OPTIMAL SEQUENCING OF TRAFFIC STREAMS AT A SIGNALIZED JUNCTION

A Thesis Presented

by

Parham Tavakolian

To

The Department of Civil and Environmental Engineering

in partial fulfillment of the requirements

for the degree of

Masters of Science

in

Civil Engineering

in the field of

Transportation Engineering

Northeastern University

Boston, MA

Spring 2011
ABSTRACT

At a signalized intersection, efficiency can be gained by running compatible traffic streams in parallel, while incompatible streams run in series. In an intersection with an asymmetric clearance times, some sequences require less clearance time than others. In a simple intersection with a limited number of sequences, finding the optimal sequence is trivial. However in a complex intersection with a large number of traffic streams or unusual geometry, the optimal sequence is not obvious, and the number of possible sequences can be too large to check manually. An optimal sequence increases intersection capacity and minimizes delay by allowing an intersection to run with a shorter cycle length while serving the demand of all the traffic streams.

Previous research has proposed methods for finding the optimal sequence. However, counterexamples given in the literature as well as in this thesis show that their algorithms sometimes fail to find the optimal sequence. This thesis applies exhaustive implicit enumeration using the branch and bound algorithm together with a network linear programming (LP) formulation for finding the minimum cycle length for a given sequence. Network LP is extremely efficient. Branch and bound is efficient because it uses the concept of conflict groups to limit the search to feasible sequences. In addition, the algorithm supports simultaneous start and staggered start constraints, along with the usual conflict constraints.

For Chapel Hill, a test intersection used in the literature, our method finds not only the same solution reported by others, it finds a better solution, with a 12.5% reduction in necessary cycle length and 18% increase in capacity. In a test of a complex U.S. intersection, Charles Circle with 18 traffic streams, the algorithm found a more efficient solution than those known previously. Computer processing time for all examples was under 3 seconds.
Table of Contents

Abstract i
Table of Contents ii
List of Figures iv
List of Tables vi
Acknowledgements vii

Chapter 1) Introduction 1
  1.1 Overview 1
  1.2 Objectives 2
  1.3 Structure of the Thesis 2

Chapter 2) Terminology and Assumptions 4

Chapter 3) Topic History and Previous Researches 9
  3.1 History 9
  3.2 Analysis of Problem Structure 11
  3.3 The Compatibility Graph and Stages 11
  3.4 The Incompatibility Graph and Conflict Groups 12

Chapter 4) Formulation 14
  4.1 A Formulation on The Directed Incompatibility Graph 14
  4.2 Formulation 15
  4.3 Circuit Analysis on The Directed Incompatibility Graph 19
  4.4 Formulation Example 24
    4.4.1 Example 1: Identical Streams with Asymmetric Clearance
### List of Figures

