morning start-up scenario ends as morning rush hour begins. The results for the morning start-up are displayed in figure 7.11 below.



The dark bars represent the data from the NLP while the light bars are from the SH for the morning startup scenario. We are measuring average Stops per Vehicle and average Wait per Vehicle, and smaller is better. It is not immediately evident which method did better in this case.

Figure 7.11

The numerical results are displayed in table 7.5 below.

NLP WAIT	SH WAIT	*NLP STOPS*	*SH STOPS*
117.16	130.981	*3.32*	*3.57*
129.88	102.011	*3.68*	*3.36*
114.136	130.482	*3.24*	*3.62*
120.479	102.556	*3.52*	*3.26*
118.499	102.295	*3.48*	*3.29*

This table displays the average stops per vehicle and wait for vehicle experienced by the SH and NLP methods for each of the extended simulation runs for the morning start-up scenario. Each row is the result of one data run for each model. Smaller is better. Neither method is a clear winner for this scenario.

Table 7.5

For the morning start-up scenario, the NLP produced a mean value of

seconds per vehicle with a standard deviation of 21.074 seconds. The NLP produced a savings of 37.63% in waiting time for this scenario.

The NLP begins to do better under the more congested conditions found in the morning start-up scenario, and the NLP *completely* dominates the SH once congestion becomes significant (e.g. the morning rush hour scenario). It is not obvious if either the SH or NLP dominates for the morning start-up. In the well known Wilcoxon rank-sum test (Berenson and Levine, 1992), for the ten results (5x2) the sum of the ranks for the average waits of the SH and NLP methods are 27 and 28 respectively, and this is as close to a tie as possible. Our statistics are shown below in table 7.6.

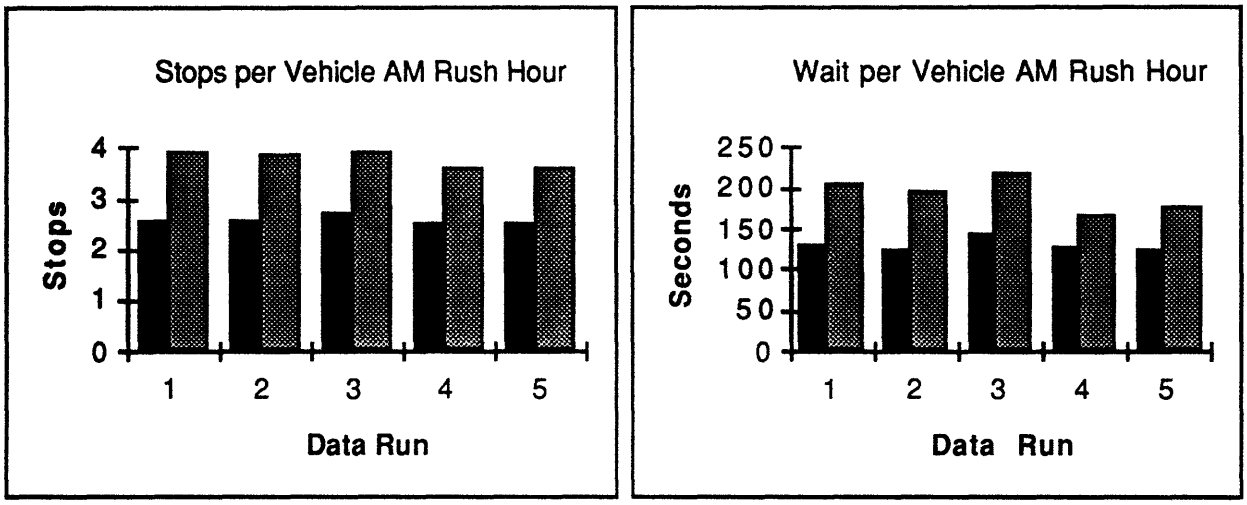
Method	Rank	Wait per Vehicle
SH	1	102.011
SH	2	102.295
SH	3	102.556
NLP	4	114.136
NLP	5	117.160
NLP	6	118.499
NLP	7	120.479
NLP	8	129.880
SH	9	130.482
SH	10	130.981

Wilcoxon Rank-Sum Table for the Morning Start-Up Scenario. There is no statistical difference between the waiting time experienced for the SH and NLP methods for this scenario at the 5% level significance.

Table 7.6

The morning rush hour scenario was designed to test the NLP and SH methods under congested conditions where clear patterns are discernible. In this case, we have a traffic flow from West to East along two major arterials,

Commonwealth Avenue and Newbury Street. The results for these data runs are shown in figure 7.12 below.



The results of the data runs for the morning rush hour scenario are displayed above. The dark bars represent the data from the NLP while the light bars are from the SH. We are measuring average Stops per Vehicle and Wait per Vehicle, and smaller is better. The NLP does better in this scenario.

Figure 7.12

The numerical results are displayed in table 7.7 below.

NLP WAIT	SH WAIT	*NLP STOPS*	*SH STOPS*
130.191	203.946	*2.58*	*3.89*
125.084	194.412	*2.58*	*3.88*
142.374	219.931	*2.69*	*3.91*
126.556	167.067	*2.51*	*3.6*
122.992	176.927	*2.49*	*3.6*

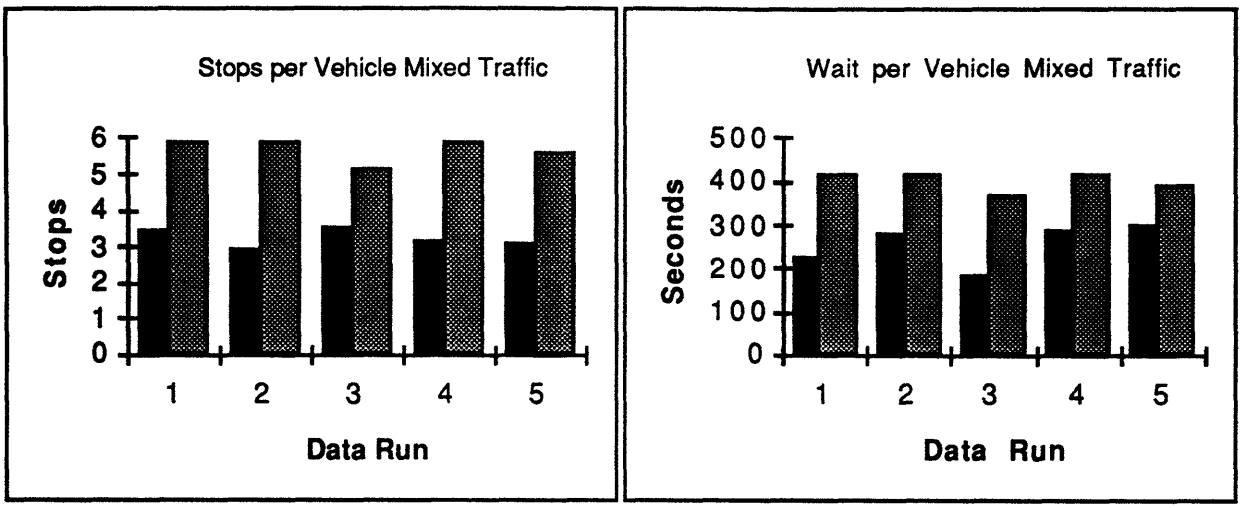
This table displays the average stops per vehicle and wait for vehicle experienced by the SH and NLP methods for each of the extended simulation runs for the morning rush hour scenario. Each row is the result of one data run for each model. Smaller is better. The NLP dominates the SH for this scenario.

Table 7.7

For the morning rush hour scenario, the NLP produced a mean value of 2.57 stops per vehicle with a standard deviation of .078 across the simulation runs, and the NLP simulation runs had a mean waiting time of 129.439 seconds per vehicle with a standard deviation of 7.69 seconds. The SH produced a mean value of 3.776 stops per vehicle with a standard deviation of .161 across the simulation runs, and the SH simulation runs had a mean waiting time of 192.457 seconds per vehicle with a standard deviation of 21.074 seconds.

The NLP solution clearly dominates the SH in each data run. Queuing effects are now significant. Now there is a dominant flow from West to East. The flow rate into the network during the morning rush hour was approximately 2650 vehicles per hour. There were times during the morning rush hour scenario runs when capacity was temporarily exceeded and vehicles were forced to wait through entire green cycles, because the street segments ahead were clogged. Even though the vehicle flow rate goes down during the next scenario, the mixed traffic scenario, the NLP and SH methods produce worse statistics, because they are unable to exploit definite traffic patterns.

The mixed traffic scenario was designed to test the SH and NLP methods under conditions of congestion when there are no clear traffic patterns. We see the results of these data runs displayed in figure 7.13 shown below.



The results of the data runs for the mixed traffic scenario are displayed above. The dark bars represent the data from the NLP while the light bars are from the SH. We are measuring average Stops per Vehicle and Wait per Vehicle, and smaller is better. The NLP does better in this scenario.

Figure 7.13

The numerical results are displayed in table 7.8 below.

NLP WAIT	SH WAIT	*NLP STOPS*	*SH STOPS*
228.331	415.071	*3.47*	*5.91*
284.422	416.908	*2.97*	*5.9*
185.988	365.78	*3.52*	*5.19*
287.31	417.924	*3.18*	*5.86*
302.616	394.454	*3.11*	*5.6*

This table displays the average stops per vehicle and wait for vehicle experienced by the SH and NLP methods for each of the extended simulation runs for the mixed traffic scenario. Each row is the result of one data run for each model. Smaller is better. The NLP dominates the SH for this scenario.

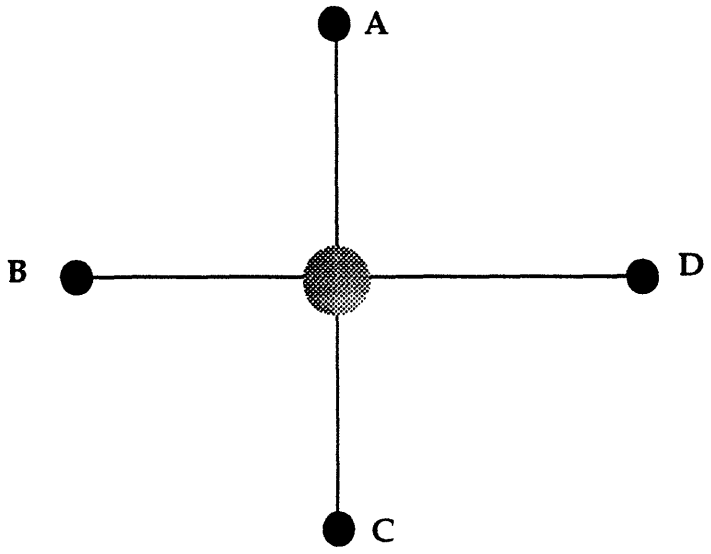
Table 7.8

For the mixed traffic scenario, the NLP produced a mean value of 3.25 stops per vehicle with a standard deviation of .237 across the simulation runs, and the NLP simulation runs had a mean waiting time of 257.735 seconds per vehicle with a standard deviation of 49.024 seconds. The SH produced a mean

value of 5.692 stops per vehicle with a standard deviation of .308 across the simulation runs, and the SH simulation runs had a mean waiting time of 402.027 seconds per vehicle with a standard deviation of 22.447 seconds. The simulation uses cautious drivers and single lanes; so, under congested conditions a driver making a left hand turn may have to wait several green periods before being able to make a left hand turn. This delays not only the vehicle making the left hand turn but also the vehicles behind it.

Although the NLP outperforms the SH, both methods do *worse* than they did under the morning rush hour scenario even though the overall traffic flow rate has *decreased* from 2650 vehicles per hour to 2450 vehicles per hour. Why would this be the case? In both of the rush hour scenarios, there is a very strong correlation between what the IVHS predictive routing information *projects* and what the simulation *actually sees* at the intersections during the run. Under congested conditions vehicles travel more slowly. PRISTINE using either the NLP or SH models projects the traffic patterns for the traffic signal control period based on PRI it receives at the beginning of the period. (In the simulation, this traffic signal control period is set as a ten minute interval.) If the traffic is sufficiently delayed, a substantial portion of the vehicles may still be in the network at the end of the cycle. If the next batch of PRI reflects different traffic patterns, PRISTINE may set the traffic signals in a sub-optimal fashion. In the case of the morning rush hour scenario, there is are preferred directions of travel through the network from West to East. So, even if the traffic from one traffic signal control period gets delayed then it will not have a large adverse effect, because the traffic patterns for the next traffic signal control period are very similar to the current period.

There are additional reasons why PRISTINE does better in situations with dominant directions of flow. Consider the intersection shown in figure 7.14 below.



Sample Intersection illustrating dominant flows

Figure 7.14

For example, in figure 7.14 we see that there are twelve possible combinations incoming and outgoing directions from the intersection. If vehicles coming from nodes "B" and "D" made up the majority of vehicles traveling through this intersection, we would say that there was a *dominant* directional traffic flow through this intersection. In this case, we could set the splits at the intersection to better accommodate the dominant flow pattern, i.e. we would increase the fraction of green time given to the vehicles traveling from nodes "B" and "D". We are even more fortunate if there is one particular traffic pattern that dominates at the intersection, because this allows us to set up the offsets to aid

this pattern. Look again at figure 7.14. If 80% of the vehicles entering this intersection came from node "C" and headed toward "D" then we could set up a progression scheme with this intersection and node "D" that would benefit 80% of our vehicles.

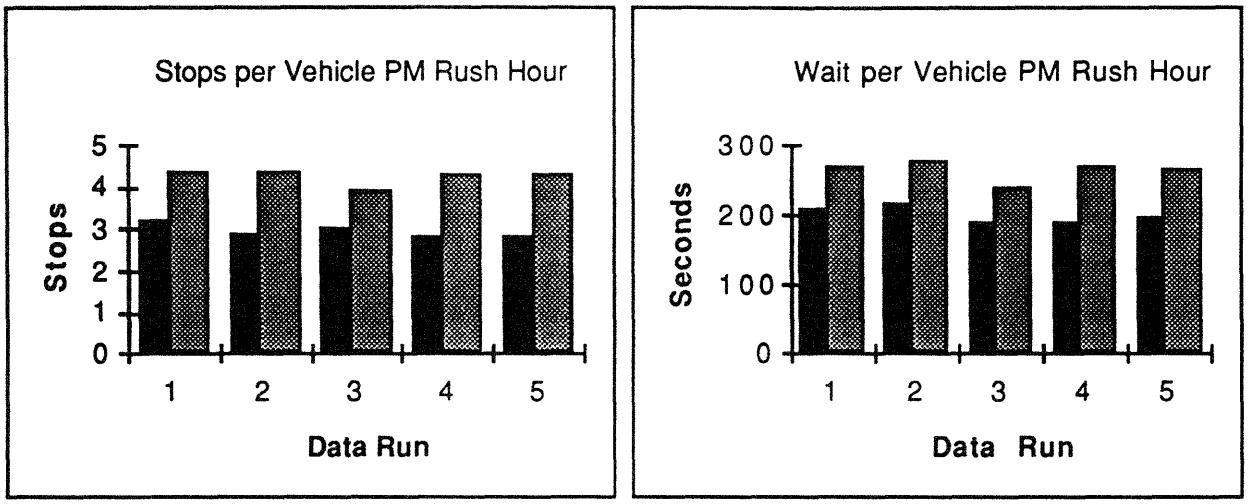
The key to understanding the issue of dominant directional flows is realizing that aiding one route in a traffic scheme will generally harm another route. To get a better insight into this issue, consider the following example. Suppose that you had a traffic signal control plan that will help "x" fraction of the population. Let us say that $.5 < x < 1$. Also assume that you can reduce the normal MOE (e.g. stops per vehicle or wait per vehicle) to some fraction "y" of its initial value of one, and we will assume that $0 < y < 1$. Assume that initially both groups, x and (1-x), both have an MOE rated at one. For simplicity, imagine a symmetric situation where helping x by y means that the remaining fraction of (1-x) will have their MOE multiplied by 1/y. For this traffic control strategy to offer an advantage over the status quo, the following relationship must hold: $xy+(1-x)(1/y) < 1$ (we are trying to get smaller MOE's in this case). Simplifying this relationship, we must have: $x > 1/(1+y)$ for this traffic signal control strategy to give an advantage. Let us say that we are able to aid 80% in the example above; then we must be able to offer a savings of at least 25% to the 80% who benefit for the control strategy to be worthwhile.

Unfortunately, dominant directional flows and patterns do not exist in the mixed traffic scenario. We generally see random patterns at the intersections. If we take this lack of dominant directions at the intersections and we add high levels of congestion, we have the situation present in the mixed traffic model. Under these conditions about the best one can do is increase the green time in both directions and try to maximize the capacity of the network. In effect, we can

consider each individual intersection as an independent optimization process where the goal is maximum capacity rather than attempting to minimize average stops or wait.

There is an additional problem for PRISTINE in the mixed traffic model. The progression scheme set up by PRISTINE is not as successful as it might be, because the traffic moves very sluggishly through the network. By the time the vehicles arrive at the intersection, the light is already red. Under its most ambitious visions, IVHS would give traffic managers the ability to track each vehicle's speed and position in the network (IVHS, 1993); using a more sophisticated traffic signal control model that integrates this real time location and speed data might alleviate some of the congestion degradation that PRISTINE experienced under the mixed traffic scenario. Under the evening rush hour scenario the traffic increases to an average flow rate in the network of 2750 vehicles per hour and performance improves.

The evening rush hour scenario was designed to test the methods on their ability to perform under conditions more congested than the mixed traffic scenario but *featuring a definite flow* from East to West on the streets of the sample network. (For our generated data, the evening rush hour is symmetric with the morning rush hour, but this was *not* the case with the real world data from the City of Boston as we will see in the next section.) The results are displayed in figure 7.15 shown below.



The results of the data runs for the evening rush hour scenario are displayed above. The dark bars represent the data from the NLP while the light bars are from the SH. We are measuring Stops per Vehicle and Wait per Vehicle, and smaller is better. The NLP does better in this scenario.

Figure 7.15

The numerical results are displayed in table 7.9 below.

NLP WAIT	SH WAIT	*NLP STOPS*	*SH STOPS*
210.175	269.927	*3.2*	*4.41*
216.28	276.644	*2.91*	*4.36*
191.188	240.963	*3.06*	*3.93*
191.061	270.748	*2.84*	*4.29*
195.774	268.456	*2.83*	*4.34*

This table displays the average stops per vehicle and wait for vehicle experienced by the SH and NLP methods for each of the extended simulation runs for the evening rush hour scenario. Each row is the result of one data run for each model. Smaller is better. The NLP dominates the SH for this scenario.

Table 7.9

For the evening rush hour scenario, the NLP produced a mean value of 2.968 stops per vehicle with a standard deviation of .159 across the simulation runs, and the NLP simulation runs had a mean waiting time of 200.9 seconds per vehicle with a standard deviation of 11.619 seconds. The SH produced a mean value of 4.266 stops per vehicle with a standard deviation of .193 across the simulation runs, and the SH simulation runs had a mean waiting time of 265.348 seconds per vehicle with a standard deviation of 13.982 seconds.

The NLP uniformly dominates the solutions from the SH for this scenario. Both the NLP and SH do better under this scenario than for the mixed traffic scenario, because the inherent progression in the spanning tree method used for setting the offsets is able to take advantage of the arterials found in the scenario, Marlborough Street and Commonwealth Avenue.

SECTION 7.2.3 Summary

In the preceding section we examined the aggregate results of some 250,000+ simulated vehicles traveling through Boston's Backbay. We tested a Split setting Heuristic (SH) form of PRISTINE which exploited the structure of the signal timing problem to derive a solution and a Non-Linear Program (NLP) which solved this theoretical problem to optimality. The NLP took longer to process the data, but the extra time was well worth the effort. When the traffic signal timing plan generated by the NLP and the SH were tested in the simulation, the NLP outperformed the SH in three of five scenarios, tied in

another, and was inferior only in the case of very light traffic (e.g. the light traffic scenario). The general findings are summarized in the table below.

Scenario	SH Average Stops(SD)	NLP Average Stops(SD)	SH Average Wait(SD)	NLP Average Wait(SD)
Light Traffic	1.860(.090)	3.338(.072)	37.265(3.259)	80.649(.294)
Morning StartUp	3.420(.165)	3.448(.173)	113.665(15.582)	120.031(5.970)
Morning Rush	3.776(.161)	2.570(.078)	192.457(21.074)	129.439(7.690)
Mixed Traffic	5.692(.308)	3.250(.237)	402.027(22.447)	257.733(49.024)
Evening Rush	4.266(.139)	2.968(.159)	265.348(13.982)	200.900(11.619)

The table above contains the summary statistics for the five scenarios used to test the SH against the NLP for use in PRISTINE. Smaller is better for the average delay per vehicle and average stops per vehicle. The numbers shown in parenthesis are the standard deviations for the quantities.

Table 7.10

Because the NLP was the more successful of the two approaches for PRISTINE, we will use the NLP to set the splits and cycle time for PRISTINE when we test PRISTINE against the optimized third generation control strategy, 3GC.

Chapter Eight

Comparing an Idealized Third Generation Control (3GC) Model and PRISTINE

In the last section we compared two versions of PRISTINE, a heuristic version and a non-linear program version. In this section we will develop an optimized Third Generation Control (3GC) system and compare its performance against PRISTINE using our Boston Backbay network as the testbed and motivated by *real* Boston traffic data. The 3GC system is developed in section 7.3.1 and appendix A.

SECTION 8.1 Implementing 3GC

Homburger and Kell described a second generation traffic signal control system as a type of control program which provides for on-line, real time traffic signal control utilizing a prediction model to predict near term changes in traffic demand (Homburger and Kell, 1988). Second generation control systems generally share common cycle lengths throughout the network; the cycle times are typically fixed or based on historical data (Ballman, 1991). Third Generation Control (3GC) systems improve on second generation control systems by allowing the cycle time to vary by intersection.

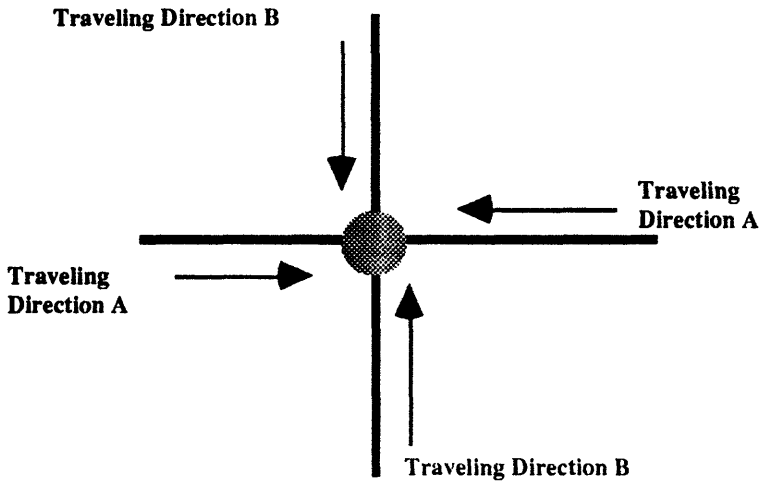
There are many versions of third generation control systems which are commercially available including: SCAT, SCOOT, OPAC, PRODYN, UTCS 3-GC, etc. (Ballman, 1991), but none have achieved more wide spread usage or better results in practice than SCOOT (Bretherton, Bowen, et.al., 1986), (Robertson and Bretherton, 1991). We will construct our 3GC model incorporating many of the features of SCOOT. As mentioned earlier, second and third generation control systems utilize predictive models to transform accumulated data into a usable TSC plan. All such systems make the same steady-state assumption; namely, the arrival patterns recorded since the last TSC plan was implemented accurately reflect what the system will experience during the time the TSC is in effect. SCOOT is successful to a large extent due to its effective use of the data extracted from traffic sensors in making its predictions (Hounsell, McLeod, et.al., 1990). In particular SCOOT maintains Cyclic Flow Profiles (CFP) which are measures of the average flow of vehicles past a specific point on the roadway; this is generally a sensor location. SCOOT works on three principles: measure CFP's in real time, update its internal queuing model continuously, and incrementally update signal settings (Robertson and Bretherton, 1991). SCOOT accomplishes this using information gathered from inground sensors; the sensors and controlling programs report back such information as average vehicle speed, roadway occupancy and vehicle flowrate (Hounsel, McLeod, et.al., 1990).

Because commercial traffic signal timing packages are required to work in real time, it is frequently not feasible to achieve optimal settings in practice. For example, SCOOT typically is allowed to complete between 3 to 5 improvement iterations before implementing the next TSC (Gartner, 1992), and the old split settings are not allowed to vary from the old split settings by more than four seconds (Robertson and Bretherton, 1991). Here is where our model, 3GC, will

break with the tradition in the literature in two important areas. First, we will allow 3GC to iterate until it finds the optimal solution. Second, although third generation control systems have the ability to vary the cycle times by intersection, in practice many of them consider the cycle time to be fixed while operating in the field (e.g. SCOOT) (Homburger and Kell, 1988); we *will allow* the cycle time to vary by intersection for 3GC.

An important aspect of third generation control systems is that they treat each intersection as an isolated unit (Papageorgiou, 1991). There are large theoretical advantages to this approach. For example, if an intersection at the far edge of the city is experiencing little traffic, it would make sense to decrease its cycle time to decrease expected wait per vehicle, but it might not make sense to decrease the cycle time for a very congested intersection in the middle of the city. Since the TSC system treats the intersections independently, it has the latitude to decrease one cycle time while leaving the other cycle intact or even increasing it as required. We will incorporate this feature into 3GC as well. (On the other hand, PRISTINE uses a common cycle time across the entire network.) We have sufficient information to begin constructing 3GC.

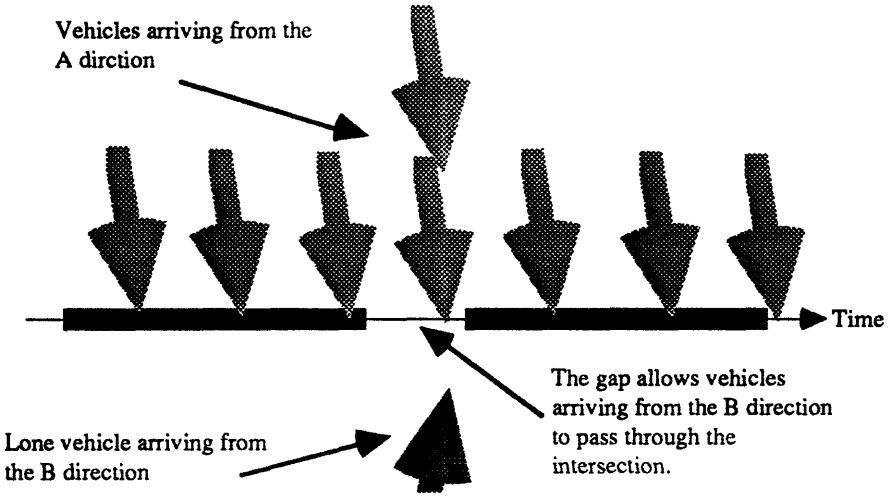
The overall objective of 3GC will be to minimize a weighted sum of the number of stops per vehicle and the wait per vehicle in the area under the control of 3GC. Although these goals are in some ways complementary, minimizing one does not assure us that the other will also be minimized as demonstrated by the example shown below. Imagine we have the simple network displayed in figure 8.1 below.



Sample network

Figure 8.1

Vehicles can approach from four street segments, and the street segments are paired according to those which share common green cycle. We will refer to one pair collectively as direction A and the other as direction B. Suppose we had recorded the arrivals shown in figure 8.2 below.



In the diagram above the gray arrows in the top half of the diagram represent vehicles arriving from the "A" direction in figure 8.1 while the dark arrow below the line represents an arrival at the intersection from the "B" direction. The arrows represent arrivals at the intersection at instances in time. If there is one arrow, there was one arrival while two arrows represent two arrivals, i.e. vehicles arrived from both street segments representing the "A" direction at the same time. Vehicles from the "A" direction can only pass through the intersection during time segments represented by thick, solid line segments, but the vehicles from the "B" direction can only pass through during the "gaps". Diagram illustrates how a strategy seeking only to minimize stops in the system could lead to poor results in practice. In this example, the vehicle from direction "B" would never be given any green time.

Figure 8.2

If our sole objective was to minimize the stops experienced per vehicle, we would just allow green time in the A direction. The poor lone vehicle arriving in the B direction would never be given any green time. We need to resort to the multi objective programming methods we refereed to in chapters one and six. In this case, we will take a convex combination of the wait per vehicle and stops per vehicle and seek to minimize that combination. Additionally, the traffic manager assigns lower bounds for the green splits in the network which also prevents the problem of not assigning any green time to a particular direction. If we examine diagram 8.2, we get an insight into how to solve this problem.

We can view our third generation control model as a *covering problem.* Imagine we have the same situation as in figure 8.2, but now we are allowed to move the slot back and forth according to a parameter θ.

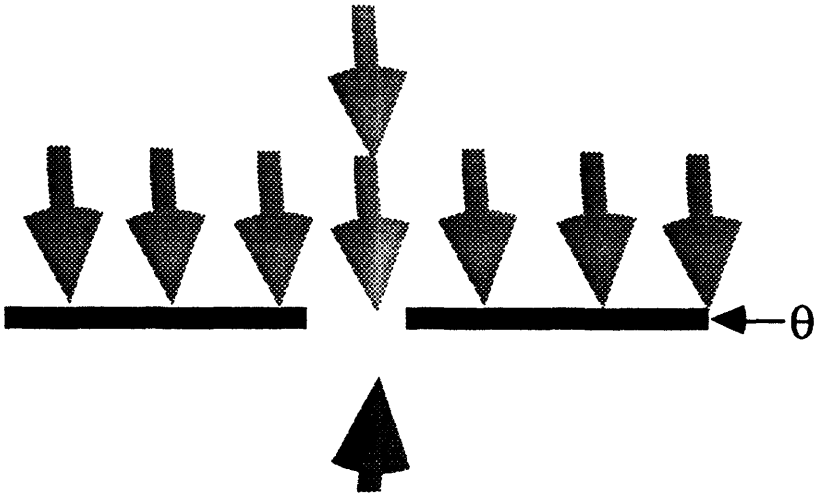
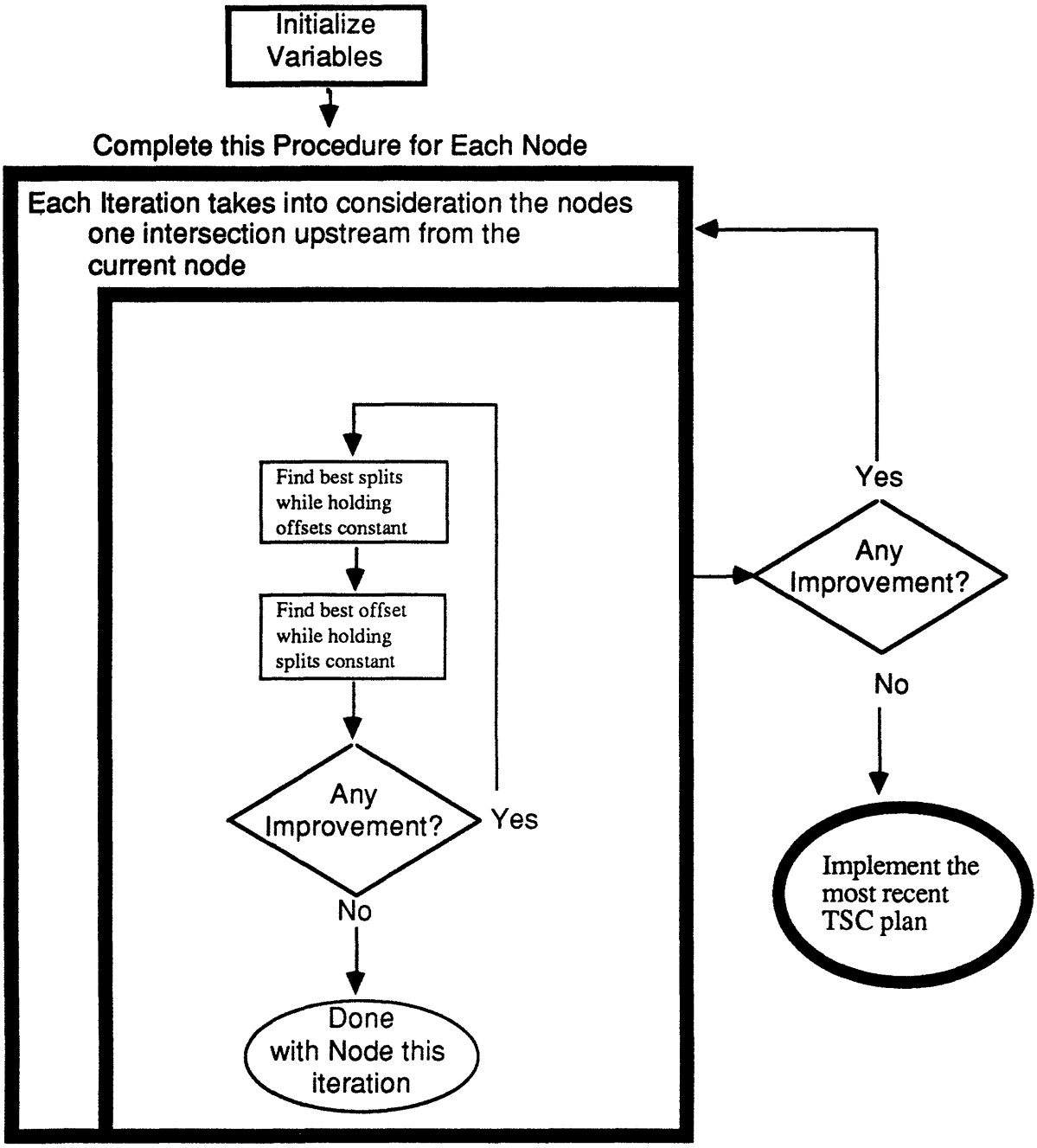


Figure illustrates the use of a parameter to shift the red and green time to either hold or pass through the recorded arrivals. The slot depicted above shifts to the left or right according to the value of θ. This can be viewed as a problem of *covering* the recorded arrivals in such a way as to minimize the wait per vehicle and stops per vehicle.

Figure 8.3

Now, we can set a value for θ depending on our goal. If we wanted to minimize the wait experienced per vehicle, we would shift the slot slightly to the right and allow the two vehicles to pass through unimpeded and then allow the lone vehicle to pass through while delaying only one vehicle from the other direction. We have three parameters which we can set at each intersection, green split (in seconds) each direction and offset for the green splits. Cycle time is just a sum of the green splits at the intersection. We will collect the arrival data for each intersection directly from the simulation; so, in essence, our 3GC has perfect information. This is a great improvement over conditions in the real world

where the traffic signal control strategy must contend with estimations, lack of sensors and sensor faults. In particular, 3GC has a large advantage over SCOOT which has to rely on secondary information such as historical turning percentages to update its CFP's under many circumstances. We will perform successive optimizations on each intersection and then the entire system. We will continue to iterate until no further improvement takes place in the network. We are greatly aided in our endeavors by the fact that the simulation uses discretized time; thus, we are able to contend with a total measure space of $(Cycle_{max})^{3n}$ discrete points rather than an infinity of them; of course, $(Cycle_{max})^{3n}$ can get to be quite large too! As we will see in the testing, it would be impractical to attempt to solve 3GC to optimality in a situation where it had to provide the TSC in real time. The flow chart below displays the logic used in 3GC to set the TSC.



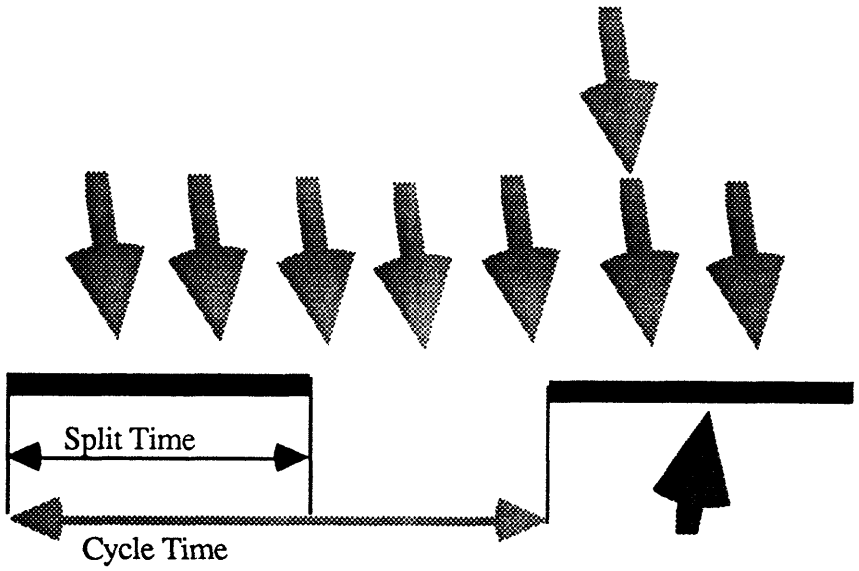
Flow chart above depicts the control structure for the 3GC strategy

Figure 8.4

The procedure begins by looking at the first node in the network. The procedure has at its disposal a complete listing, stored in three dimensional

array, of the arrivals along that street segment moment by moment since the last time 3GC was run. It conducts a point by point search of the cycle time space and selects the cycle time which gives the *best objective function value* consisting of a weighted average of the stops and average delays experienced by vehicles over the collection period. The splits are increased or decreased proportionally to their current values during this phase of 3GC. The cycle time is allowed to range from **2*Green**$_{min}$ to **Cycle**$_{max}$.

The next parameters set are the splits. This also turns into a best objective function search for the splits, but we only need to make one pass through a linear range of split times. Why? Consider figure 8.5 shown below.



The figure shows how one can change the green split in one direction while the total cycle time remains constant. In this case, the gap shown above increases or decreases in size as the green split changes in length.

Figure 8.5

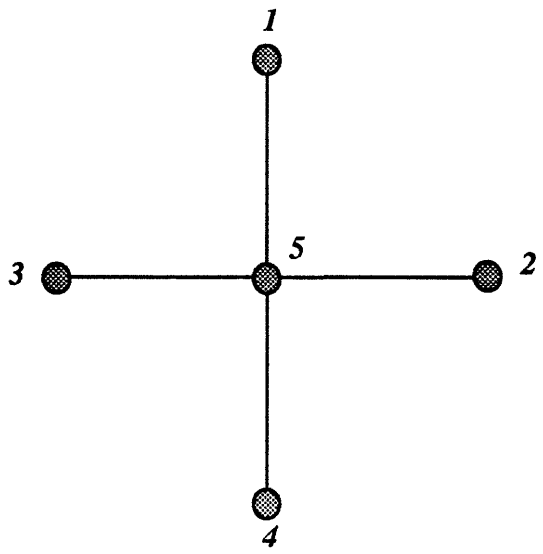
the lowest ranking arc entering the intersection as measured from an arbitrary zero point (i.e. integral multiples of the current intersection cycle time from the simulation start time). Offset value for direction A was allowed to vary from zero seconds to the green split for direction B.

However, 3GC was very greedy when it came to CPU time. A typical run for a two hour block of simulated time took between one hour to one hour and fifteen minutes. The vast majority of that time was spent by 3GC determining the optimal settings for the TSC plan.

Of course, 3GC makes some assumptions. It assumes for example, that if the light is green then vehicles are allowed to flow forward in the network; thus, 3GC ignores the case of gridlock and delays due to traffic maneuvers such as left turns onto busy thoroughfares. However, these are intrinsically taken into consideration, because the arrival rate on the various arcs will decrease as traffic becomes clogged. Additionally, since 3GC only looks one street segment ahead at a time, there is no over arching progression scheme, but again, this is also implicitly considered. If one considers linking each of a nodes neighbors with itself then eventually the entire network is linked. Like all second and third generation TSC routines, 3GC implicitly assumes that the last data collection period will accurately represent events during the next TSC period. This is the assumption which routines using predictive routing information are able to exploit.

Before moving on, it would instructive to examine a simple example of how the 3GC model sets the traffic signal controls given a particular set of inputs. Let us assume that the maximum cycle length is 5 seconds. The minimum green split is 2 seconds. The entire traffic signal timing period is 10 seconds long. We

will use the network shown below to illustrate how the 3GC model works. (Note that nodes 1,2,3 and 4 are *dummy* nodes; so, we can focus on generating the traffic signal setting for node 5.) Assume that the speed limit on each arc is 5 MPH, and the travel time from any node to node 5 is two seconds.



Node and incoming arcs used to illustrate 3GC method

Figure 8.7

Arcs (1->5) and (4->5) share a green cycle at node 5 as do arcs (2->5) and (3->5). So, we can completely specify our traffic signal controls for node 5 by setting the following quantities: green split for (1->5), green split for (2->5), and offset for (1->5). We will weigh the delays and stops in the objective function with a value of one each.

Our arrivals for the last traffic signal control period (ten seconds) are shown below.

Time	Arrivals on (1->5)	Arrivals on (2->5)	Arrivals on (3->5)	Arrivals on (4->5)
1				
2	X		X	
3		X	X	
4				
5		X	X	X
6	X			X
7				X
8			X	
9				
10				

The "X" represent arrivals at the tail node of each arc shown. For example, seven seconds into the last traffic signal timing period, a vehicle arrived on arc (4->5) heading toward node five.

Table 8.1

There are 15 possible traffic signal settings as shown below:

Split for (1->5)	Split for (2->5)	Offset for (1->5)
2	2	0
2	2	1
2	2	2
2	2	3
2	2	4
3	2	0
3	2	1
3	2	2
3	2	3
3	2	4
2	3	0
2	3	1
2	3	2
2	3	3
2	3	4

The table shows the possible traffic signal settings for node five in the diagram above.