**Figure 1**: Traffic streams at a 4-legged intersection  
4

**Figure 2**: Asymmetric Location of the Conflict Point Between Stream 2 and Stream 9  
6

**Figure 3**: Leading Left versus Lagging Left Phase Sequences for a 4-leg, 8-phase Intersection  
7

**Figure 4**: Directed Incompatibility Graph for a 4-legged intersection  
14

**Figure 5**: Formulating Staggered Start Constraints  
22

**Figure 6**: Circuit Analysis with a Simple 4-legged Junction  
24

**Figure 7**: The Five Stream – Odd-Hole Junction, Incompatibility Graph  
26

**Figure 8**: Three Arm Junction With 6 Streams & Single Ped Phase  
27

**Figure 9**: Signal Timing for a Three Arm Junction with all Pedestrian Phase  
28

**Figure 10**: Augmented Tree (Ch.5’s example)  
33

**Figure 11**: Network Simplex Algorithm for Minimizing c, given F  
36

**Figure 12**: Numerical Example for a Network Simplex Algorithm  
38-39

**Figure 13**: Searching Using the Conflict Group Permutation Array  
41

**Figure 14**: Branch and Bound Algorithm  
45

**Figure 15**: 4-legged Intersection, Webster’s Formula Solution  
50

**Figure 16**: 4-legged Intersection, Augmented Tree  
51

**Figure 17**: Chapel Hill, Layout  
57

**Figure 18**: Chapel Hill, Optimal Solution Timing Plan  
59

**Figure 19**: Chapel Hill, Directed Incompatibility Graph  
59
**Figure 20:** Chapel Hill, Augmented Tree 60

**Figure 21:** Stage-base result comparison 61

**Figure 22:** 4-legged Intersection (optimal solution timing plan) 63

**Figure 23:** Charles Circle – Case 1 (Layout) 65

**Figure 24:** Charles Circle – Volume Counts 66

**Figure 25:** Charles Circle- Case 1 (Solution Timing plan) 68

**Figure 26:** Charles Circle – Case 1 (Augmented Tree) 69

**Figure 27:** Charles Circle – Case 2 (Layout) 70

**Figure 28:** Charles Circle- Case 2 (Solution Timing plan) 62

**Figure 29:** Charles Circle – Case 2 (Augmented Tree) 73
## List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Table 1</strong>:</td>
<td>Network Simplex Algorithm Example</td>
<td>37</td>
</tr>
<tr>
<td><strong>Table 2</strong>:</td>
<td>Numerical Example Critical Circuit</td>
<td>37</td>
</tr>
<tr>
<td><strong>Table 3</strong>:</td>
<td>4-legged intersection input data</td>
<td>62</td>
</tr>
<tr>
<td><strong>Table 4</strong>:</td>
<td>4-legged intersection - Clearance Times Matrix</td>
<td>62</td>
</tr>
<tr>
<td><strong>Table 5</strong>:</td>
<td>4 - legged intersection Arc List</td>
<td>64</td>
</tr>
<tr>
<td><strong>Table 6</strong>:</td>
<td>4-legged initial sequence Matrix</td>
<td>64</td>
</tr>
<tr>
<td><strong>Table 7</strong>:</td>
<td>Chapel Hill input data</td>
<td>58</td>
</tr>
<tr>
<td><strong>Table 8</strong>:</td>
<td>Chapel Hill - Clearance Times Matrix</td>
<td>58</td>
</tr>
<tr>
<td><strong>Table 9</strong>:</td>
<td>Charles Circle – Case 1 - input data</td>
<td>67</td>
</tr>
<tr>
<td><strong>Table 10</strong>:</td>
<td>Charles Circle - Case 1- Clearance Times Matrix</td>
<td>67</td>
</tr>
<tr>
<td><strong>Table 11</strong>:</td>
<td>Charles Circle – Case 2 - input data</td>
<td>71</td>
</tr>
<tr>
<td><strong>Table 12</strong>:</td>
<td>Charles Circle - Case 2- Clearance Times Matrix</td>
<td>71</td>
</tr>
</tbody>
</table>
I would like to express my gratitude to my advisor, Professor Peter G.Furth, for his guidance, patience and invaluable support throughout this study. I would like to thank Mr. J.Parenti and Mr. R.Emery for their help and cooperation. I would like to thank Professor Daniel Dulaski for kindly accepting to read my thesis. I also wish to express my appreciation to my parents for their support and encouragement.
Chapter 1 Introduction

1.1 Overview

At intersections, traffic signals are installed to allocate the right of way to conflicting traffic streams for safety and efficiency. Traffic signals can be controlled in two different ways: **Fixed-time** control uses a fixed schedule for green, yellow and red intervals. This timing will not change during cycles, but various schedules may be used during AM peak hour, PM peak hour and off peak hours. **Actuated** control is the other option for traffic signal control, in which the operation of the signal depends on the actual traffic volume. In actuated control, traffic timing does not follow a fixed schedule timing, but is based on detectors data, so that various timing will be used in different cycles. At an extreme point of actuated control, where the intersection is almost saturated, actuated control behaves like a fixed-time control intersection, with maximum green time constraints governing the operation.

Since fixed timing is widely used in urban areas, this study is concentrated on fixed-time control for traffic signals. The objective is to decrease cycle length as much as possible, in order to achieve the minimum delay in the intersection. In an intersection with a large number of traffic streams, unusual geometry, asymmetric clearance times, and other properties can change a simple intersection to a complex junction, the optimal sequence at a signalized intersection can increase intersection efficiency. An efficient traffic stream sequence allows compatible streams to run at a same time while conflicting streams run in series. Searching for the optimal sequence, that is, the one that meets all the capacity and clearance constraints while requiring the shortest cycle length, is not a trivial problem. To find the optimal solution, all possible combinations between streams should be checked, however, at complex intersections, this number of combinations is too many to check by hand. A practical algorithm to search all sequences to find the optimal sequence will be a significant achievement for future designs.