Table 8.2

We will work through one complete setting to see how the stops and delay are calculated. Let the offset for (1->5) be zero, and the splits for (1->5) and (2->5) be set at two seconds each. The results for this setting are shown in the table below. (*Recall there is a two second delay from the time a vehicle enters one of the arcs until it reaches node five.*)

Time	Arrivals to node 5 from Arcs (1->5) & (4->5)	Light	Stops	Delay	Arrivals to node 5 from Arcs (2->5) & (3->5)	Light	Stops	Delay
1	0	Green	0	0	0	Red	0	0
2	0	Green	0	0	0	Red	0	0
3	0	Red	0	0	0	Green	0	0
4	1	Red	1	1	1	Green	0	0
5	0	Green	0	0	2	Red	2	4
6	0	Green	0	0	0	Red	0	0
7	1	Red	1	2	2	Green	0	0
8	2	Red	2	2	0	Green	0	0
9	1	Green	0	0	0	Red	0	0
10	0	Green	0	0	1	Red	1	1
11	0	Red	0	0	0	Green	0	0
12	0	Red	0	0	0	Green	0	0

The table displays the stops and delays which would have been accrued by the arrivals during the last traffic signal control period if the settings for node 5 been an offset of zero for arc (1->5) and green splits of two seconds each for arcs (1->5) and (2->5).

Table 8.3

The total objective for this setting would be equal to 17.

Now, 3GC would explore changing the splits while *leaving the offset constant*. 3GC would select the splits which minimize the objective. In this case 3GC would find the best splits to be a split of three seconds for arc (1->5) and two for arc (2->5) for a total objective value of 12.

Since this is smaller than the previous best of 33, 3GC would minimize over the offsets while the *splits were held constant*. 3GC would discover that the

offset of two for (1->5) gives an objective function value of 10 which is best given the splits.

The process repeats until 3GC reaches an iteration where it cannot improve over its last solution. In this case, 3GC would stop with the solution: split for (1->5) equal to two, split for (2->5) equal to three, and offset for (1->5) equal to two. This gives an overall objective value of two which is also global minimum.

We have assumed (see appendix A) that the waits and stops are convex with respect to *both* offsets and the splits. Under these conditions, this alternate line minimization strategy will reach the optimal solution in a finite number of iterations. It is much faster than using an exhaustive enumeration technique.

SECTION 8.2 PRISTINE compared to 3GC

In this section we will explore the simulated performance of PRISTINE (using the NLP to set its splits and cycle time) versus the 3GC system we developed in the last section. PRISTINE and 3GC were compared using five sets of data on the Lin-Sarkar-Staats simulation model. The simulation was run on a DEC Station 5000 using the following parameters: maximum cycle length=2 minutes, minimum green split=10 seconds, (for PRISTINE, the minimum priority green split=30 seconds, maximum acceptable average delay per intersection for non-priority routes=30 seconds). We used the Boston Backbay area as our idealized network (see figure 7.9 above). We chose to conduct ten runs for each scenarios in the comparisons between 3GC and PRISTINE to better illustrate any differences between the MOE's produced by the two methods.

Physically the network consists of a moderate number of nodes and street segments, 28 and 45 respectively. The arcs are one-way street segments, and the maximum speed on each of the segments is 30 MPH. The five scenarios used to compare PRISTINE and 3GC's performance were as shown in table 8.4 below.

Name	Time Period	Source
Early Morning	12 AM - 2 AM	Boston Traffic
Mid-day Traffic	11 AM - 1 PM	Boston Traffic
PM Rush Hour	4 PM - 6 PM	Boston Traffic
AM Rush Hour	7 AM - 9 AM	Boston Traffic
Surge Response	N/A	Constructed

The table above displays the key information about each of the scenarios used to test the PRISTINE and 3GC.

Table 8.4

We used actual traffic data, collected by the traffic sensors in the Backbay, to set the average traffic flows (see appendix C for an example of how the PRI was generated from this data). The last scenario is the surge response scenario.

We define a *surge* as a sudden increase in traffic flow such that the *average* flow rate over the next TSC period will exceed the capacity of one or more street segments under the *current* signal settings but *does not* exceed the *potential* capacity of the street segments, i.e. if the traffic signals were readjusted, the average traffic flow could be accommodated. By TSC period, we mean the length of time that a particular TSC plan will be in force; in the case of our simulation, we used a ten minute interval. Since we reset our TSC plan every ten minutes, "the next TSC period" would be the ten minute period when the next TSC plan is in effect. We refer to the *average* flow rate, because it is entirely possible even under light traffic conditions for an intersection to have its capacity exceeded for

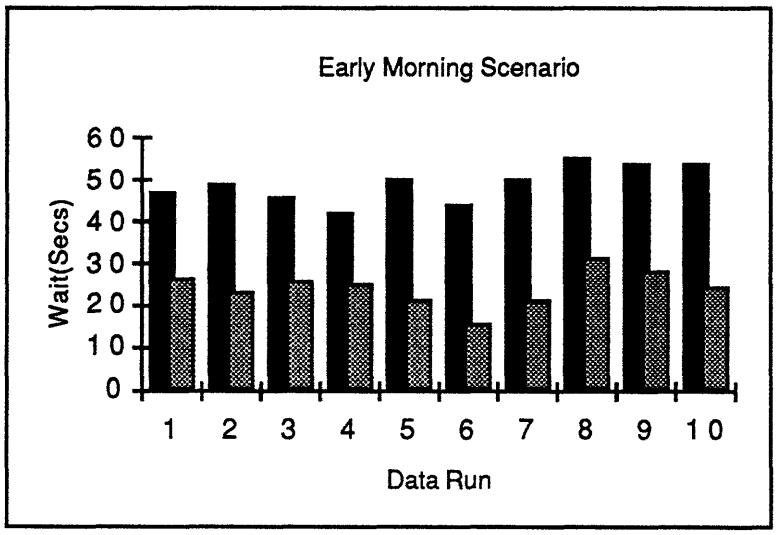
one or more cycles even though the average flow rate does not exceed the street's capacity. Last, we want to consider cases where the potential capacity of the network is not exceeded so that the problem is feasible.

A surge might arise in practice when a concert or sporting event suddenly lets out. Another example would be if an earthquake, fire or other disaster suddenly caused a mass exodus from a metropolitan area. Severe weather could cause certain roads to be impassable for periods of time; this could also result in a surge condition as we have defined it.

Each of the possible combinations of PRISTINE or 3GC and choice of scenarios was run five separate times, using like seed values for the random number generator for the runs of PRISTINE or 3GC. For example, the number 31051 was always used for the fifth run of a sequence. PRISTINE ran significantly faster than 3GC. Whereas a typical two hour block of simulated time ran in seven to ten minutes for PRISTINE, the same simulation typically ran for over an hour using 3GC.

The information from the Boston Traffic Department was extracted from the sensors data actually collected in the Backbay area of Boston. Mr. George Hawat, Boston Traffic Manager, supplied the sensor data broken out by fifteen minute intervals. The routings were constructed in such a way as to support the sensor information. Sensor locations and an example of PRI are included in appendix C. Specific results by run and analysis are included in subsequent paragraphs (the lighter color is always the PRISTINE result in the charts below).

The results for the early morning traffic runs are displayed in figure 8.8 below.



The results of the data runs for the early morning scenario are displayed above. The dark bars represent the data from 3GC while the light bars are from PRISTINE. PRISTINE does better in this scenario.

Figure 8.8

The numerical results for this scenario are displayed in table 8.5 below.

3GC Wait	PRISTINE Wait	*3GC Stops*	*PRISTINE Stops*
46.831	26.298	*1.189*	*1.899*
48.852	22.911	*1.237*	*1.267*
46.03	25.556	*1.16*	*1.881*
42.023	24.967	*1.144*	*1.897*
50.062	21.47	*1.353*	*1.228*
44.13	15.467	*1.202*	*1.076*
50.432	21.426	*1.236*	*1.255*
55.093	31.412	*1.323*	*2.133*
53.659	28.225	*1.324*	*1.891*
53.959	24.389	*1.274*	*1.238*

This table displays the average stops per vehicle, wait for vehicle experienced and objective value for 3GC and PRISTINE for each of the simulation runs for the early morning scenario. Each row is the result of one data run for each model. Smaller is better. PRISTINE dominates 3GC for this scenario in the MOE of average wait per vehicle.

Table 8.5

For the early morning scenario, PRISTINE does substantially better on waiting time while 3GC does slightly better on stops. 3GC produced a mean value of 1.244 stops per vehicle with a standard deviation of .072 across the simulation runs and had a mean waiting time of 49.107 seconds per vehicle with a standard deviation of 4.377 seconds. PRISTINE produced a mean value of 1.577 stops per vehicle with a standard deviation of .394 across the simulation runs and had a mean waiting time of 24.212 seconds per vehicle with a standard deviation of 4.322 seconds.

We would expect 3GC to do best under conditions where the individual intersections can be viewed as isolated systems. The traffic during this time frame was *light* varying from an average of about 46 cars per minute throughout the entire network to a low of about 20 cars per minute. PRISTINE did better in

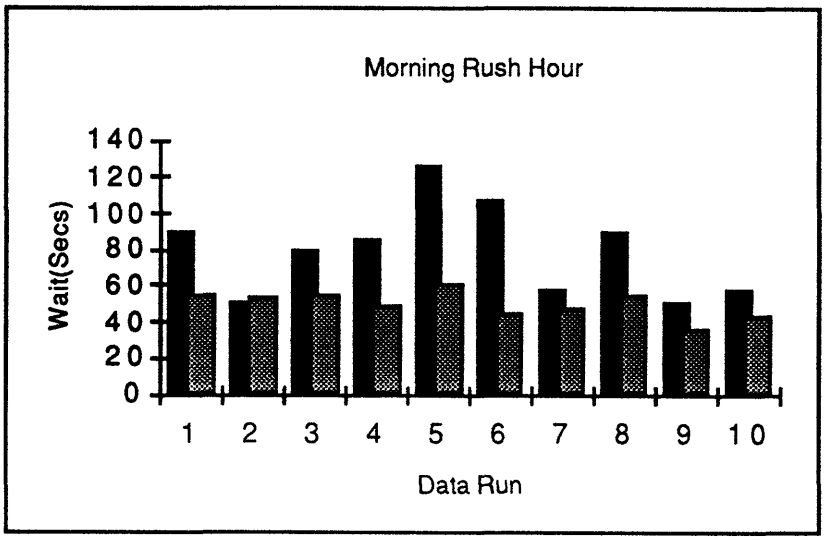
our primary MOE, delay. On the other hand, 3GC did better than PRISTINE in the area of stops. It is quite surprising to see PRISTINE the more consistent algorithm for light traffic. Why would this be the case? The key is in recognizing how 3GC derived its solution compared to PRISTINE's technique. Whereas PRISTINE considers an average anticipated flow rate into an area (based on PRI data) and finds an optimal solution, based on expected values, 3GC tries to optimize over the *exact* values measured during the last measurement interval.

The traffic this time of the morning does not follow definite patterns and certainly is not in a steady state situation. What is the impact of this on 3GC? Suppose that 3GC observed only a single vehicle coming along Arlington and turning onto Commonwealth during the last observation period, it would respond the next period by setting the green time to its maximum value ($\mathbf{cycle}_{max}$-$\mathbf{green}_{min}$) along each street the vehicle traveled and making the offsets support the vehicle's route.

Since traffic is so light though, it is *just as likely* that the next vehicle could be traveling *across* the specified route as *along* it. The low stop values tell us that this is exactly what is happening. Only a few vehicles have to stop, but the ones that do are waiting for long periods. Thus, 3GC does not do as well as PRISTINE which tends to set the individual splits closer to the light traffic optimal values.

Some commercial TSC packages overcome this problem by integrating historical data into the input for the package, but this creates problems of its own if the traffic does not follow the old patterns. Typically in practice, the traffic manager observes the traffic sensor input, and if it varies significantly from the historical data or bottle necks begin to occur, he or she manually intervenes (Hawat, 1992), (Gartner, 1992).

The morning rush hour runs gave predictable results. The results for the morning rush hour runs are displayed in figure 8.9 below.



The results of the data runs for the morning rush hour scenario are displayed above. The dark bars represent the data from 3GC while the light bars are from PRISTINE. PRISTINE outperforms 3GC for this scenario.

Figure 8.9

The numerical results for this scenario are displayed in table 8.6 below.

3GC Wait	PRISTINE Wait	*3GC Stops*	*PRISTINE Stops*
89.518	54.552	*1.723*	*1.91*
51.019	54.011	*1.263*	*2.819*
80.152	55.542	*1.527*	*1.829*
84.739	49.644	*1.408*	*1.763*
126.024	60.296	*2.015*	*1.966*
107.585	45.342	*1.726*	*2.165*
57.411	48.017	*1.343*	*2.247*
90.384	54.211	*1.607*	*2.419*
50.499	35.594	*1.588*	*1.833*
58.141	43.142	*1.285*	*2.176*

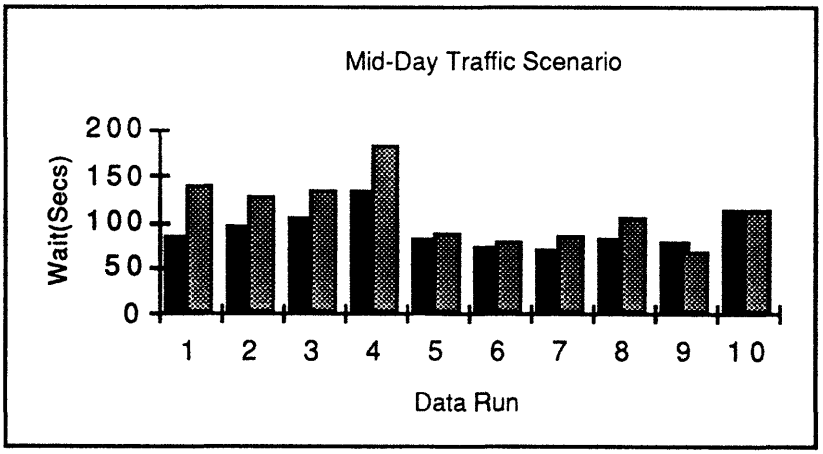
This table displays the average stops per vehicle, wait for vehicle experienced and objective value for 3GC and PRISTINE for each of the simulation runs for the morning rush hour scenario. Each row is the result of one data run for each model. Smaller is better. PRISTINE performs better than 3GC for this scenario in our primary MOE of average wait per vehicle.

Table 8.6

For the morning rush hour scenario, PRISTINE again does substantially better on waiting time and slightly worse in the area of stops. 3GC produced a mean value of 1.549 stops per vehicle with a standard deviation of .235 across the simulation runs and had a mean waiting time of 79.547 seconds per vehicle with a standard deviation of 25.378 seconds. PRISTINE produced a mean value of 2.113 stops per vehicle with a standard deviation of .326 across the simulation runs and had a mean waiting time of 50.035 seconds per vehicle with a standard deviation of 7.239 seconds.

It is not surprising that PRISTINE beats 3GC for this scenario. Throughout the morning rush hour, there is a preferred direction of travel, West to East, and the traffic flow builds up steadily from 7 AM till 8:45 AM, and even then, it only tapers off a little. This is precisely the type of situation where PRISTINE *should* do well.

The mid-day traffic provided an interesting set of results. The mid-day traffic run results are displayed in figure 8.10 below.



The results of the data runs for the mid-day traffic scenario are displayed above. The dark bars represent the data from 3GC while the light bars are from PRISTINE. 3GC did better than PRISTINE for this scenario.

Figure 8.10

The numerical results for this scenario are displayed in table 8.7 below.

3GC Wait	PRISTINE Wait	*3GC Stops*	*PRISTINE Stops*
85.636	139.413	*1.654*	*3.265*
94.542	128.851	*1.746*	*3.163*
103.183	131.874	*1.759*	*3.257*
132.043	182.791	*2.466*	*3.217*
82.391	86.005	*1.577*	*2.662*
72.995	77.543	*1.482*	*2.718*
70.417	83.744	*1.487*	*2.806*
82.826	103.098	*1.625*	*2.744*
79.814	66.801	*1.54*	*2.676*
113.875	113.838	*1.896*	*3.16*

This table displays the average stops per vehicle, wait for vehicle experienced and objective value for 3GC and PRISTINE for each of the simulation runs for the mid-day traffic scenario. Each row is the result of one data run for each model. Smaller is better. 3GC does better than PRISTINE for this scenario.

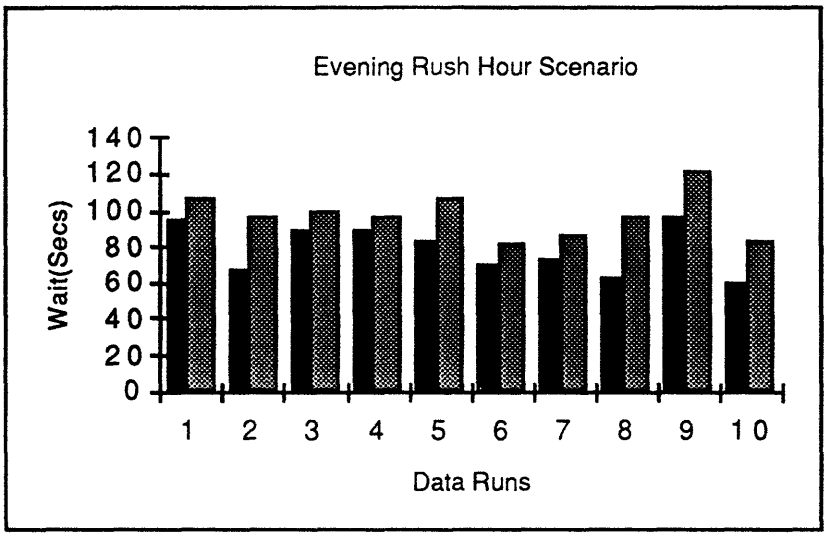
Table 8.7

For the mid-day traffic scenario, 3GC dominates PRISTINE. 3GC produced a mean value of 1.723 stops per vehicle with a standard deviation of .292 across the simulation runs and had a mean waiting time of 91.772 seconds per vehicle with a standard deviation of 19.452 seconds. PRISTINE produced a mean value of 2.967 stops per vehicle with a standard deviation of .264 across the simulation runs and had a mean waiting time of 111.396 seconds per vehicle with a standard deviation of 35.316 seconds.

Why was PRISTINE was beaten by 3GC in this scenario? In the mid-day pattern, there is a large amount of congestion on the streets, but there is no particular *pattern* to it. The lack of pattern allows one to regard the individual traffic signals as isolated units. It does no good to set the offsets to work for any particular route, because there is nearly as much cross traffic as there is through

traffic along the major routes. At the same time, the fact that there is significant congestion means that the traffic patterns cannot shift significantly from one evaluation period to the next. (We discussed these disadvantages in greater detail under the discussion of the results for the mixed traffic scenario in section 7.2.3.) Thus, the assumptions intrinsic in 3GC are met. The traffic patterns from the last evaluation period *are* good indicators for the next TSC period, and the traffic lights can be viewed as isolated control elements.

The last two data sets have confirmed our earlier suspicions, but not so with the evening rush hour. The results for the evening rush hour runs are displayed below.



The results of the data runs for the evening rush hour scenario are displayed above. The dark bars represent the data from 3GC while the light bars are from PRISTINE. 3GC did better than PRISTINE for this scenario.

Figure 8.11

The numerical results for this scenario are displayed in table 8.8 below.

3GC Wait	PRISTINE Wait	*3GC Stops*	*PRISTINE Stops*
95.475	107.007	*1.838*	*2.681*
67.364	96.121	*1.349*	*2.978*
89.303	98.818	*1.636*	*3.053*
89.56	96.666	*1.759*	*2.978*
83.029	107.06	*1.5*	*2.86*
69.47	81.379	*1.377*	*2.731*
72.502	86.179	*1.168*	*2.556*
62.164	96.896	*1.338*	*2.987*
95.775	121.595	*1.77*	*2.887*
60.068	83.294	*1.239*	*2.782*

This table displays the average stops per vehicle, wait for vehicle experienced and objective value for 3GC and PRISTINE for each of the simulation runs for the evening rush hour scenario. Each row is the result of one data run for each model. Smaller is better. 3GC does better than PRISTINE for this scenario.

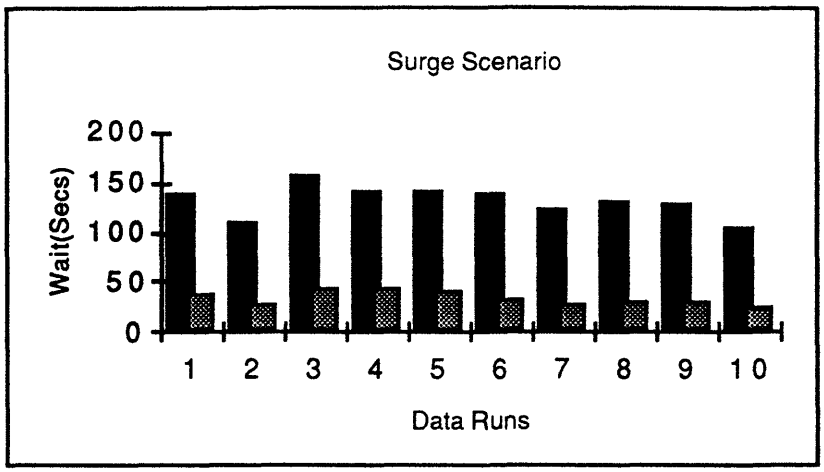
Table 8.8

For the evening rush hour scenario, 3GC again dominates PRISTINE. 3GC produced a mean value of 1.497 stops per vehicle with a standard deviation of .239 across the simulation runs and had a mean waiting time of 78.471 seconds per vehicle with a standard deviation of 13.720 seconds. PRISTINE produced a mean value of 2.849 stops per vehicle with a standard deviation of .159 across the simulation runs and had a mean waiting time of 97.502 seconds per vehicle with a standard deviation of 12.279 seconds.