Several approaches have been studied in past to find the optimal sequence, but have not been successful in complex intersections. In this study, an optimization algorithm is developed to search faster and easier.
1.2 Objectives

Common objectives that have been advanced for optimizing sequence for fixed time control [1] are (1) minimizing the necessary cycle time for a given maximum degree of saturation, (2) maximizing reserve capacity for a given cycle length, (3) minimizing vehicular delay. For actuated control, it is common to seek a sequence that minimizes necessary cycle time. Of course, the three objectives are highly correlated, and a signal control program that minimizes necessary cycle time will usually be optimal for other objectives as well. There are also objectives that are harder to quantify, such as pedestrian convenience, correspondence with motorist expectations, and effect on traffic distribution. Therefore our approach is to concentrate on one objective with the goal of finding all the signal sequences that perform “well” in a sense that may be defined by the user (e.g., within 10% of the optimal solution, or better than the existing sequence). Usually the number of “good” solutions thus obtained will be manageable so that a more detailed analysis can be carried out with respect to whatever objectives the engineer responsible desires.

The objective on which the program concentrates is minimizing cycle time. As will be shown, the model serves equally well for maximizing reserve capacity. While the model could also be adapted to non-linear objectives such as minimizing delay, it seems more efficient to use available methods that take the sequence as an input [2], using sequences found to be “good” with respect to a linear objective such as minimizing cycle time.

1.3 Structure of the Thesis

In chapter 2, terminology and assumptions that are used in programming and modeling the study intersections are discussed. In chapter 3, different views of approaching the optimal sequence problem are discussed. The approaches, taken by previous research studies are explained and a chosen approach for this study is discussed. Counterexamples of previous studies are also mentioned in chapter 3. Chapter 4 contains the mathematical formulation that of the sequencing problem. Chapter 5 explains the network simplex optimization algorithm for minimizing cycle length for a given sequence. Chapter 6 explains the branch and bound algorithm which is used in outer level of optimization. In chapter 7, a model extension using Webster’s cycle length formula
is described. Model limitations are also presented in chapter 7. Chapter 8 is an overall view on program including noticeable hints on how to use the program. Chapter 9 includes 3 examples of applying the new algorithm. The first two examples are cases studied in earlier research. The last example is a local complex intersection optimized by the program and compared to the current timing plan in the intersection.

In chapter 10, concluding remarks are presented along with recommendations for future works.
Chapter 2 Terminology and Assumptions

In an intersection where two or more streets meet each other, the variety of traffic streams makes the intersection an interesting point for traffic planners. In traffic engineer studies, each approach has one or more traffic streams or streams, which are the primary element in traffic timing plans. A stream is a group of vehicles that shares a common queuing space and signal at intersection. Figure 1 displays an example intersection with nine different streams including vehicular, pedestrian, and public transport (tram or trolley) streams.

![Diagram of traffic streams at an intersection]

*Figure 1: Traffic Streams at a 4-legged intersection*

A traffic stream may be controlled in several ways: permitted in which a stream is allowed to move carefully at a same time with vehicles in a conflicting stream. In this type, vehicles on a permitted stream should wait for a safe interval to pass through other streams that are using same space at intersection. A second way is protected, in which a stream receives green light with no conflicting stream allowed to run at the same time. (Of course, other non-conflicting streams...
may have a green light at the same time.) The last way is protected – permitted, a combination of two previous types. It is common for left turns to have protected - permitted signal which first allows left turn by exclusive green signal arrow, then the permission to run during green time of the opposite through movements. During the permitted phase, left turners should wait for a safe gap to cross the intersection.

Every stream has a green, yellow and red interval during a signal cycle. Similar to previous research, this study applies the restriction that a stream may not have more than a single, continuous green period per cycle. The duration of each stream’s yellow phase is assumed given, based on vehicles speeds, geometry and local policy. A set of streams that share the right of way at any point in time is a stage in the cycle. In Figure 1, streams 3 and 9 could form a stage.