PRISTINE is again defeated, but why is the evening rush hour any different than the morning rush hour? In actual Boston traffic, there is not as strong a directional component to the data for the evening rush hour. In the morning rush hour, the background traffic at 6 AM is light, and as the commuters begin to superimpose, their West to East pattern dominates. On the other hand, the streets of Boston are hardly dead at 4 PM. In fact there is a reasonably high level of congestion. So, as the commuters begin to make their way from East to West across the Backbay, they are joining the ranks of package deliveries, taxi cabs, fuel truck deliveries and a host of other vehicles. The East to

West patterns are more masked by the ambient business traffic, and overall, the traffic patterns resemble the mid-day pattern far more than an inverted version of the morning rush hour.

The last scenario we tested was our constructed *surge* scenario from chapters one, three and six. We define a *surge* as a sudden increase in traffic flow such that the *average* flow rate over the next TSC period will exceed the capacity of one or more street segments under the *current* signal timing plan but *does not* exceed the *potential* capacity of the street segments. In this particular scenario we introduced a vehicle flow rate along Commonwealth Avenue and Newbury Street that was four times the value of the proceeding background traffic. The surge began thirty minutes into the run and lasted for the remaining thirty minutes of the run; the total run lasted an hour. (The surge was introduced at the intersections of Arlington and Commonwealth and Arlington and Newbury, and the desired routing for the surge traffic was to flow West, i.e. the traffic leaves figure 7.9 at the intersections of Gloucester and Commonwealth and Gloucester and Newbury.) The background traffic flow on all the arcs 6 vehicles per minute. All of the traffic was orthogonal, i.e. there were no turning movements, etc. The surge traffic example results are displayed in figure 8.12 below.



The results of the data runs for the surge scenario are displayed above. The dark bars represent the data from 3GC while the light bars are from PRISTINE. PRISTINE clearly dominated 3GC for this scenario.

Figure 8.12

The numerical results for this scenario are displayed in table 8.9 below.

3GC Wait	PRISTINE Wait	*3GC Stops*	*PRISTINE Stops*
139.28	36.904	*3.225*	*1.281*
108.441	26.572	*2.844*	*1.186*
156.609	43.31	*3.393*	*1.481*
141.783	42.676	*3.241*	*1.348*
140.452	40.465	*3.11*	*1.866*
137.257	33.847	*3.248*	*1.189*
122.709	27.97	*3.043*	*1.089*
129.452	30.71	*3.088*	*1.194*
128.239	29.296	*3.0442*	*1.15*
105.15	24.472	*2.858*	*1.107*

This table displays the average stops per vehicle, wait for vehicle experienced and objective value for 3GC and PRISTINE for each of the simulation runs for the surge scenario. Each row is the result of one data run for each model. Smaller is better. PRISTINE totally dominates for this scenario.

Table 8.9

For the surge scenario, PRISTINE completely dominates 3GC. 3GC produced a mean value of 3.109 stops per vehicle with a standard deviation of .030 across the simulation runs and had a mean waiting time of 130.937 seconds

per vehicle with a standard deviation of 15.723 seconds. PRISTINE produced a mean value of 1.289 stops per vehicle with a standard deviation of .235 across the simulation runs and had a mean waiting time of 33.622 seconds per vehicle with a standard deviation of 6.880 seconds.

In the surge scenario, the relatively light background traffic is overwhelmed by the large surge from East to West while traffic on side streets remains relatively constant. PRISTINE uniformly dominates 3GC through each of the simulation runs. PRISTINE dominates 3GC for two reasons. The primary reason is 3GC's assumption that the last observation period accurately represents the upcoming TSC period is grossly violated. Second, since 3GC is a one street segment look ahead, it necessarily only can accommodate changes one street at a time. If changes occur slowly enough (as they do with the mid-day traffic and evening rush hour examples), the traffic is able to pass through the streets relatively unimpeded using the old settings, but in the surge example, the new traffic patterns are clogged up in the early street segments because the surge *exceeds* the streets' capacities given the TSC plan. Thus, the streets actually act as storage for vehicles and the congestion moves sluggishly forward through the network. In PRISTINE, the TSC strategy knows where the surge traffic wants to go and sets the TSC plan to support this from the start. The traffic flow is accommodated throughout its routing, and the congestion levels never reach the extremes seen with 3GC.

It would be interesting to see how the results change for the surge scenario when the level of the surge changes. The tests above were done for a surge level of four times the previous flow. First, we will explore a variety of surge rates for the example given above. Then, we will look at what happens in a slightly more congested case where 3GC is initially the better strategy. For the case above, we

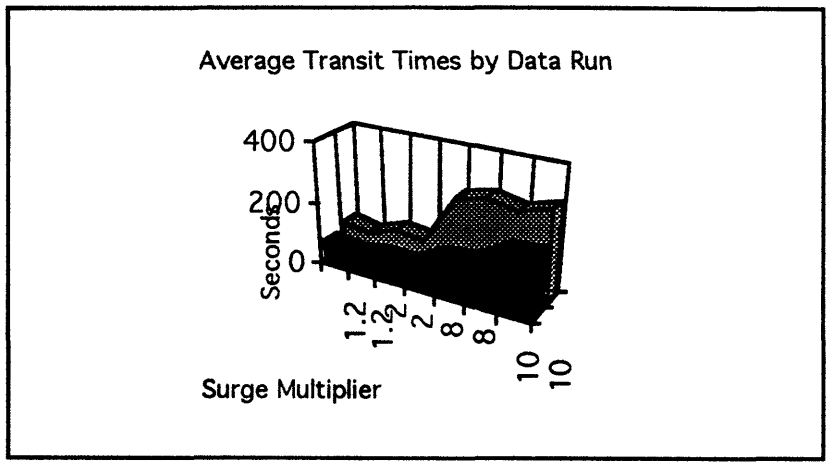
will explore a minimum surge rate of 20% and a maximum surge rate of 1000% increase in traffic flow. Below 20%, the traffic signal control plan is still able to handle the traffic; so, surges below 20% for this network do not fit our definition of a surge. Above 1000%, we exceed the theoretical capacity of the roadway. (As *a point of comparison,* we will run the simulation with *no surge* for both the 3GC and PRISTINE models to better be able to see the effects of the surge on the two traffic signal timing strategies.) In the table below we look at the average wait and stops per vehicle change as we modify the intensity of the surge.

Surge Multiplier	PRISTINE Waits	3GC Waits	PRISTINE Transit	3GC Transit	PRISTINE Stops	3GC Stops
No Surge	18.253	41.502	59.455	80.211	1.051	1.905
No Surge	18.400	36.481	58.859	74.277	0.981	1.830
1.2	21.705	54.460	66.68	95.384	1.099	2.076
1.2	18.862	41.456	61.427	81.013	0.983	1.907
2	21.151	68.919	69.553	112.935	1.089	2.291
2	21.926	61.463	69.72	104.880	1.026	2.146
4 (Base)	33.622	130.937	-N/A-	-N/A-	1.289	3.109
8	50.723	204.17	127.515	261.920	1.642	3.771
8	53.389	223.755	130.988	285.211	1.651	4.034
10	106.306	210.655	193.86	269.079	2.486	3.818
10	117.409	235.419	199.917	298.682	2.502	4.148

This table displays the MOE's for 3GC and PRISTINE for each of the simulation runs for the surge scenario. Each row is the result of one data run for each model with the exception of the row labeled four which is the average of the five earlier runs. Smaller is better. PRISTINE totally dominates for this scenario.

Table 8.10

PRISTINE dominates 3GC for each of the data runs. Note for these supplemental runs, we also obtained the average transit time through the network. The results are displayed below in figure 8.13.



The lighter colored area represents the 3GC results. PRISTINE provides uniformly better average transit times than 3GC. This is a confirmation of the results displayed in table 8.10.

Figure 8.13

Throughout the range from the near light traffic example (where the traffic only surges by 20%) to near capacity (where the traffic flow rate increases by a factor of ten), PRISTINE does very well. As the increase in flow goes to ten times the background traffic, we begin to reach the capacity of the roadway.

As a point of comparison, we also examined a surge scenario where, in the base case, 3GC dominated PRISTINE. Recall that 3GC tends to dominate PRISTINE when there is a fairly high level of congestion in the network and no clear dominant directions of flow. Unless otherwise noted, the conditions for this surge scenario were identical to the last surge scenario. We selected a situation where the ambient traffic flow was 10 vehicles per minute on every arc in the network. As in the last surge scenario, all traffic in this example was orthogonal; this allowed us to filter out the effects of delays due to left turns, etc. from the data. (We note that the average stops and delay per vehicle under the base case in these data runs are *significantly* lower than those found in a similar level of congestion for the mid-day traffic scenario where vehicles frequently made right and left turns onto congested roadways.)

Surge Multiplier	PRISTINE Waits	3GC Waits	PRISTINE Stops	3GC Stops	PRISTINE Transit	3GC Transit
No Surge	39.334	30.635	2.156	1.263	131.420	119.549
No Surge	37.655	24.640	2.096	1.037	129.327	117.946
2	33.221	34.925	2.093	1.346	128.953	137.918
2	32.576	45.176	2.028	1.474	126.266	147.878
4	48.213	65.005	2.463	2.050	159.085	184.031
4	38.291	71.960	2.373	1.956	157.694	185.922
6	53.113	83.344	2.497	1.626	164.173	197.212
6	51.983	77.044	2.141	1.788	180.735	194.408

This table displays the MOE's for 3GC and PRISTINE for each of the simulation runs for the surge scenario. Each row is the result of one data run for each model Even though 3GC performs better for the base scenario (no surge), PRISTINE performs better than 3GC as the surge multiplier increases. The roadways are reaching saturation in the runs with a multiplier of six.

Table 8.11

Again, under the surge scenario PRISTINE performs significantly better than 3GC for the MOE's of average wait per vehicle and average transit time per vehicle. It is interesting to note that PRISTINE actually *decreases* the average stops per vehicle and transit time per vehicle when the surge rate of flow on Commonwealth and Newbury increases from 10 to 20 vehicles per minute. It seems counterintuitive until we recognize that PRISTINE has a clear dominant direction of flow when the flow rate on the two major East to West streets becomes 20 vehicles per minute. Also, the predictive routing information projects well onto this scenario, because there are no turning movements. So, even if congestion on the streets causes vehicles to slow down, the vehicles ten minutes later will still be going the same direction. Some of the benefits of PRISTINE's benefits are masked in this scenario, because the background flow rate is so high that the two major East to West thoroughfares only make up a fraction of the total flow in the network. So, our average values for the MOE's disguise some of the benefit of using the predictive routing information.

We also wanted to see what effects the *duration* of the surge had on the MOE's. In our next scenario, the surge has exactly the same characteristics as the first surge scenario with the light traffic background, but this time the surge only lasts for ten minutes. The simulation run lasts for an hour. The first half hour has light traffic. This is followed by a ten minute surge, and the last twenty minutes go back to the light traffic background. Our results for these runs are displayed below.

Multiplier	PRISTINE Wait	3GC Wait	PRISTINE Transit	3GC Transit	PRISTINE Stops	3GC Stops
1.2	17.768	42.824	59.04	82.623	1.017	1.923
1.2	18.135	38.799	59.133	78.025	0.954	1.861
2	20.385	50.413	68.727	70.645	1.063	2.034
2	26.535	68.839	51.057	90.882	1.092	2.135
4	24.944	90.792	64.218	122.879	1.265	2.49
4	34.791	102.178	91.817	135.77	1.271	2.619
8	40.245	97.696	106.454	133.807	1.406	2.638
8	40.606	108.476	107.357	144.412	1.355	2.712
10	37.215	98.92	103.373	145.726	2.342	2.664
10	39.91	109.534	109.054	154.425	2.387	2.705

This table displays the MOE's for 3GC and PRISTINE for each of the simulation runs for the ten minute surge scenario. Each row is the result of one data run. Smaller is better. PRISTINE totally dominates for this scenario.

Table 8.12

We also ran a scenario using a twenty minute surge. In this case, there was: a half hour of light traffic, a twenty minute surge, and it ended with ten minutes of light traffic. The results for these runs are shown below.

Multiplier	PRISTINE Wait	3GC Wait	PRISTINE Transit	3GC Transit	PRISTINE Stops	3GC Stops
1.2	17.94	34.656	60.882	70.403	0.989	1.848
1.2	16.153	28.103	54.379	62.726	0.952	1.747
2	17.372	46.199	60.042	85.206	1.048	2.075
2	31.741	49.953	72.098	88.428	1.237	2.162
4	23.396	68.315	75.784	110.36	1.178	2.44
4	24.19	81.411	78.688	127.005	1.099	2.726
8	54.886	125.531	121.602	174.266	1.815	3.254
8	39.515	135.76	106.732	187.891	1.423	3.426
10	69.175	136.893	133.216	196.602	2.476	3.325
10	82.639	146.44	151.971	209.549	2.055	3.508

This table displays the MOE's for 3GC and PRISTINE for each of the simulation runs for the twenty minute surge scenario. Each row is the result of one data run. Smaller is better. PRISTINE totally dominates for this scenario.

Table 8.13

We see in the ten minute scenario that 3GC does better relatively than it did in the original thirty minute surge scenario. This is because 3GC is able to react to the surge (in a limited fashion) after the first ten minutes. Although 3GC's reaction comes after the surge traffic has entered the network, it still will alleviate some of the follow on congestion. Additionally, the surge traffic makes up a much smaller proportion of the total traffic which flows through the network, because the surge only lasts ten minutes.

The results of the twenty minute scenario are very similar to the original thirty minute surge, and the original comments found in the section on the thirty minute surge (see above) apply here too.

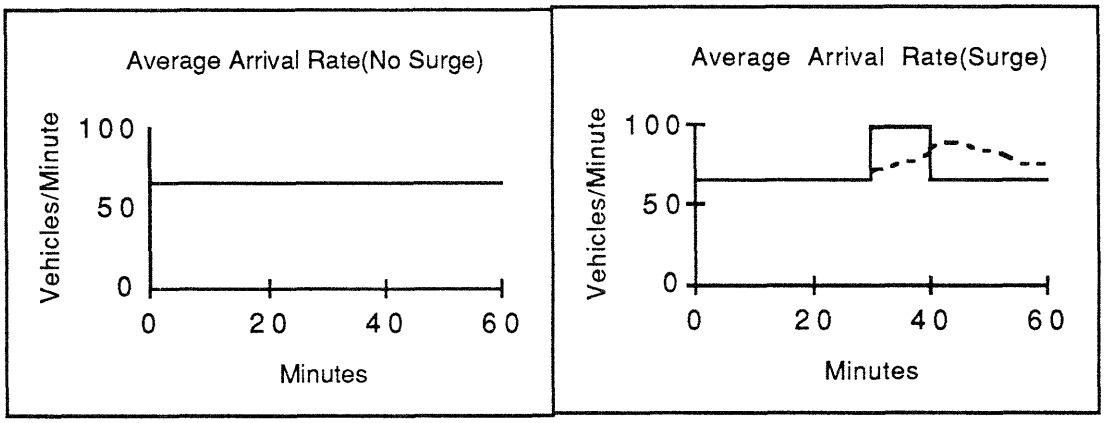
SECTION 8.3 Further Analysis of Surges

The discussion in section 8.2 underestimates the advantage of PRISTINE over 3GC for surges. The reason is that, under 3GC (and to a lesser extent under PRISTINE), some of the vehicles in the surge are prevented from entering the network when they wish because of the congestion created by the surge. The simulation does not record the vehicles' delay while they are still in parking lots but only starts counting it when they enter the network. Additionally, the earlier results averaged a pre-surge period of 30 minutes (included in the scenarios to bring the network to steady-state before introducing the surge) with the surge period, thereby reducing its apparent effect.

In this section we adjust the results of the simulation for several representative data runs to take these phenomena into account. The analysis will be approximate but should capture the main effects and demonstrate the enhanced advantage of PRISTINE. We shall use transit time as the measure of effectiveness in this work.

Consider what happens in the case of the ten minute surge. We will examine the average arrival rate, number of vehicles in the network, and cumulative arrivals over the course of a sixty minute simulation period. For this example, we will assume the surge to last ten minutes during which time the average vehicle arrival rate on our two major Westbound arterials, Commonwealth and Newbury (see figure 7.9), will jump from 6 vehicles per minute to 24 vehicles per minute.

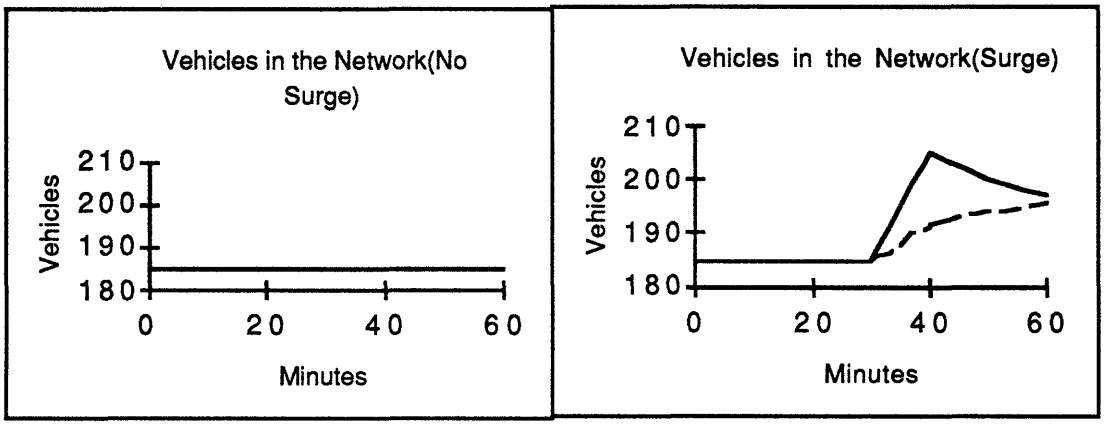
First we will examine the average arrival rate. For the no surge case, we have an average vehicle arrival rate of 6 vehicles per minute at each of eleven nodes for a total network average arrival rate of 66 vehicles per minute. This remains constant across the entire sixty minute period. In the case of the surge, the average arrival rate across the network is initially 66 vehicles too, but from the 30 minute mark until the 40 minute mark, the network arrival rate jumps to 99 vehicles per minute. Then, the average arrival rate drops back to 66 vehicles per minute. In the simulation some of the vehicles which are generated at the nodes may be unable to enter the network, because the street segments ahead are clogged. Thus, the actual average arrival rate in the simulation falls below the projected rate. We will say the vehicles unable to enter the network are trapped in parking lots to add a physical intuition to the problem. The results are displayed in the two figures below.



In the no surge case, the average vehicle arrival rate remains constant while in the surge case, the average vehicle arrival rate increases for ten minutes. The dashed curve in the surge case represents the number of vehicles that actually enter the network. (Some vehicles that are generated at the nodes are unable to enter the simulation because the streets are full.)

Figure 8.14

The number of vehicles in the network is fairly constant for the no surge case, but in the surge case, the number of vehicles in the network increases as the surge occurs. In cases where the traffic control system is unable to compensate for the increased traffic flow, the number of vehicles in the network falls short of the projected number.



For the no surge case, the number of vehicles in the system is fairly constant. For the surge case the top (solid) line represents the number of vehicles both in the network and waiting in parking lots. The bottom (dashed) curve does not include the vehicles which are in the parking lots. The simulation does not start tracking a vehicle's time in the network until it actually enters the first street segment..

Figure 8.15

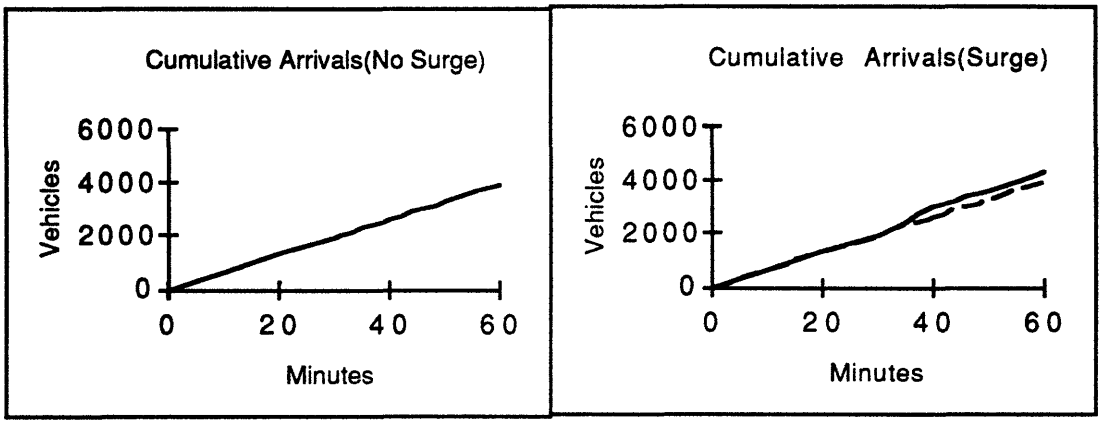
The actual number of vehicles we would expect to enter the network during the sixty minute simulation run are shown below in table 8.14.

Surge Multiplier	10 Minute Surge	20 Minute Surge	30 Minute Surge
No Surge	3960	3960	3960
1.2	3984	4008	4032
2	4080	4200	4320
4	4320	4680	5040
8	4800	5640	6480

This table displays the total expected number of vehicles to arrive during the sixty minute simulation period for each of the surge durations and surge multipliers. The numbers shown include both the vehicles which enter the simulation and those vehicles which are stuck in parking lots.

Table 8.14

The cumulative arrivals for the network are also affected by the surge. In figure 8.15 below we show the cumulative arrival rate for both the no surge and surge cases.



In the no surge case, the vehicles arrive at a constant rate. In the surge case, the top (solid) line represents the cumulative arrivals to the simulation (including the parking lots) while the dashed (bottom) curve represents only the vehicles which actually enter the network. The area between the two is the time spent by vehicles in the parking lots before they enter the network. (We will refer to this as the "lost area".)

Figure 8.16

In the figure above, the area between the solid line and the dashed curve represent the total time spent by vehicles in parking lots before they entered the network. In the table below, we display the cumulative number of vehicles which entered the network in each of the 30 simulation runs displayed below.

Surge Multiplier	10 Minute Surge		20 Minute Surge		30 Minute Surge	
	PRISTINE	3GC	PRISTINE	3GC	PRISTINE	3GC
No Surge	3975	3961	3975	3961	3975	3961
1.2	4001	3967	4032	3987	4035	3998
2	4085	3840	4203	3887	4284	3964
4	4331	3959	4637	4098	4690	4271
8	4620	4410	4656	4410	4656	4410

The table above shows the cumulative number of vehicles which entered the network for each of the simulation runs show. Notice that PRISTINE consistently allows more vehicles to pass through the network than 3GC. Also note, by the time the multiplier reaches eight, the network is saturated.

Table 8.15

PRISTINE consistently allows more vehicles to pass through the network than 3GC. By the time the surge multiplier reaches eight, the network is saturated. We will calculate the missing waiting time spent by the vehicles in the parking lots and subtract off the pre-surge period's effects for the surge multipliers of 1.2, 2, and 4.

Before we can complete the pre-surge subtraction and calculate the lost waiting time, it will be necessary to introduce some new notation.

We will begin by examining the base case, i.e. no surge.

Let $n_b(t_1, t_2)$ = number of departures from the network between times t_1 and t_2. For example, $n_b(0,60)$=3960.

We assume that $n_b(0,30) = n_b(30,60) = (\frac{1}{2})n_b(0,60)$. (That is, we assume we are at steady-state for the base case.)

Let $\tau_b(t_1, t_2)$ = the average transit time for the departures between times t_1 and t_2.

We assume that $\tau_b(0,30) = \tau_b(30,60) = \tau_b(0,60)$, i.e. we are at steady-state.

Let $T_b(t_1, t_2)$ = total transit time spent in the system by vehicles departing between times t_1 and t_2.

We assume

$$T_b(0,60) = n_b(0,60)\tau_b(0,60) =$$

$$n_b(0,30)\tau_b(0,30) + n_b(30,60)\tau_b(30,60)$$

For the surge case, we use the same notation, but will use the subscript "s". As an overview, for the first thirty minutes of a surge scenario (i.e. pre-surge), the vehicles flow smoothly through the network. We will designate the first thirty minute period as (0,30). During (0,30) virtually all of the vehicles generated are allowed to enter the network. During (30,60), we need to differentiate between the transit time seen by (1) departures and cars recorded as departures (i.e. the simulation records the vehicles still in the network at the end of the simulation as departures, typically less than .5% of the vehicles which passed through the network during the simulation) and (2) the same *plus* the vehicles caught in parking lots.

Now, $\mathbf{n_s(30,60)}$ = the number of departing (and existing) cars seen by the simulation during the surge period (30,60).

Let, $\mathbf{n_s'(30,60)}$ = the estimated departing (and existing) cars seen by the simulation *plus* the vehicles still in the parking lot during the surge period (30,60).

Our estimates of $\mathbf{n_s'(0,60)}$ are shown in table 8.14.

We assume $\mathbf{n_s'(30,60) = n_s'(0,60) - n_s'(0,30)}$ and

$\mathbf{n_s(30,60) = n_s(0,60) - n_s(0,30)}$ where

$\mathbf{n_s(0,30) = n_s'(0,30) = (\frac{1}{2})n_b(0,60)}$.

Let $\mathbf{T_s'(30,60)}$ include total time spent in the system by all vehicles generated by the simulation during the period (30,60) including exiting vehicles, vehicles still in the network at the end of the scenario and vehicles still caught in parking lots.

In words, we could describe $T_s'(30,60)$ as (equal to)

= total time for vehicles recorded by the simulation + time spent by vehicles in the parking lots before entering the network ("lost area" described in figure 8.16)

which is (equal to)

$= n_s(30,60)\tau_s(30,60)$ + lost area.

We would like to determine $\tau_s'(30,60)$. We can calculate $\tau_s'(30,60)$ using terms we have already defined. Specifically,

$$T_s'(0,60) = n_s'(0,60)\tau_s'(0,60)$$
$$= n_s'(0,30)\tau_s'(0,30) + n_s'(30,60)\tau_s'(30,60)$$

By assumption, we have $n_s(0,30) = n_s'(0,30) = (\frac{1}{2})n_b(0,60)$. Substituting we obtain

$$T_s'(0,60) = (\frac{1}{2})n_b(0,60)\tau_b(0,60) + n_s'(30,60)\tau_s'(30,60).$$

Finally, we have $\tau_s'(30,60) = \dfrac{T_s'(30,60)}{n_s'(30,60)}$.

Using this methodology, we calculate the surge period average transit time per vehicle. (We approximate the bottom (dashed) curve in figure 8.16 for the surge scenario as a straight line. Given this approximation, the lost area becomes a quadrilateral, and we integrate over time to find the lost area's value.) The results are displayed in the table below.

Surge Multiplier	10 Minute Surge		20 Minute Surge		30 Minute Surge	
	PRISTINE	3GC	PRISTINE	3GC	PRISTINE	3GC
No Surge	59.455	80.211	59.455	80.211	59.455	80.211
1.2	58.633	85.027	58.640	60.727	62.259	110.271
2	43.157	102.241	60.565	90.392	78.243	145.593
4	68.229	165.569	87.952	138.545	-N/A-	-N/A-

The estimated average transit time per vehicle of all the vehicles (including the time spent in parking lots) during the surge period is shown in the table above. PRISTINE's absolute advantage in the area of average transit time per vehicle is magnified by subtracting the pre-surge contribution and adding the time spent waiting in parking lots before entering the network. Raw figures are shown in tables 8.10, 8.12 and 8.13.

Table 8.16

In each case, PRISTINE dominates 3GC in the area of average transit time per vehicle. The lowest savings demonstrated by PRISTINE is in the case of the 20 minute surge with a multiplier of 1.2 where PRISTINE only saves approximately 3.4%. The largest savings demonstrated by PRISTINE displayed by the table occur in the case of the 10 minute surge with a multiplier of two where PRISTINE saves 57.8% on the average transit time per vehicle offered by 3GC under the same circumstances. We varied two parameters of the surge, its length and its multiplier. The *ratio* of the corrected average transit time (ATT) for 3GC compared to that for PRISTINE tends to *increase* as the surge multiplier *increases*. For example in the base case for the ten minute surge, the ratio of ATT using 3GC to that of PRISTINE was 1.35 while with a multiplier of 4 that same ratio increases to 2.43. Again looking at the ratio of the corrected ATT, 3GC seems most adversely affected under the ten minute case. With a multiplier of two, the ratio of the ATT using 3GC to the same under PRISTINE was 2.37 for the ten minute surge but only 1.86 under the thirty minute case. This is logical, because 3GC has the chance to begin correcting the traffic signals to accommodate the increased flow under the longer surge scenarios.

SECTION 8.4 Conclusions

In this section we developed an optimized third generation control model, 3GC, and compared it to PRISTINE. PRISTINE and 3GC were compared using six sets of data on the Lin-Sarkar-Staats simulation model. PRISTINE (using the predictive routing information available under IVHS) did better than 3GC, the optimized third generation traffic signal control strategy, in cases where PRISTINE's basic assumptions about the traffic held, and 3GC did better than PRISTINE in scenarios where PRISTINE's assumptions were less accurate. For example we found in the mid-day traffic and evening rush hour scenarios PRISTINE's assumption that the predictive routing information would accurately reflect the actual traffic flows occurring in the network over the next traffic signal control period did not hold. The intersections could be viewed as independent entities with no large degradation in the traffic signal control performance; so, 3GC did well when compared to PRISTINE in these cases. Conversely, 3GC's assumption of steady state conditions did not apply well in the morning rush hour or the surge scenarios. These cases will be examined in more detail below.

To aid in our analysis of 3GC and PRISTINE's performance, table 8.14 is printed below. It contains descriptive statistical summaries for 3GC and PRISTINE for each of the four non-surge scenarios.

Scenario	3GC Average Stops(SD)	PRISTINE Average Stops(SD)	3GC Average Wait(SD)	PRISTINE Average Wait(SD)
Early Morning	1.244(.072)	1.577(.394)	49.107(4.377)	24.212(4.322)
AM Rush Hour	1.549(.235)	2.113(.327)	79.547(25.378)	50.0351(7.239)
Mid-Day Traffic	1.723(.292)	2.967(.264)	91.772(19.452)	111.396(35.316)
PM Rush Hour	1.497(.030)	2.849(.159)	78.471(13.720)	97.502(12.279)

The table above contains the summary statistics for the four non-surge scenarios used to test PRISTINE against the optimized third generation control model (3GC).

Table 8.17

The early morning traffic scenario showed a savings of nearly 25 seconds average waiting time for the PRISTINE model over the 3GC model. PRISTINE was able to better control the traffic signals in this scenario, because PRISTINE uses average flow rates to calculate its signal timing plan. Whereas 3GC uses actual vehicle arrivals to calculate its timing plan. When traffic is very light, a single arrival can make a big difference in the timing plan for 3GC, and during the early morning scenario, 3GC tended to overcompensate.

For the morning rush hour scenario, PRISTINE waiting time per vehicle averaged 37% below the waiting time for 3GC. Throughout the morning rush hour, there is a preferred direction of travel, West to East, and the traffic flow builds up steadily from 7 AM till 8:45 AM, and even then, it only tapers off a little. This is precisely the type of situation where PRISTINE *does* well. The spanning tree heuristics can set up progression schemes on the inbound, West to East, routes. At most intersections there is a *dominant flow direction.*, and PRISTINE takes advantage of dominant flows by setting the splits to accommodate the greater volume of traffic for the approaches *sharing a common green cycle.*. Such dominant directions and patterns do not exist in the mid-day traffic scenario.

In the mid-day traffic scenario, the 3GC model posted a savings of over 20 seconds from the average wait experienced by PRISTINE for this scenario. The reason for this can be traced back to the chaotic traffic conditions that exist in the roadway during this time frame. We generally see random patterns at the intersections. *There is no clearly dominant direction of travel at each intersection or even a majority of them.* Plus, there are high levels of congestion. Under these conditions about the best one can do is increase the green time in *both* directions and try to *maximize the capacity* of the network. In effect, we can consider each individual intersection as an independent optimization process where the goal is

maximum capacity rather than attempting to minimize average stops or wait. The situation was much the same under the evening rush hour scenario.

The 3GC model saved 19.5% on the waiting time experienced under PRISTINE for the evening rush hour scenario. The reason for this is exactly the same as the rationale for 3GC's superior performance in the mid-day traffic model. There is not as strong a directional component to the data for the evening rush hour. In the morning rush hour, the background traffic at 6 AM is light, and as the commuters begin to superimpose, their West to East pattern dominates, but for the evening rush hour this is not the case. The streets of Boston are not deserted at 4 PM. In fact there is a reasonably high level of congestion. As the commuters begin to make their way from East to West across the Backbay, they are joining the ranks of package deliveries, taxi cabs, fuel truck deliveries and a host of other vehicles. The East to West patterns are more masked by the ambient chaotic business traffic. In the case of the surge scenario, there was very little interference from ambient traffic, and PRISTINE did very well.

The best performance from PRISTINE came in the surge scenarios. We examined a base case scenario where there was a sudden four fold increase in traffic along Commonwealth and Newbury moving from East to West. The large savings experienced for this scenario prompted us to explore the parameters of the surge more closely. We allowed the surge increase to range from only 20% (hardly a surge at all) to a ten fold increase (reached the capacity of the roadways). In each case PRISTINE demonstrated *significant* savings over 3GC. We also examined a scenario where 3GC initially defeated PRISTINE. Recall that 3GC tends to beat PRISTINE under congested conditions where there is no dominant direction of flow. In this case, we examined a smaller range of surge values, because we ran into roadway capacity constraints sooner. Again though,

PRISTINE demonstrated the value of using predictive routing information in setting traffic signals.

Traditional methods must rely on information collected in real time, but if something is *about* to happen then traditional methods are at the mercy of circumstance. Like all third generation control methods, 3GC assumes that the arrivals seen during last observation period accurately represent the arrival patterns to be experienced during the upcoming traffic signal control period. Under the surge scenario, this assumption is grossly violated. But, 3GC really cannot compensate *even after the traffic flow dramatically increases*. Since 3GC is a one street segment look ahead, it necessarily only can accommodate changes one street at a time. In the surge example, the new traffic patterns clog up the first street segments in their paths, because the surge *exceeds* the streets' capacities given the traffic signal control plan. Thus, the streets actually act as storage for vehicles and the congestion moves sluggishly forward through the network. In PRISTINE and IVHS traffic signal control strategies in general, the traffic signal control strategy *knows where* the surge traffic *wants* to go and is able to set the traffic signal control plan to support this from the start. The traffic flow is accommodated throughout its routing, and the congestion levels never reach the extremes seen with 3GC.

The greatest contribution from PRISTINE and traffic signal control systems that will be integrated into IVHS will come from their ability to react in real time to the *unexpected*, non-historically arising traffic situations. As we saw in the surge example, anticipatory traffic signal control can have huge effects in reducing congestion and promoting traffic flow. We will revisit our findings in the next chapter.

Chapter Nine

Conclusions

Man lives on a placid isle of ignorance in a vast, black ocean, and it was not meant that we should venture far!

-H.P. Lovecraft

SECTION 9.1 Overview

In this thesis we explored the potential benefit from integrating predictive routing information with dynamic traffic signal control. In this final chapter, we summarize our general findings and offer additional areas of research uncovered during our investigations which may prove to be fruitful areas further research. We found that predictive routing information could be *highly beneficial* in cases where *sudden* changes in traffic flow caused congestion. Under such conditions, the Predictive Routing Information Signal Timing INtEgration (PRISTINE) system, the IVHS traffic signal control strategy, was able to *cut delays by significant amounts* from what was experienced using a system which optimized the signal settings using information which is currently available. The benefits of using predictive routing information were not as great when traffic conditions changed gradually, and in some cases there was no benefit. We found the greatest benefits from using the predictive routing information came in cases

where there were dominant directions of flow in the network. We will begin by summarizing our findings.

SECTION 9.2 Findings

In this section we will review the findings of the thesis by chapter. The results printed here are summary, and the interested reader is encouraged to read the applicable chapters for further detail.

The thesis set out to determine the effects of integrating predictive routing information with dynamic traffic signal control. The goal of this thesis was to determine the *first order* effects of using predictive routing information in traffic signal control. To a large extent, this thesis took a very conservative approach to the availability of traffic information available under IVHS. We examined the use of predictive routing information in traffic signal control; this is only one component of the information that will be available when IVHS is fully implemented. We established a general framework for the evaluation of traffic signal control systems. The methodologies described in chapters three, four and five could be applied to any traffic control system and are applicable even if the predictive routing information under IVHS never materializes. We developed PRISTINE in chapters six and seven, and in chapter eight we used a simulation to compare PRISTINE and an optimized third generation traffic signal control system (3GC) using traffic data which is currently available.

In chapter four we combined aspects of network flow, graph and traffic signal control theory into the *spanning trec concept* for sequencing the green lights within the network. (This process of synchronizing the lights is called setting the

offsets.) The spanning tree concept is the notion of selecting a connected set of directed street segments from the entire network in such a way as to form a spanning tree. A spanning tree is a set of connected, directed street segments selected in such a way that the set of street segments include every intersection, and there is a *unique* path between every set of intersections using the spanning tree.

We developed the concept of defining the arterials in the network based on the spanning tree we derived. Traditionally, one selects a major thoroughfare or several important streets or avenues as the arterials for an urban traffic grid, but under our methodology, the arterials become those *street segments* which we plan on *coordinating* the offsets for. The street segments in the spanning tree are not restricted to be along major thoroughfares. In fact, the spanning tree *does not* have to include a street from start to finish; the spanning tree contains street *segments*. (Thus, the spanning tree can follow vehicle routes and turn from street to street at the intersections.) Our *only* restrictions are that the spanning tree must touch every intersection, and there must be a *unique* path between every pair of intersections through the spanning tree.

We were able to demonstrate that under ideal circumstances, a traffic manager would be able to synchronize all the lights under his or her control in all directions, but in a network with **n** traffic signals, the traffic manager would *always* be able to synchronize *at least* **(n-1)** street segments. We went on to show that those **(n-1)** street segments would cover at least one quarter of the traffic in the network under even the most unfavorable conditions.

We developed three methods of selecting the spanning tree. The first method was called the Route Augmentation (RA) heuristic. The idea behind the RA heuristic is that we should select the spanning tree to cover the busiest routes first. In section 4.2, we showed that in the area of traffic signal control, intuition occasionally fails. For example, it seems like a reasonable idea to set the offsets in a network in such a way as to favor the most heavily traveled routes (i.e. use the RA heuristic), but it is possible to construct a case where this *reasonable* procedure leads to the worst possible average delay per vehicle and average stops per vehicle in a network.

The Maximal Spanning Tree (MST) heuristic uses the aggregate flow rates to determine the spanning tree. We showed in section 4.3 that the arcs of the spanning tree selected using this method will contain a *minimum* of one fourth of the total flow in the network. Under more favorable conditions, where there are dominant directions of flow through the network, the MST heuristic will capture a much larger fraction, and under ideal circumstances the MST heuristic could capture all of the flow in the network. Because the MST uses the aggregate flow rates on the street segments, it could be implemented using existing traffic data.

The last method we developed for determining the spanning tree was called the Potential Function (PF) heuristic. The PF heuristic considered the length of the routes as well as the flow rates in determining the spanning tree. All things being equal, we would rather coordinate the longer routes through the

network than a series of shorter routes. We showed that the RA and MST heuristics were special cases of the PF heuristic.

In chapter five our purpose was to develop general methods of calculating the effects of traffic signal control settings in terms of average stops per vehicle and average delay per vehicle. (In chapter five we considered delay to be the amount of time a vehicle was stopped at a traffic light. A vehicle stopped if it was either stopped by a red light or unable to pass through an intersection because of congestion.) We began by looking at the light traffic model. Under the light traffic model queuing effects are unimportant. Vehicles only have to stop in the network if the upcoming traffic signal is red, and all vehicles stopped at an intersection during a red light will be able to proceed during the next green light. The light traffic model is the most widely used model in traffic signal control literature, and we used the light traffic model to gain intuition into the effects of changing splits on the average stops per vehicle and average delay per vehicle. We also demonstrated that using stops per vehicle as the primary measure for a traffic signal control plan could result in undesirable traffic settings.

Using the light traffic model assumptions, we developed a method of evaluating traffic signal settings in terms of expected stops per vehicle and expected delay per vehicle. This methodology could be used under any circumstances where queuing effects are unimportant.

However, queuing effects are generally important in traffic signal control, and we established a Queue Effects Model (QEM) that considered the effects of congestion and bulk arrivals on average wait per vehicle and average stops experienced per vehicle at an intersection. The Queue Effects Model expands the earlier work of Cedar (1989), but where Cedar used an M/G/1 queuing system, we allow for bulk arrivals. The Queue Effects Model is able to predict upper bounds on both the average stops and wait a random vehicle will experience when approaching an intersection using only average vehicle flow rate and the first two moments of the platoon size distribution. We derived an approximation for Z-transform of time average queue length.

The best possible situation in a traffic flow environment is if **no one** encounters **any** red lights and passes directly through the network. We can mathematically approximate this by allowing every vehicle to shrink to an infinitesimal point and *turn all of the traffic lights green.* Under these assumptions, the urban traffic flow turns into a network multi-commodity flow problem. We could view every vehicle *flowing* from a particular origin to a particular destination as being composed of a specific fluid. We have a source, the route's origin, and a sink, the route's destination, corresponding to each route through the network. We retain the capacity limits on the arcs in terms of vehicles per unit time; this is a function of the speed limit and the number of lanes on the street segment. As a cost per arc, we will use the amount of time required to traverse it at the speed limit. The solution to this flow problem gives the theoretical lower limit on the average travel times through the network.

One of our stated goals in the thesis was to develop a methodology for explicitly utilizing the predictive routing information available under IVHS in a dynamic traffic signal control environment. In chapter six, we integrated the spanning tree method of setting offsets and the Queue Effects Model into a traffic signal control strategy, PRISTINE, which explicitly uses the predictive routing information available under IVHS. It is important to note that we applied the predictive routing information very rigidly. We assumed that drivers would follow their projected routings through the network *even if* the street segment ahead appeared to be congested. In reality, some non-trivial fraction of the drivers would desert their projected paths and take new routings from the point of congestion onward.