Streams that may cause a collision if run at a same time are incompatible. Two mutually incompatible streams make a conflict pair. Incompatible streams can’t use the intersection space at the same time and both should be protected. Streams in conflict must clear the intersection before other incompatible stream approaches the same space. The clearance time between two incompatible streams is the minimum time between the start of one stream’s red and the start of the next stream’s green; it is usually in the range of 0 to 3s, depending on the intersection geometry, vehicle speeds and local policy. For some streams, such as left turn, more clearance time may be needed. In this case, some manuals define exceptions for clearance time. i.e. clearance time for left turn can be greater than 6s [3]. At a complex intersection, heavy vehicles will also need more clearance time than passenger vehicles. It is common to express the clearance times in a clearance time matrix, with one row and one column per stream, for which cell \((i,j)\) is empty if streams \(i\) and \(j\) are compatible, and otherwise is the clearance time from the start of stream \(i\)’s red to the start of stream \(j\)’s green.

In reality, clearance times can be asymmetric based on intersection geometry and conflict streams. Asymmetric clearance time makes the stream sequencing issue more important. In common U.S. practice, clearance (all-red) time when changing the right of way from stream \(i\) to conflicting stream \(j\) does not consider stream \(j\) at all; it is set to allow the last stream \(i\) car (passing the stop line at the end of yellow) to clear the entire intersection. In contrast, practice in England [4] and many other European countries is to set clearance time to ensure that conflict
will be avoided between the last car of stream i and the first car of stream j (assumed to accelerate from a stop at the stop-line at the start of green). This principle often leads to asymmetric clearance times. For example, in a typical intersection layout like Figure 2, the conflict point between streams 2 (a through movement) and 9 (opposing left) is close to the stop line of stream 2 and far from the stop line of stream 9. For this reason, no clearance time is needed when stream 2 yields to stream 9, while a clearance time of 2s or so is needed when stream 9 yields to stream 2. In general, lagging left turns require less clearance time than leading left turns, an important consideration for sequencing traffic streams. In the example shown in Figure 3, two different sequences are shown, one with lagging and one with leading left turns. In lagging left turns, clearance time between streams (2, 8) and streams (3, 9) is zero and for streams (3, 9) and streams (5, 11) is 2 seconds. The same value is used for conflicts (5, 11) and (6, 12). The total clearance time per cycle using lagging left turns will be 4 seconds. In the same example, leading left turns need a clearance time of 2 seconds to clear the intersection for start of opposing through movement, for a total clearance time of 8 seconds. By comparing calculations, it is obvious that lagging left turns require less clearance time than leading left turns.
In this study, in addition to clearance time matrix, traffic stream *volume* is needed for calculation. Each stream has a given *arrival rate*, with an arrival process assumed independent of the signal timing. The *saturation flow* rate of each stream is assumed given and independent of whether other compatible streams are green (thus the method is limited in how accurately it models permitted conflicts), and is assumed to apply during a stream’s *effective green time*. *Lost time*, which is assumed to be given for each stream, is the difference between a stream’s *split* and the effective green time. Split for each lane group is a time that the lane group controls the intersection, including green and *change interval*. Change interval for each stream is its yellow plus all red period. For normal vehicular streams, lost time is usually 3 or 4 seconds to account for losses at the start and end of green. For pedestrian streams (crosswalks) and exclusive bus/tram streams, the necessary green time is usually a fixed minimum that depends on intersection geometry. There is a needed time for pedestrian stream and exclusive public

![Diagram](https://example.com/diagram.png)
transport to travel through the intersection which is independent of cycle length and depends on intersection geometry. For such streams the saturation flow rate will be close to infinity, and the minimum split may be taken as the lost time.

Streams that are compatible usually don’t constrain each other’s timing. An exception is when safety or operations efficiency requires a simultaneous start and staggered start. Simultaneous start might be used for opposing movements with through and permitted left turns, in order to discourage left turns until the opposing through traffic has cleared. The other coordination constraint is staggered start, in which one stream’s green must start a few seconds before or after another’s. An example of this constraint is the leading pedestrian interval (LPI), which gives the right of way to the pedestrian by starting the crosswalk earlier than right turn vehicles on the parallel street. Turning cars are permitted to make turn while pedestrians are on crosswalk but according to right of way, vehicular stream should yield to pedestrians. Staggered start can also be used to provide a green wave for vehicle or pedestrian flows that pass through more than one signal at a complex intersection.
Chapter 3  Topic History and Previous Research

3.1  History

The signal timing plan for signalized junction is designed by setting the sequence of phases or streams and their green times. At first, this problem was studied assuming the sequence was known. (Webster [5]) Finding and optimal cycle length and splits, the “design problem” including phase sequence has been investigated by Stoffers [6] and Zuzarte Tully and Murchland [7]; Tully developed, on the basis of Stoffer’s work, the computer program SEQUENCE GENERATOR (SQGN) which determines all possible stages and maximum stage sequence that satisfy some natural criteria.