We explored two methods of generating the splits and cycle time for PRISTINE in chapter six. One involved the use of a non-linear program, and the other, a heuristic approach, exploited the structure of the problem to provide a solution after only two passes. Both methods use the Queue Effects Model to represent the average delay per vehicle and average stops per vehicle for the street segments which are not part of the spanning tree.

The non-linear program has an objective function that is both discontinuous and piece-wise convex. Traditional techniques for non-linear programs do not offer ready solutions to these problems. In chapter seven we turned to a gradient descent method utilizing pseudo-gradients to allow the solver to work with the discontinuities. We used a limited line search technique, barrier functions, and projection methods to limit the solver's chance of being

caught in a local, non-global minimum. Our methods produced a non-linear program solver which generally converged to a solution in less than 1,000 iterations.

To evaluate PRISTINE, we developed a traffic simulation grounded on the earlier works of Lin (1992) and Sarkar (1993). The simulation is able to integrate predictive routing information and implements traffic signal control plans while still operating the simulation. (The simulation is capable of varying its traffic signal control plan in mid-simulation run without loss of data, i.e. vehicles continue on within the simulation from their current locations while responding to the new traffic signal control plan.) We call this last feature the capability for *dynamic interface.*. Additionally, the traffic simulation integrates key real world aspects into its routines such as reasonable acceleration/deceleration performance, platooning and vehicle safe driving clearances.

In chapter seven, we used the simulation to test the performance of the non-linear program versus the heuristic approach for setting splits in PRISTINE. We found that under all but light traffic conditions, the non-linear program had significantly better performance in both the areas of stops per vehicle and average wait per vehicle than the heuristic approach. Under one scenario tested, the non-linear program decrease average wait by over 30%. The non-linear program was able to converge to an optimal solution very quickly, typically taking less than 800 iterations before selecting the optimal traffic signal setting. We elected to use the non-linear program to test the performance of PRISTINE versus our optimized third generation traffic control strategy, 3GC.

There are many third generation traffic signal control software packages available in the commercial sector, but due to pragmatic concerns such as funding the purchase of these packages and the time it would take to modify the packages, we elected to develop our own third generation control system to compare PRISTINE against. The strategy integrated key elements of third generation control such as: look ahead, minimization of projected stops and wait per vehicle, and the ability to vary not only the splits and offsets but also the cycle time for each intersection.

Our third generation control system (3GC) offers three major advantages over most commercial packages which will make it a more competitive opponent for PRISTINE than most commercial packages would be. First, since commercial packages must compute traffic signal settings in real time, they often are not allowed to iterate until they find an optimal solution. We allowed 3GC to run until it determined its optimal solution. Second, 3GC has access to *far* more information than any comparable commercial system. 3GC records *every vehicle* arrival and departure from *every street segment* in the network. Third, although third generation control systems have the ability to vary the cycle times by intersection, in practice many of them consider the cycle time to be fixed while operating in the field (e.g. SCOOT) (Homburger and Kell, 1988); we *will allow* the cycle time to vary by intersection for 3GC.

We compared the performance of PRISTINE versus 3GC using five scenarios with multiple runs for each scenario. We used actual traffic data, collected by the traffic sensors in the Backbay, to motivate the first four scenarios (e.g. early morning, morning rush hour, mid-day traffic and evening rush hour). The fifth scenario was the surge scenario which we examined under a variety of conditions. Our conclusions from this area of research are included in the next section.

SECTION 9.3 Conclusions

Based on our research, we have come to several overall conclusions. (We will denote each of our major conclusions by the use of a bullet, "•".)

• The first of these is that, although predictive routing information (as used by PRISTINE)*can* offer large savings in the areas of delays and stops per vehicle, it is not *always* helpful. Our simulation runs showed that PRISTINE, using the predictive routing information available under IVHS, was able to substantially decrease the average wait per vehicle in two cases.

First, PRISTINE performed better than 3GC in scenarios where there were unanticipated, sudden increases in traffic flow. We call these circumstances "surge situations". (We define a *surge* as a sudden increase in traffic flow such that the *average* flow rate over the next traffic signal control period will exceed the capacity of one or more street segments under the *current* traffic signal settings but *does not* exceed the *potential* capacities of the street segments.)

Second, PRISTINE did well in cases where there were *dominant directions of flow* in the network. (We will discuss dominant directions of flow in more detail below.) By a dominant direction of flow, we mean *a clear majority of the flow enters an intersection from a particular direction.* A set of dominant flows was present in the Boston morning rush hour scenario, and PRISTINE cut about one-third off of the delay experienced under the third generation control system. (The definition of delay we used in our simulation is the total time a vehicle is traveling at less than 5 MPH in the network. This measure captures not only full stops but also times when a vehicle is creeping along. These tend to be the times of greatest driver frustration and the times of lowest fuel efficiency and greatest emission of pollutants.) For the surge scenario, described above, PRISTINE consistently outperformed the third generation control system.

Just as important as understanding when predictive routing information will be valuable, it is important to know when such information will not be of substantial benefit. Under conditions of heavy congestion when there are not dominant directional flows in the network about the best one can do is increase the green time in both directions and try to *maximize the capacity* of the network. In effect, we can consider each individual intersection as an independent optimization process where the goal is maximum capacity rather than attempting to minimize average stops or wait. (Every time the light shifts from red to green and green to red, there is some effective green time lost due to accelerations/decelerations and the time needed to clear the vehicles in the intersection when the light changes color. By lengthening the splits, you cut down the fraction of the green time lost due to these factors.) There are several traffic signal control methods which do well under congested conditions without

dominant directions of flow. For example, MAXBAND and MULTIBAND maximize the green bands on the arterials for the network and allow the greatest throughput in times of heavy congestion.

In both the mid-day traffic and evening rush hour scenarios for the Boston data, the third generation control system outperformed PRISTINE by saving approximately 18% on the average delay per vehicle. In the mid-day traffic scenario, there is a large amount of congestion on the streets, but there are no particular dominant directions of flow. The lack of dominant directions of flow make it advantageous to regard the individual traffic signals as isolated units. It does no good to set the offsets to work for any particular route, because there is nearly as much cross traffic as there is through traffic along the major routes. At the same time, the fact that there is significant congestion means that the traffic patterns *cannot* shift significantly from one evaluation period to the next. Thus, the assumptions intrinsic in the third generation control system are met. Namely, the traffic patterns from the last evaluation period *are* good indicators for the next traffic signal control period, and the traffic lights *can* be viewed as isolated control elements.

There is another case where we would expect the benefits of integrating predictive routing information to be marginal at best. This is the case when the traffic flows are highly *predictable*. For example, every weekday morning in New York City, the George Washington Bridge is totally congested in the Eastbound direction from approximately 7:30 AM till 9 AM. It is very predictable, and having additional predictive routing information could do little in the way of alleviating congestion on this roadway during the morning rush hour.

Even with the additional information available under IVHS, a new traffic signal control strategy will do no better than existing methods if that new strategy either cannot use the information or it uses that information incorrectly. It is *imperative* that the groups working on strategies for ATMS and ATIS communicate both the needs and the capabilities of their portions of IVHS. We assumed in this thesis that the predictive routing information was available in a form that was usable by PRISTINE. Without prior coordination, it is unlikely that the group implementing a traffic control strategy would be fortunate enough for this to be the case.

From our research we would conclude that, under highly predictable or congested conditions with no dominant directions of flow in the network, predictive routing information will *not* provide a significant advantage over existing traffic control strategies. On the other hand, traffic signal control methods integrating predictive routing information *will* reap *substantial* benefits under circumstances where large surges occur or dominant directional flows emerge.

- Our second conclusion is that if given the traffic data in a usable format, it will not require extensive improvements in computation efficiency to implement a traffic signal control strategy using predictive routing information in real time. PRISTINE is not a commercial package, but we were able to obtain signal settings for our sample network (i.e. a network with 28 nodes and 45 arcs) using a DEC Station 5000/20 in less than 30 seconds. This involved solving the non-linear program; when PRISTINE used the heuristic approach to set the splits, the entire process took only a couple of seconds.

The key to using the predictive routing information in traffic signal setting in real time will be in developing an appropriate data base. The data base must be accessible by the traffic signal control routines under ATMS, but additionally, the data base must be timely and accurate. This means the data base must be continually updated with information from both ATMS and ATIS. It will require a concerted effort between academia, industry and the government to develop such a system.

- Our third conclusion is that most reasonable measures of effectiveness (e.g. average delay per vehicle, average transit time per vehicle, and average queue length across the network) used to evaluate traffic signal control systems tend to offer the same conclusions. For example, the measures of average transit time per vehicle and average delay per vehicle are related. Certainly, one can also forward the argument that the average delay per vehicle is closely related to the average queue length in the network, another popular measure of traffic signal efficiency. Average stops per vehicle is not a good measure of effectiveness. Using average stops per vehicle as a primary measure of effectiveness puts a high premium on keeping *the majority of vehicles moving*, but it does not address the speed of that movement. Few motorists would trade single stop of 30 seconds and an average speed of 25 MPH for no stops with an average speed of 5 MPH if their paths through the network are more than a mile long. Average queue length, average delay per vehicle and the average transit time per vehicle would all indicate the strategy with the 25 MPH average speed as the better choice.

Our experience showed that, when a particular traffic strategy demonstrated savings in the area of delay per vehicle, the same traffic strategy generally produced the lowest transit times as well. The measures of effectiveness tend to be correlated.

- Our fourth conclusion is that the presence or absence of dominant direction flows play an important role in traffic signal control. In particular, predictive routing information was of much more benefit when there were dominant directions of flow within the network. The key to understanding the issue of dominant directional flows is realizing that aiding one route in a traffic scheme will generally harm another route. To be effective, the traffic signal control strategy must aid the majority of the motorists by a sufficient amount to *more than* compensate for the vehicles the strategy puts at a disadvantage. Under congested conditions, traffic flows are very volatile (i.e. even small disturbances in the traffic flow such as a stalled vehicle can cause large delays and stops for vehicles). The reason for this is straightforward. Drivers are not able to react instantaneously, and likewise, vehicles take time to decelerate and accelerate. If a single vehicle flashes its brake lights, the vehicle causes a series of braking actions throughout the traffic stream behind it.

In this thesis we explored only one aspect of new traffic signal control possibilities available under IVHS, the use of predictive routing information, but IVHS also offers the possibility for *giving vehicle operators directions* as well as

setting the traffic signals. Since we know that having dominant directions of flow greatly aid in reducing congestion, a IVHS strategy that directs vehicles should strive to guide vehicles in a way that emphasizes dominant directions of flow within the network. A system that imposed dominant directions of flow combined with a traffic signal control system integrating predictive routing information would provide *substantial* benefits to the motorist and society as a whole.

SECTION 9.4 Contributions

We have made three primary contributions in this thesis. First, we developed a *platform* which allows us to evaluate the benefits of predictive routing information. Second, we have explored new strategies for setting traffic signals which could be applied even without the additional information available under IVHS, and third, we have examined a particular case, the surge situation, where the use of predictive routing information offers its greatest benefits.

- We have developed a platform for evaluating the potential benefits of integrating predictive routing information with traffic signal control by comparing methods which use this information against methods which use existing information. The platform consists of three parts: a traffic signal setting system which uses predictive routing information (PRISTINE), an optimized third generation control system (3GC), and a simulation package (LS3). PRISTINE and 3GC have been discussed in detail earlier.

The LS3 simulation package offers many features, but its most important feature is that it allows a dynamic interface between the simulation and the traffic signal control strategy. The dynamic interface allows the simulation to fully integrate with both 3GC and PRISTINE. PRISTINE extracts the expected average flow rates from the same data base that LS3 uses to generate the vehicles entering the network. 3GC requires information on the vehicle arrival and departure times by arc while the simulation is operating. PRISTINE and 3GC are able to change the signal timing plan while 3GC remains in operation. Thus, we can accurately model dynamic traffic signal control.

- We have developed two new methodologies for setting traffic signals which could be implemented using existing traffic data. (Each of these was discussed in more detail earlier in this chapter.) First, the concept of using a spanning tree to define the arterial for the network could be applied without predictive routing information and offers an intuitive approach for synchronizing the traffic lights in an urban environment. Second, the Queue Effects Model uses the average flow rate on a street segment and first two moments of the size distribution for platoons approaching the intersection to determine the expected wait per vehicle and expected stops per vehicle. The maximal spanning tree heuristic uses average flow rate along the various street segments to determine the spanning tree. The first two moments of the platoon size and the average flow rate could all be gathered using induction loop sensors.

The methods used in PRISTINE explicitly consider platooning. Unlike other queuing systems used in traffic modeling, the Queue Effects Model explicitly allows for bulk arrivals at intersections. In practice, vehicles tend to travel in groups, platoons. Most queuing systems use the light traffic model approximation or consider congestion where vehicles come as single arrivals. These approaches ignore the congestion effects of the bulk arrivals. The spanning tree approach to setting the offsets is designed to improve flow for platoons along the major routes through the network. Most other methods for setting offsets focus on maximizing the flow rate on several major thoroughfares than coordinating traffic routing across the entire network.

- We have isolated and examined a condition under which predictive routing information offers substantial benefits in setting the traffic signals. We call this the surge condition. We define a *surge* as a sudden increase in traffic flow such that the *average* flow rate over the next traffic signal control period will exceed the capacity of one or more street segments under the *current* signal settings but *does not* exceed the *potential* capacity of the street segments, i.e. if the traffic signals were readjusted, the average traffic flow could be accommodated. By traffic signal control period, we mean the length of time that a particular signal timing plan will be in force. For example, in our simulation, we used a ten minute interval. Since we reset our timing plan every ten minutes, "the next period" would be the ten minute period when the next signal plan is in effect. We refer to the *average* flow rate, because it is entirely possible even under light traffic conditions for an intersection to have its capacity exceeded for one or more cycles even though the average flow rate does not exceed the street's capacity.

Predictive routing information is very useful for setting traffic signals during a surge situation for two reasons. The primary reason for the difference in performance is that current methods cannot react to changes in traffic flow until these changes occur. Existing traffic signal control methods assume that the last observation period accurately represents the upcoming, and in the surge situation this assumption is grossly violated. Second, most existing traffic signal control methods use a one street segment look ahead, it necessarily only can accommodate changes one street at a time. (Even systems that attempt to correlate usage's on street segments are inaccurate if the data changes suddenly.) If changes occur slowly enough, the traffic is able to pass through the streets relatively unimpeded using the old settings, but in the surge example, the new traffic patterns are clogged up in the early street segments because the surge *exceeds* the streets' capacities given the current signal timing plan. Thus, the streets actually act as storage for vehicles and the congestion moves sluggishly forward through the network.

A traffic signal control strategy using predictive routing information *knows* where the surge traffic wants to go and sets the signal timing plan to support this from the start. The traffic flow is accommodated throughout its routing, and the congestion levels never reach the extremes one sees if forced to rely on existing traffic data.

Section 9.5 Significance of Research to the IVHS Community

This thesis represents the culmination of over three years of research into the integration of predictive routing information with dynamic traffic signal control. The IVHS community should be both heartened and sobered by the results of that research.

There is a large potential for alleviating congestion, decreasing the emission of pollutants and increasing the economic viability of our roadways using predictive routing information under IVHS. At the same time, considerable time, effort and capital must be expended between now and the realization of that vision, and it will require an unprecedented level of cooperation between the government, academia and industry.

We were able to show under certain circumstances that an algorithm using the predictive routing information available under IVHS was able to save

substantial amounts in delay time over a third generation control strategy using traffic flow information which could be gathered using existing technology. The scenario where this savings was realized was a surge situation where there was an abrupt increase in the traffic flow such as one might see when a sporting event is finished or in the event of an evacuation. The traffic control model using the predictive routing information, PRISTINE, was able to react to the increase in traffic flow *as soon as the information becomes available to ATMS.* (If trip planning comes to fruition under IVHS, ATMS could be aware of a vehicle's projected routing before the vehicle leaves the garage.)

By increasing cycle times and the green splits allotted to the surge traffic, PRISTINE allowed the traffic to proceed smoothly through the network; the third generation control model was only able to *react* to the increase in traffic flow *after* it had already occurred, and by this time, the traffic had already reached heavy conditions. PRISTINE also performed well under conditions where there were clearly defined directions of flow through the network such as the morning rush hour. These were the positive findings in the thesis.

There were times when having the additional information available under IVHS did not help PRISTINE. When conditions were both congested and without clearly defined directions of flow, the third generation control system outperformed PRISTINE. Under these conditions, the best one can do in terms of stops, delay and transit time is to maximize the capacity of the traffic network and trying to push as much traffic out of the congested area as possible. Predictive routing information was not of substantial benefit in these cases.

It is important to note that PRISTINE did not take advantage of all the information or flexibility that IVHS potentially offers the traffic management community. In this thesis we took a fairly traditional view of traffic signal control. We set the traffic signals in terms of splits, offsets and cycle time. IVHS offers the traffic manager the opportunity to reach beyond the status quo and derive *entirely new* methods of looking at coordinating traffic and decreasing congestion.

For example, IVHS offers the capability to *direct* traffic as well as react to the drivers' wishes. If an algorithm were developed that *interacted* with ATIS and ATMS in such a way as to be able to redirect traffic away from congestion then further savings in delay and stops could be realized.

The thesis did not use all of the information that would be available under the most ambitious plans for IVHS. If a traffic control system tracked each vehicle as it progressed through the network and knew the vehicle's position, speed and route, the potential savings would even be greater. Researchers are currently looking at the use of IHVS information to dynamically determine origin-destination pairs; this could be of potential benefit in the area of vehicle routing. Additionally, research has been done in the area of ramp metering, variable speed control, etc. for major thoroughfares using IVHS technology. If this research were integrated with research into traffic signal control of urban grids, the benefits could out weigh the benefits of considering these approaches

individually. All these concepts will require considerable effort to bring to fruition.

The traffic signal control strategies must be able to use the data supplied by ATMS and ATIS, and traffic managers developing algorithms for use under ATMS must coordinate closely with the agencies responsible for ATIS and AVCS to see what information can realistically be supplied and in what form that data can be provided. The thesis assumed that the predictive routing information would be supplied in a format that was usable by PRISTINE. Currently, this information does not exist at all. For a traffic control system to work in real time, the information hidden within the real world traffic data must be extracted before it is fed to the traffic control systems. Now is the time to think about what data a traffic signal control program could use and in what form. Conversely, the IVHS community should be realistic about the type of data it requires. Some data is more costly than others in terms of technology and infrastructure. The IVHS community should coordinate early on and aggressively determine a course which will take it to the economic break even point.

The anticipated gains from IVHS are large, but there is a long way to go from where we are now before these gains become realities. The thesis demonstrated that the payoff for even a small part of IVHS, such as the availability of predictive routing information, could be substantial under certain circumstances. A broader application of the other features of IVHS would lead to further benefits, we suspect.

The gains promised under IVHS will only be realized if the IVHS

community is able to coordinate and manage the data and computational necessities required by ATMS.

Section 9.6 Opportunities for Further Research

No study of an interesting topic area is ever complete, and this is certainly true of the study of integrating predictive routing information with dynamic traffic signal control. The potential applications of IVHS in the area of traffic signal timing are nearly limitless. One potential area of research would be to consider an iterative approach that both integrated predictive routing information, perhaps using PRISTINE as a base, *and* allowed for changing drivers' routings via the on-board guidance and information systems available under IVHS. This would allow the system to guide the vehicles into dominant directions of flow and greatly reduce congestion effects.

Another topic to consider would be the use of a limited combinatorial approach for setting the traffic signal control, such as we used in our optimized third generation control system, but allow the integration of predictive routing information. The system should do well under light traffic conditions as well as compensate for surges. The third generation control system would have to be modified to take into account flows rather than individual vehicles.

It would be interesting to see a field test of PRISTINE in a metropolitan area. PRISTINE would have to be expanded to consider: multiple lanes, left turn lanes, etc.

In its final form IVHS would allow two way communication between the traffic control center and individual vehicles. A traffic signal control strategy that used not only every vehicle's routing but also its position and speed in real time would be able overcome some of the obstacles that PRISTINE experienced in congested, disorganized traffic. This would be a fruitful area of research as well.

Expanding the LS3 traffic simulation to include any subset of the following: multiple lanes, multiple types of drivers, multiple types of vehicles, left turn lanes/logic, etc. would be useful. This would allow the user to more accurately model traffic flows and behaviors.

We know that predictive routing is useful under conditions where there are unexpected surges in traffic flow. It would be interesting to examine situations where one or more traffic links are lost (e.g. a building fire, etc.) and see how useful predictive routing information is under these circumstances.

We hope that investigators will find these ideas useful for structuring further research.

Appendix A

Mathematical Description of the Optimized Third Generation Control System (3GC)

Copies of the 3GC traffic generation control system are available from the author on request.

SECTION A.1 Assumptions, Definitions and Inputs

In this appendix we will use the notation and assumptions developed in chapter three *unless explicitly* stated otherwise. Interested readers are encouraged to review sections 3.1, 3.2 and 3.3 before continuing.

SECTION A.1.1 Assumptions

Under 3GC we will relax the following two assumptions. We will let the cycle time vary by intersection; there is no overall cycle time for the network under 3GC. This counters assumption 3.2(1). We do not assume that 3GC knows the desired usage rates of the arcs in the network. This counters assumption 3.2(6).

We will make the following additional assumptions.

(1) The traffic encountered at each intersection during the last traffic signal control period *accurately represents* the traffic which will be encountered during the upcoming traffic signal control period. This is also referred to as the *steady state assumption*. During the simulation runs in chapter seven, the traffic signal control period was set uniformly to ten minutes, but the period could be allowed to vary. The assumptions in the thesis are *not* dependent on a fixed traffic signal control period.

(2) Time is discrete. Traffic events such as turning movements, acceleration and deceleration, etc. take place at discrete moments in time. For the simulation, we divided all time periods into discrete increments of 1.007 seconds. This increment could be set to a different value.

(3) We are able to record the arrivals at, departures from and turning movements of each vehicle in the network on each arc it traverses throughout **G(N,A)**. This is *substantially more* information than existing third generation control systems have access to.

(4) The waits and stops occurring at each intersection are convex functions with respect to the splits and offsets for these intersections.

SECTION A.1.2 Definitions

All definitions from chapter three remain in effect. In addition, we make the following supplemental definitions.

t ≡ The time as measured from the beginning of the last implemented TSC plan; in the simulation in chapter seven **t** varied from 0 to 599.

$\mathbf{s_j}$ ≡ The *set* of green splits for incoming arcs to node "j"

Green(Θ, (i->j), t, s_j) = {0 if the traffic flowing along (i->j) would find the light green at time "t" given Θ and $\mathbf{s_j}$, 1 otherwise}.

Grid((i->j)) = {1 if $\exists$ **(i->j)** $\in$ **A**, 0 else...}

Next(Θ, (i->j), t, s_j) = {0 if **Green(Θ, (i->j), t, s_j)** equals 0, the time from "t" until the next green light for traffic traveling along arc (i->j) into node "j" otherwise}.

Arrival((i->j), (j->k), t) = {1 if a vehicle passed through node "i" onto arc (i->j) at time "t" during the last TSC period, 0 else...}

Θ_j ≡ The set of offsets for node "j". Note that setting the offset for *any arbitrary* direction at a node *automatically* sets the remaining offsets, because directions sharing the same green split at node "j" have the same offset. Call the direction for which the offset is selected (i->j). If arc (k->j) does share a green split with arc (i->j) then it has an offset equal to $\Theta_{ij}+s_{ij}$. In 3GC, we selected the first offset occurring lexigraphically in G(N,A) to set.

SECTION A.1.3 Inputs

We used the following inputs to 3GC:

(1) All arrivals at, departures from and turning movements of each vehicle in the network on each arc it traverses throughout **G(N,A)** [from traffic data].

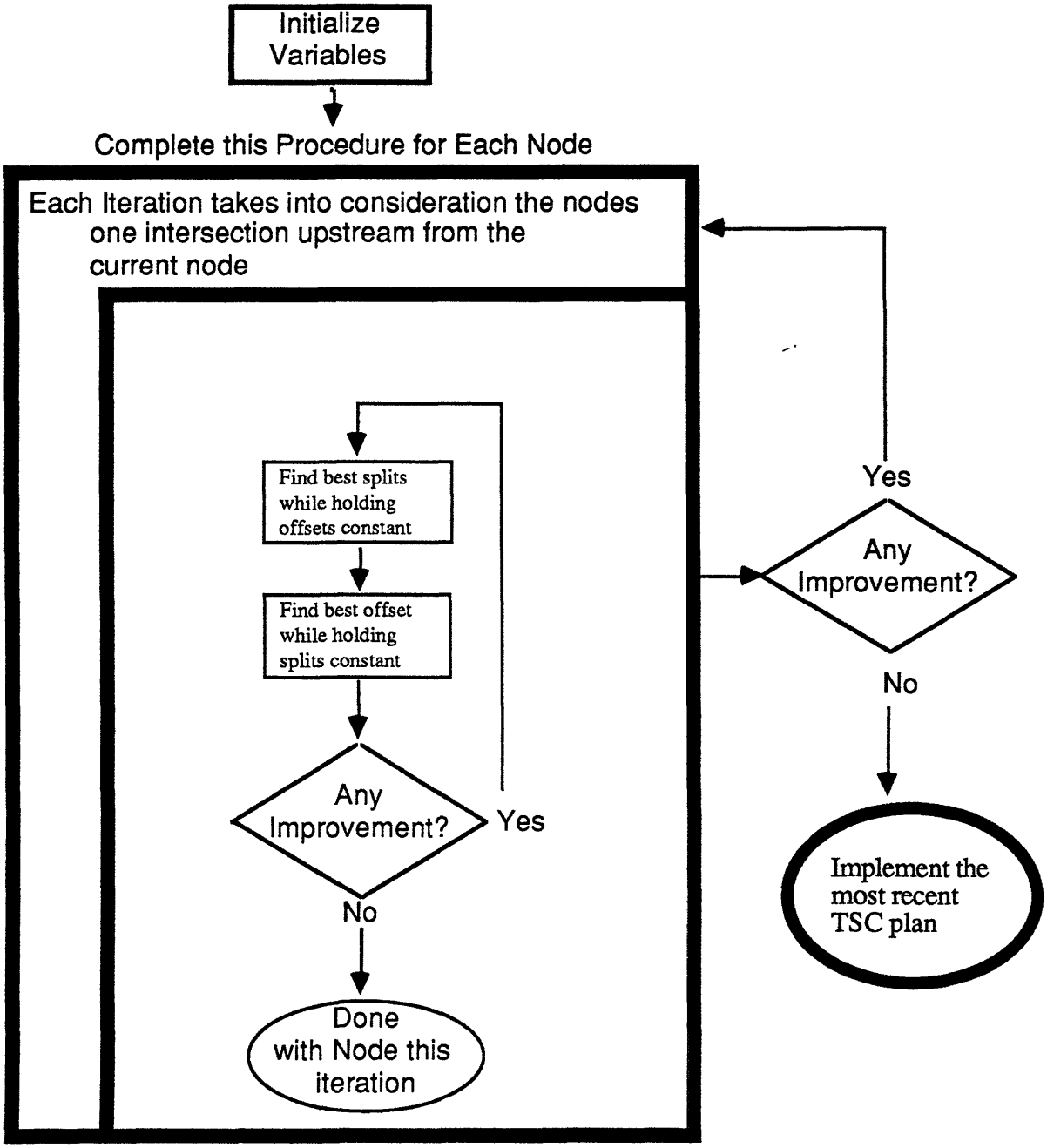
(2) The maximum cycle time for the network, $\mathbf{Cycle_{max}}$ [from the traffic manager].

(3) The minimum acceptable green split, $\mathbf{Split_{min}}$ [from the traffic manager].

(4) The weightings given to stops, β, and to waits, γ, in the objective function [from the traffic manager].

SECTION A.2 Mathematical Statement of 3GC

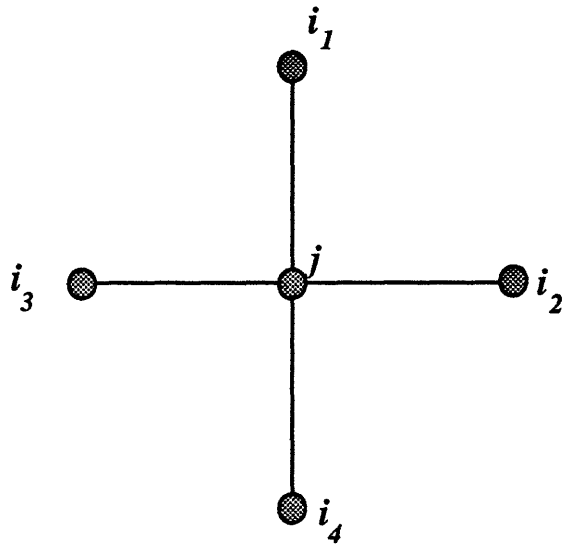
The 3GC method uses the following logical flow to obtain a traffic signal control plan.



Flow chart for the 3GC method

Figure A.1

To more fully appreciate the method, we will examine how 3GC would set the offset and split for a single node on one pass through the system. Assume we have the intersection shown in the diagram below.



Node and incoming arcs used to illustrate 3GC method

Figure A.2

The objective function for 3GC is a weighted sum of delay and stops aggregated over the entire network for a given Traffic Signal Control (TSC) plan. The stops for the intersection shown above would be calculated as follows:

$$\mathbf{Stops}_j \triangleq \sum_{t=0}^{599} \sum_{k=1}^{4} \sum_{p=1,p\neq k}^{4} \mathbf{Arrival}(\,(i_k->j),(j->i_p),t-\hat{t}_{i_k j})\,*\,\{\mathbf{Green}(\Theta,(i_k->j),t,s_j)+ \mathbf{Green}(\Theta,(j->i_p),t+\hat{t}_{ji_p}+\mathbf{Next}(\Theta,(i_k->j),t,s_j),s_j)\}$$

Using a similar approach we can calculate the wait:

$$\mathbf{Wait}_j \triangleq \sum_{t=0}^{599} \sum_{k=1}^{4} \sum_{p=1,p\neq k}^{4} \mathbf{Arrival}(\,(i_k->j),(j->i_p),t-\hat{t}_{i_k j})\,*\,\{\mathbf{Next}(\Theta,(i_k->j),t,s_j)+ \mathbf{Next}(\Theta,(j->i_p),t+\hat{t}_{ji_p}+\mathbf{Next}(\Theta,(i_k->j),t,s_j),s_j)\}$$

We see that the waits and stops are a function of the arrivals, splits and offsets. The arrivals remain fixed *throughout* the signal setting process. Thus, our problem can be rewritten as:

$$\left(\Theta_j^*, s_j^*\right) = \underset{\Theta_j, s_j}{\arg\min} \{\beta * \mathbf{Stops}_j + \gamma * \mathbf{Wait}_j\}. \qquad [A.1]$$

Initially, we set our current objective function to a very large number. We select the optimal splits and offsets using a combinatorial approach. We allow the sum of the non-overlapping green splits, i.e. the cycle, for the intersection to vary from $\mathbf{2*Split_{min}}$ to the $\mathbf{Cycle_{max}}$. The offset is allowed to vary from **0** to $\mathbf{Cycle_{max}}$. We allow the problem to reach the stated optimal condition which is frequently quite time consuming. (In practice, most third generation control systems only complete three to five passes before selecting a traffic signal timing plan.) The process described above is completed for each node. The set of generated splits and offsets is the new traffic signal control plan for the network.

An example of the 3GC method is included in section 7.3.2 above.

SECTION A.3 Summary of Significant Differences between 3GC and existing Third Generation Control Systems

Every third generation control system is unique. All of them have their own methods for setting the traffic signals varying from fixed equations to heuristic and dynamic programming techniques. Likewise, these third generation control packages all have their own measures of effectiveness.

3GC is *unique* in two primary ways. First, in real world applications, decisions about the traffic signal control settings must be made within a reasonable time frame, and 3GC is not constrained in such a way. We allow 3GC to continue processing until it finds its optimal solution. In real world applications, traffic signal control systems are generally allowed to complete only three to five passes. Second, 3GC has access to *far* more information than any real world system. We assume that 3GC knows all arrivals at, departures from and turning movements of each vehicle in the network on each arc it traverses throughout **G(N,A).** Existing third generation control systems have to get by with much less data. It would require a minimum of four induction loop sensors per street segment for 3GC to gather the type of data it uses to make its calculations. Most real world systems are lucky to have access to one operational sensor per street segment.

Appendix B

Brief Technical Overview of the Lin-Sarkar-Staats (LS3) Traffic Simulator

Copies of the LS3 traffic simulator are available from the author upon request.

SECTION B.1 Features

The LS3 simulator has the following features.

(1) It allows vehicles to enter and depart from any set of nodes in the network. Additionally, these nodes are *not* required to be traffic signal control nodes. Consequently, vehicles can be made to enter or leave the simulation in mid-arc.

(2) The simulator allows roadway segments of lengths varying from 16 feet to over 120 miles. The speed limits on these segments can vary from 5 MPH to 65 MPH in increments of 5 MPH.

(3) The LS3 simulator is designed to exploit platooning behavior in traffic and moves platoons as units.

(4) Individual vehicles retain their identity and routings throughout the network even when part of a platoon.

(5) Vehicles accelerate and decelerate according to realistic parameters (see technical aspects below).

(6) Vehicles enter the networks in platoons. The distribution of platoon size can be varied by entry node.

(7) The simulator monitors stops, delay and transit time per vehicle and supplies these measures of effectiveness as part of its standard output package. Stops represent a count of the times a vehicle's speed reached 0 MPH. Delay represents the total amount of time in the simulation when a vehicle's speed was strictly less than 5 MPH. Transit time represents the amount of time it takes a vehicle to go from its point of origin to its destination in the network.

(8) It is capable of varying its traffic signal control plan in mid-simulation run without loss of data, i.e. vehicles continue on within the simulation from their current locations while responding to the new traffic signal control plan.

(9) The LS3 simulator is capable of accepting new origin-destination pairs and vehicle generation rates during simulation runs.

(10) The simulator models congestion effects such as stop and go traffic, multiple stops per street segment due to crowding and acceleration/deceleration requirements. The LS3 simulator recognizes phenomenon such as link blockages due to spillback.

(11) The simulator is written in ANSI FORTRAN 77 with the exception of the random variable calls. With minor modifications, the LS3 simulator could be exported to any platform supporting this language.

SECTION B.2 Technical Aspects

The LS3 program consists of over 400 kilobytes of code. No attempt is made in this section to exhaustively describe the operation of the LS3 simulator. The purpose of this section is to give a *brief* overview of some of the simulation's primary operating characteristics and parameters. This section is divided into four sub-sections.

SECTION B.2.1 System Characteristics

This section contains information on technical aspects of the simulation which are germane to all succeeding sections. The simulation operates on a discrete time system. The simulation clock is updated in 1.007 second increments. This is long enough for vehicles to make unobstructed turning movements, decelerate from 5 MPH to a stop or accelerate from a stop to 5 MPH. A vehicle moving at 5 MPH for 1.007 seconds will cover a distance of approximately 7.4 feet which is almost exactly half of a standard car length.

The roadways are discretized into *slots* of 7.4 feet in length. An automobile fills two such slots. The vehicle maintains its *exact* position and speed (see data structures below), but in the representation of the roadway, the vehicle is placed in the slot which most closely matches its current position. For example, a vehicle which was 742 feet into a street segment would be placed in slot 10 for that segment. (Conflicts are resolved working from the head to the tail of the street segment.)

Roadways or street segments are represented as one-way arcs. Two-way streets are represented by two distinct arcs, facing opposite directions. The simulator currently only handles single lane traffic.

Vehicles are always in one of four states in the simulation: (1) accelerating/cruising, (2) decelerating/braking, (3) stopped or (4) following. Vehicles in the simulation accelerate at a constant rate of 4 feet/sec^2, and they decelerate at a rate of 10 feet/sec^2. Vehicles will not accelerate past the speed limit for a street segment.

The vehicles follow uniformly cautious driving policies throughout the simulation. Vehicles already on a street segment have priority in cases of conflict. Specifically, vehicles making driving maneuvers such as crossing an intersection, making a left turn, etc. would wait for the existing or conflicting traffic to clear before completing this maneuver. As noted in section 7.1 under heavily congested traffic conditions, these policies can lead to significant delays for individual vehicles.

When vehicles enter the network, they are randomly assigned a routing. Vehicles will not deviate from this routing. Vehicles enter the network in platoons. (A platoon can consist of a single vehicle.) For the LS3 simulator, a platoon is any group of vehicles that effectively moves as a unit. A simple test for determining whether a subject vehicle is part of a platoon is if the lead vehicle in the platoon brakes and the subject vehicle would brake as well then the subject vehicle is part of the platoon. (Remember, all vehicles *in the simulation* follow the *same driving policies;* so, all vehicles brake under the same conditions.) Platoons disperse when they are broken up by changes in traffic signals or as vehicles turn off the primary direction of travel to follow their own routings. Platoons are *generated* with exponentially distributed headways between the platoons. In the

simulator's current configuration platoon sizes are distributed according to a geometric distribution which varies with the vehicle generation rate at each node.

SECTION B.2.2 Assumptions

The simulation is based on the following assumptions.

(1) All vehicles are identical in size and performance.

(2) All drivers exhibit the same performance.

(3) Time is discretized, and all driving maneuvers take place at fixed instances in time.

(4) All traffic signals have two colors, red and green.

(5) Vehicles enter the system in platoons of varying sizes with exponentially distributed headways.

(6) All street segments are single lane.

SECTION B.2.3 Data Structures

The LS3 simulator has literally hundreds of thousands of individual strings, variables, files and array elements associated with it. Some of the representative data structures are discussed in this section. (Also see section 7.1.3 for a description of the data structures.)

Structure AUTO holds information that would be known by a vehicle traversing the network. One major element of AUTO is the data structure CAR. CAR contains the following information:

CAR(i,j): * Refers to vehicle "i"

* Refers to the following aspect of the vehicle depending on the value of "j"

--j=1: Start node of current street segment which the car is on

--j=2: End node of current street segment which the car is on

--j=3: Absolute position of the vehicle on the arc

--j=4: Current speed of the vehicle(the arc specifies the direction)

--j=5: Status(i.e. accelerating, decelerating, stopped, following)

--j=6: Vehicle this car is following (if any)

--j=7: Vehicle which is following this car (if any)

The last two portions of CAR help move the platoons. There are additional data structures in AUTO which hold routing information.

Structure PRI has information on the expected usage rates for the routes through the network. PRI has both the frequency of vehicle generation by node as well as the distribution of route choices by node.

Structure NET contains information about the physical characteristics of the network. GRAPH is a listing of all connections in the network. SPEED and DISTANCE hold information on the maximum speeds and arc length by street segment respectively. ORDER contains information about intersection by outgoing and incoming arcs; this allows LS3 to determine whether a vehicle is continuing straight through an intersection or making a left or right hand turn.

Structure ROAD holds information as seen from the perspective of the roadway. For example, ROAD has a listing of which vehicle (if any) each slot on a street segment is currently holding. When a vehicle is making a left hand turn or moving onto a new street segment, the vehicle checks its safety clearances by examining the appropriate slots in ROAD and then verifying the on-coming vehicles' speeds and distances to impact.

SECTION B.2.4 Input Requirements

The LS3 simulator requires several inputs to operate. They are as follows.

(1) The simulator requires the network structure. This is contained in the file NET which is read in once during the initialization phase of the simulation.

(2) LS3 needs to know the duration of the simulation and the seed for the random number generator. These quantities are entered by the simulation operator during the initialization phase.

(3) The simulator requires the node generation rates, routings and route selection rates by node. This information is read from a sequential *book* of files called PATH and RATE. These parameters can vary throughout the simulation.

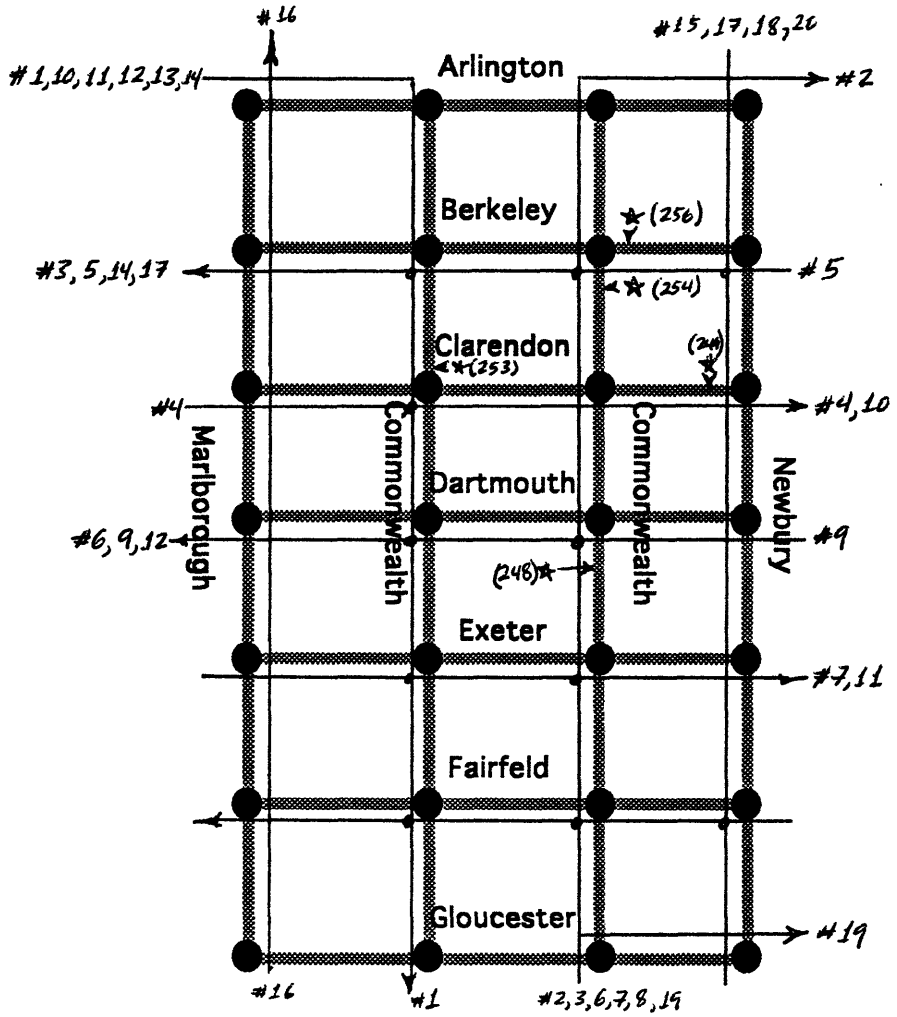
(4) The LS3 simulator requires the traffic signal control plan to operate. The plan is provided by called subroutines. The subroutines are called approximately every ten minutes, simulation time (i.e. 600 iterations), under the current configuration for LS3.

Appendix C

Sample Predictive Routing Information

The thesis used hundreds of sets of predictive routing information in the simulation. For example, the mixed traffic scenario alone had 70 sets each with multiple routes. In this section, we will give an example of how the predictive routing information was generated for one of the sets in the morning rush hour scenario. This will accomplish two purposes. One, it will give the reader an appreciation of the method we used to translate the sensor data from the city of Boston into predictive routing information. Second, it will give the reader a better feel for the form of the predictive routing information used by the thesis.

We used the routings shown in figure C.1 for the morning rush hour scenario. We obtained a printout that had the average vehicle flow rates broken out in fifteen minute intervals. Since we used ten minute intervals, we had to interpolate the time segments that did not fall evenly on the half hour breaks. For simplicity, we will use the 8 AM data. Before progressing any further, it is important to note that the routing data we generated is not uniquely determined by the sensor data.



The routings used in the morning rush hour scenario are displayed in the figure above. The numbers in parenthesis represent the numbers assigned to the nodes. The stars represent sensor locations

Figure C.1

The sensor data is displayed below in table C.1.

Sensor	Vehicle Flow Rate (Vehicles/Minute)
248	3
249	6.4
253	7.1
254	1.0
256	5.6

The traffic flow data from Boston Traffic Center for April 5, 1994 at 8 AM.

Table C.1

Now, let the flow rate on route "i" be represented by **r(i)** and the data from sensor "j" be represented by **s(j)**. The sensor data generates the following equations:

s(248)=r(2)+r(3)+r(6)+r(8)
s(249)=r(4)+r(10)
s(253)=r(1)+r(10)+r(11)+r(12)+r(13)
s(254)=r(2)+r(3)
s(256)=r(17).

We used symmetry (in this case) to obtain the following relationships:

r(5)=r(9)=r(6)
r(7)=r(19)=r(11)
r(14)=r(13)=r(12)
r(13)=r(20)
r(16)=r(2)+r(6)+r(8)+r(19).

This allowed us to generate the following predictive routing information:

Route	Aggregate Average Flow Rate (Veh/Min)
1	1
2	.5
3	.5
4	3.2
5	1
6	1
7	1
8	1
9	1
10	3.2
11	1
12	1
13	1
14	1
15	1
16	3.5
17	5.6
18	1
19	1
20	1

The figures above represent the average vehicle generation rate in vehicles per minute by route. To be usable by the simulation, this data must be aggregated by node.

Table C.2

This information is then combined to form the generation data for the nodes.

Node	Average Platoon Arrival Rate (Platoons/Iteration)	Platoon Size Parameter
1	.017	.998
5	.028	.523
7	.035	.263
15	.031	.380
25	.017	.998
28	.036	.252

The average platoon arrival rate is the parameter for a negative exponential random variable which generates the inter arrival times for the platoons. The platoon size parameter is the single parameter for a geometric probability mass function which generates the platoon size.

Table C.3

The simulation generated the platoons with exponentially distributed headways. The average platoon arrival rate represents the single parameter for the negative exponential distribution sampled to determine platoon inter arrival times; note, this parameter varies by node. The platoon size parameter is the single parameter for the geometric distribution used to generate the platoon size.

The parameters in table C.3 are used to determine the platoon size when a platoon enters the simulation, but the actual platoon sizes on the roadway during the simulation can be quite different. For example, during our initial testing of the LS3 using the Backbay, we did a simulation run for the mixed traffic model where we sampled each arc every one hundred iterations. We discovered the platoon length in the street segments in the simulation varied from a single vehicle to platoons of up to 22 vehicles.

In the simulation, a maximum of one vehicle can enter the network from each node during each iteration period, 1.007 seconds. The additional vehicles are held out of the network and released on each subsequent iteration when the roadway has cleared sufficiently to allow them to enter, i.e. if spillbacks have caused an entire road segment to become filled then new vehicles could be generated, but they would not be able to enter the street segment until there was room to accept them.

As a vehicle is generated it is randomly assigned to a route originating at its generating node. The probability of being assigned to a specific route is equal to the average fraction of total flow out of the node that route makes up. For example, if a platoon is generated at node 25, every vehicle in that platoon has 100% chance of being assigned to route 9. On the other hand, a vehicle generated at node 7 has only a 12.1% chance of being assigned to route 12.

G L O S S A R Y

a Number of columns in a rectangular idealized urban grid

A Arc set for G(N,A)

$\mathbf{After}_i$ Average number of customer type "i" a random vehicle arriving after it enters the system which must be served before it can cross the intersection

A_r Arc list for arteries

APTS Advanced Public Transportation Systems

$\mathbf{Arterial}_{\mathbf{min}}$ Minimum green split for a priority direction

ATIS Advance Traveler Information System

ATMS Advanced Traffic Management System

AVCS Advanced Vehicle Control System

b Number of rows in a rectangular idealized grid

b(h) Probability of having "h" vehicles in a platoon

b(l) PMF for bulk arrival sizes in the uncoordinated direction

$\mathbf{Before}_i$ Average number of customer type "i" a random vehicle sees when it enters the system which must be served before it can cross the intersection

C Cycle length (time) for G(N,A)

C_i "Branch Count Index" for S

CVO Commercial Vehicle Operations

$\mathbf{Cycle}_{\mathbf{max}}$ Maximum allowable cycle length for the network

$\delta_i(l)$ Indicator variable which equals 1 if sample path "i" was stopped at node "l" due to traffic signal control

Δ_i Number of delays from traffic signals for sample path "i"

D Matrix of distances between nodes in G(N,A)

D(j,h) Boolean function equal to one if node j is the destination for route h

$f_w^T(s)$ S-transform for PDF of waiting time

FHWA Federal Highway Administration

G_A Amount of green time in the coordinated direction

G_B Amount of green time in the uncoordinated direction

G(N,A) Network representation of Urban Grid, consisting of several sub-sets.

H Minimum platoon size to be accommodated for an arc belonging to **S** in terms of green light time

i^* Root node for **S**

$(i \rightarrow j)$ One-way arc going from node "i" to node "j"

I_i The index function which gives the weighting for the particular route flow rate in the PF Heuristic

IVHS Intelligent Vehicle Highway System

λ_1 Actual arrival rate in the uncoordinated direction, taking into account bulk arrival sizes

λ_{ij} Desired average usage rate for $(i \rightarrow j)$ for a fixed time-frame

$\hat{\lambda}_{ij}$ Estimate for λ_{ij}

Λ Vector of desired usage rates for each route in **R**

L Minimum platoon size (in vehicles) which the traffic signal control plan should accommodate crossing for in the coordinated direction

m Number of arcs in G(N,A)

MCC Master Clock Concept

v Time average number of vehicles a randomly arriving vehicle sees in front of it in its own group

n Number of nodes in G(N,A)

$n_b(t_1, t_2)$ Number of arrivals between times t_1 and t_2 in the base case (so surge)

n_i Number of nodes in sample path "i"

n_R	Number of routes in R with non-zero desired usage
$\mathbf{n_s(t_1, t_2)}$	Number of vehicles whose transit time is recorded by the simulation between times t_1 and t_2 for the surge case
$\mathbf{n_s'(t_1, t_2)}$	Estimated number of vehicles whose transit time is recorded by the simulation plus the vehicles in parking lots between times t_1 and t_2 for the surge case
N	Node set for G(N,A)
N_S	Number of sample paths evaluated
$\overline{\mathbf{N}}_\mathbf{S}$	Time average number of vehicles in system
Network$_{min}$	Minimum green split across the network
NLP	Non-Linear Program
$\vec{\varphi}$	Index of elements of Λ in *descending* sequence
O(j,h)	Boolean function equal to one if node j is the origin for route h
$\mathbf{p(x_0)}$	Probability distribution for number of stops due to TSC in the network
$\mathbf{p_x^T(z)}$	Z-Transform of PMF for number of stops due to TSC in the network
PH	Pure Heuristic
PRI	Predictive Routing Information
PRISTINE	Predictive Routing Information Signal Timing INtEgration model
Ψ	Alignment function which indicates if arcs share common green cycle
$q_{ij}(t)$	Instantaneous flow rate (vehicles/time) along $(\mathbf{i} \rightarrow \mathbf{j})$
QEM	Queue Effects Model
ρ_1	Time average fraction of time the server is occupied during
ρ_2	Time average fraction of time the light is red in the uncoordinated direction
ρ_{ij}	Maximum flow rate along $(\mathbf{i} \rightarrow \mathbf{j})$

R Set of all possible routes through G(N,A) *[Note: **not** to be confused with $\Re$ which is the set of real numbers]*

ROGUS ROute GUidance Simulation

S Spanning tree for G(N,A)

$S_{G(N,A)}$ Average stops per vehicle in G(N,A)

$\vec{S}_{\theta}(j)$ Fraction of sample paths passing though node "j" which had to stop due to signal control

Θ Vector of off-sets for G(N,A); must be used in conjunction with an orientation at each intersection (derived from S and i^*) to *fully specify* the signal timing.

$\tau_b(t_1, t_2)$ Average transit time for departures between times t_1 and t_2 for the base (no surge) case

$\hat{t}_{ij}$ Travel time along $(i \rightarrow j)$

$T_i(j)$ Time vehicle "i" arrived at $j^{\underline{th}}$ node in its sample path

$T_b(t_1, t_2)$ Total transit time for departures between times t_1 and t_2 for the base (no surge) case

$T_s'(t_1, t_2)$ Total estimated transit time for all vehicles between times t_1 and t_2 for the surge case

TRAF-NETSIM Traffic Network Simulation

TSC Traffic Signal Control

UTSM Urban Traffic Simulation Model

V Matrix of nominal velocities between nodes in G(N,A)

w Amount of time waiting due to traffic signal control in the network; note, this does **not** include congestion time, etc.

$\overline{W}_0$ Average amount of time a randomly arriving customer will have to wait for the customer in service

$\overline{W}_S$ Average amount of time a randomly arriving customer must wait before passing through an intersection

x	Number of stops in the network due to traffic signal control
$\bar{x}_1$	Average amount of time it takes a vehicle to cross the intersection
X_i	Sample path for vehicle "i"

BIBLIOGRAPHY

Adams, W. 1989. Road Traffic Considered as a Random Series. *Journal,* Institute of Civil Engineers (4)121-130.

Baass, K. and Lefebvre, S. 1990. Analysis of Platoon Dispersion with respect to Traffic Volume. *Transportation Research Record*(1194)64-74.

Ballman, K. 1991. *Cost-Effectiveness of Smart Traffic Signals.* Ph.D. Thesis, Sloan School of Management, MIT.

Barlas, Y. 1989. Multiple Tests for validation of Systems Dynamics Type of Simulation Models. *European Journal of Operation Research*(42)59-87.

Berenson, M. and Levine, D. 1992. *Basic Business Statistics.* Prentice Hall, Englewood Cliffs, New Jersey, 569-572.

Bertsekas, D. 1992. *Course Notes for Non-Linear Programming.* Electrical Engineering/Computer Science Department, MIT.

Bretherton, R., Bowen, G., et.al. 1986. The Use of SCOOT for Traffic Management. *Road Traffic Control, IEE Conference Publication* (260)81-84.

Busch, F. 1993. *MOTION - Ein neues Verfahren fur die Stadtische Lichtsignalsteuerung und seine Erprobung im Rahmen des EG - Programms ATT.* HEUREKA - Congress, Karlsruhe, Germany. March 1993.

Cayley, A. 1857. On the Theory of the Analytic Forms called Trees. *Philosophy Magazine* (4)172-176.

Cedar, A. 1989. Signalized Intersections with Variable Flow Rates II: A Model for Delay Estimation. *Journal of Advanced Transportation* (23)53-66.

Chang, G. and Kanaan, A. 1990. Variability Assessment for TRAF-NETSIM. *Journal for Transportation Engineering*(116)636-657.

Chaudhary, N. and Pinnoi, A. 1993. Mixed Integer Linear Programs for Optimizing Signal Settings in Traffic Networks, Part I: Characteristics and Computational Efficiency. *Applications of Management Science, Volume 9: Network Optimization Applications [Draft].* Texas Transportation Institute.

Cohen, S. and Little, J. 1982. The MAXBAND Program for Arterial Signal Timing Plans. *Public Roads* (46)2, 61-65.

Edie, L. and Bavarez, E. 1967. Generation and Propagation of Start-Stop Traffic Waves. *Vehicular Traffic Science.* New York, 26-37.

Euler, L. 1736. Solutio Problematis ad Geometriam Situs Perinents. *Commentarii Academiae Scientarium Imperials Petropolitanae* 8(1736), 128-140.

Garber, N. and Hoel, L. 1988. *Traffic and Highway Engineering.* West Publishing Company, New York, 219-231.

Gartner, N. 1981. Prescription for Responsive Urban Traffic Control. *Control of Urban Traffic Systems,* United Engineering Trustees, Inc., 31-42.

Gartner, N. 1983. OPAC: A Demand-Responsive Strategy for Traffic Signal Control. *Transportation Research Records*(906)75-81.

Gartner, N., et. al. 1990. MULTIBAND: A Variable-Bandwidth Arterial Progression Scheme. *Transportation Research Record.*

Gartner, N. 1992. Interview by author, AUG 1992. University of Lowell, Lowell, Massachusetts.

Gazis, D. and Herman, R. 1992. The moving and "Phantom" Bottlenecks. *Transportation Science* (26)223-229.

Gerlough, D. 1955. Use of Poisson Distribution in Highway Traffic. The Eno Foundation for Highway Traffic Control, Saugatuck.

Hadi, M. and Wallace, C. 1991. Improved Optimization Efficiency in TRANSYT-7F. Transportation Research Center, University of Florida, Gainesville, Florida.

Harris, S., Rabone, A., et.al. 1992. ROGUS: A Simulation of Dynamic Route Guidance Systems. *Traffic Engineering and Control*(33)327-329.

Haver, E. 1994. Can One Estimate the Value of Life or is it better to be Dead than Stuck in Traffic? *Transportation Research,* (28A)109-118.

Hawat, G. 1992. Interview by author, SEP 9, 1992, Boston City Hall, Boston, Massachusetts.

Heidelberger, P. and Towsley, D. 1989. Sensitivity Analysis from Sample Paths using Likelihoods. *Management Science*(35)1475-1488.

Herman, R., Jayakrishnan, R., et.al. 1990. Microscopic Simulation of Traffic in Networks: Supercomputer Experience. *Journal of Computing in Civil Engineering*(4)1-19.

Herman, R., Mahmassani, H., et.al. 1990. Investigation of Network-Level Traffic Flow Relationships: Some Simulation Results. *Transportation Research Record*(971)121-129.

Herman, R. and Rothery, R. 1967. Propagation of Disturbances in Vehicular Platoons. *Vehicular Traffic Science*. New York, 14-25.

Hobbs, F. 1979. *Traffic Planning and Engineering, 2d Ed.* Permagon Press, New York.

Holroyd, E. and Miller, J. 1966. Route Crossings in Urban Areas. *proceedings Australian Road Research*(1)394-419.

Homburger, W. and Kell, J. 1988. *Fundamentals of Traffic Engineering, 12th Ed.* University of California at Berkeley, Institute of Transportation Studies: Course Notes.

Hounsel, N., McLeod, F., et.al. 1990. SCOOT: A Traffic Database. *Road Traffic Control, IEE Conference Publication* (320)99-103.

Huijun, H. and Fu, J. 1992. UTSM Simulation Model and Its Application to Evaluation of a Scheme of Traffic Systems. *Journal of Shanghai Jiaotong University*, China.

Hutson, V. and ReVelle,C. Maximum Direct Covering Tree Problems. *Transportation Science*(23)288-299.

IVHS America. 1992. *Strategic Plan for "Intelligent Vehicle-Highway Systems in the United States"*. Washington, D.C.

Jayakrishnan, R., Tsai, W., et.al. 1993. Algorithms for Efficient Real-Time Traffic Assignment. *Proceedings of the 5th International Conference on Computing in Civil and Building Engineering*, ASCE, New York, 599-606.

Kaufman, D. and Smith, R. 1993. Fastest Paths in Time-Dependent Networks for Intelligent Vehicle-Highways Systems Application. *IVHS Journal*(1)1-11.

Kim, C. and Hobeika, A. 1993. Short-term Demand Forecasting Model from Real-Time Traffic Data. *Proceedings of the Infrastructure Planning and Management Conference*, ASCE, NewYork.

Kinzer, J. 1933. *Application of the Theory of Probability to Problems of Highway Traffic*. B.C.E. Thesis, Polytechnic Institute of Brooklyn.

Kleinrock, L. 1975. *Queueng Systems, Volume II: Computer Applications.* John Wiley and Sons, New York.

Kleinrock, L. 1975. *Queuing Systems, Volume I: Theory.* John Wiley and Sons, New York.

Lan, C., Messer, C., et.al. 1992. Compromise Approach to Optimize Traffic Signal Coordination Problems during Unsaturated Conditions. *Transportation Research Record*(1360)112-120.

Leonard, J., Ramanathan, B., et.al. 1992. A Real-Time Information Processing Algorithm for the Evaluation and Implementation of ATMS Strategies. *Proceedings of the Intelligent Vehicles 1992 Symposium,* IEEE, 225-229.

Lin, E. 1992. *Large Scale Simulator for Testing Smart Traffic Lights.* MS Thesis, Electrical Engineering/Computer Science Department, MIT.

Little, J. and Morgan, J. 1964. Synchronizing Traffic Signals for Maximal Bandwidth. *Operations Research* (12)6,896-911.

Liu, C. and Kanaan, A. 1992. Evaluation of Freeway Improvement Alternatives using CORFLU. *Microcomputer in Transportation, Proceedings of the 4th International Conference,* American Society of Civil Engineers, 494-505.

Matson, T. and Smith, W. 1955. *Traffic Engineering.* McGraw-Hill Book Company, New York, 325-353.

May, A. and Montgomery, F. 1986. Avoiding the Need for Manual Intervention in Signal Control: A Case Study in Bangkok. *Road Traffic Control, IEE Conference Publication* (260)85-94.

Nelson, P. and Palacharia, P. 1993. Neural Network Model for Data Fusion in ADVANCE. *Proceedings of the Pacific Rim Transportation Technology Conference,* ASCE, NewYork.

Ore, O. 1963. Graphs and Their Uses. Random House, New York.

Papageorgiou, M. 1991. Concise Encyclopedia of Traffic and Transportation Systems, Pergamon Press, New York.

Ploss, G., Phillips, P., et.al. 1990. *MOTION - A New Traffic Control Concept based on Real-Time Origin-Destination Information.* Elsevier Science Publishing Company, 633-653.

Rathi, A. and Santiago, A. 1990. Variability Assessment for TRAF-NETSIM. *Journal of Transportation Engineering*(116)734-743.

Robertson, D. 1969.TRANSYT: A Traffic Network Study Tool. *Transport and Road Research Laboratory Report 253.*

Roberston, D. and Bretherton, R. 1991. Optimizing Networks of Traffic Signals in Real Time - The SCOOT Method. *IEEE Transactions on Vehicular Technology,* (40)11-15.

Sarkar, N. 1993. *Measures of Effectiveness and Independent Arc Velocities in a Traffic Simulator.* BS Thesis, Electrical Engineering/Computer Science Department, MIT.

Schul, A. 1955. The Probability Theory Applied to Distribution of Vehicles on Two-Lane Highways. The Eno Foundation for Highway Traffic Control, Saugatuck.

Shepherd, S. 1994. Traffic Control in Over-Saturated Conditions. *Transport Reviews*(14)13-44.

Smith, M. 1979. Traffic Control and Route Choice: A Simple Example. *Transportation Research* (13B)289-294.

Strang, G. 1986. *Introduction to Applied Mathematics.* Wellesley-Cambridge Press, Wellesley, Massachusetts, 137-154.

Strong, D. And Tompkins, R. 1991. *ITE 1991 Compendium of Technical Papers,* 224-228.

Sullivan, E., Staley, C., et.al. 1993. Simulation and Data Management Environment for Off-Line Testing of IVHS Technologies and Strategies. *Proceedings of the Pacific Rim Transportation Technology Conference,* ASCE, New York.

Texas Transportation Institute. 1984. *PASSER II: User's Guide,* Texas A&M, College Station, Texas.

Williamsen, L., Bolland, J., et.al. 1993. Multi-Modal Modelling in Congested Networks. *Traffic Engineering and Control*(34)294-301.

Wright, C., et. al. 1989 Graph Theory and Traffic Management: A Review of Recent Progress and Some Potential Applications. *Traffic Engineering and Control,* June 1989.