The optimization problem can be formulated in two ways. One is the “stage-based” approach, which uses the compatibility graph; the other is “group-based” (or “phase-based” or “stream-based”) approach which is based on conflict groups and the incompatibility graph. The stage-based approach assumes the stage change intervals and therefore the intergreen times to be fixed; only the duration of the stages will be regarded as variable. A group-based approach seems to be appropriate for European conditions as in the current control practice two clearance times are calculated for each pair of conflicting movements depending on two cases, example as shown in Figure 3.

Many researchers have explored the problem of finding an optimal sequence for a signalized intersection, beginning with the fundamental paper of Stoffers [6], whose analysis introduced both the compatibility graph and the incompatibility graph. Unfortunately, he concluded erroneously that in an optimal sequence, each stage in the signal cycle should be a maximal set of mutually compatible streams. Zuzarte Tully and Murchland [7], Heydecker and Dudgeon [8], Gallivan and Heydecker [9], and Heydecker [2] build on this conclusion, developing methods for optimally sequencing stages so constructed. Cantarella and Improta [10] take another approach, also suggested by Stoffers, arguing that the minimum cycle length is governed by maximal sets of mutually incompatible streams known as blocking group or conflict groups. Cantarella and Improta [10] described the individual signalized junction by a binary-mixed-integer-linear program which is based on knowledge of the flow parameters, the geometrical characteristics of the junction and the simultaneous crossing incompatibilities between streams. In their paper [10], an adimensional formulation of the previous binary-mixed-integer-linear program model was
used with a solution method based on graph theory. The Cantarella and Improta [10] approach can be thought of as a generalization of Webster’s method [5], based on an idea first presented by Stoffers [6], namely that the minimum cycle time can be determined by considering sets of mutually incompatible streams. However, Dickson and Murchland [11] and Moller [12] present counterexamples that demonstrate that both the maximal compatibility stage approach and the maximal conflict group approaches can result in suboptimal or misleading solutions. Counterexamples as discussed in Dickson and Murchland [11] are:

1) If the incompatibility graph is an odd-hole with five nodes, that is, a pentagon. An example is found in chapter 4 (Figure 7)

2) A three-arm junction with an exclusive all pedestrian phase. This example is also elaborated in chapter 4 (as shown in Figure 8)

Indeed, researchers using both approaches recognize that while their approach works most of the time, they do not always (Cantarella and Improta [10]; Gallivan and Heydecker [9]; Silcock and Sang [2]).

Cantarella and Improta [10] formulated the problem without imposing any stage constraint, but because their formulation failed to recognize the special network structure of the problem, they found it intractable for all but the simplest junction.

Moller [12] further develops the network analysis of the problem by examining circuits in the directed incompatibility graph. However, his analysis only goes as far as establishing conditions that an optimal sequence must satisfy, as opposed to finding the optimal sequence.

Recognizing that traditional operations research methods had not yet solved the problem, Palaniswami and Krishnamoorthy [13] used neural networks in a heuristic approach to finding the optimal stream sequence.

In this study, we extend and integrate the formulation of Cantarella and Improta [10] and the network representation of Moller [12] to develop a solution methodology that quickly finds the stream sequence and splits that minimize cycle time for complex junctions. The two key features of the solution methodology are a highly efficient network simplex algorithm used to find the minimum cycle length for a given sequence, and an efficient, conflict-group based formulation that speeds the execution of the branch and bound search for the optimal sequence. Our
formulation also admits staggered start and simultaneous start constraints between compatible streams, constraints that often play a role in signal sequencing. It can also be applied to maximize reserve capacity.

3.2 Analysis of Problem Structure

Stoffers’ analysis, which influenced many of those who followed, centered around two graphs: the compatibility graph, in which each stream is a node and an undirected arc connects stream pairs that are compatible, and its complement, the incompatibility graph, in which each stream is node, and an undirected arc connects stream pairs that are incompatible.

3.3 The Compatibility Graph and Stages

When viewed globally, it can be said that the signal cycle moves from one stage to the next, and the compatibility graph gives insight into stages. Because only streams that are mutually compatible can appear together in a stage, a stage must consist of a completely connected set of streams, i.e., a set of streams (nodes) that in the compatibility graph have an arc connecting every stream pair. A clique in the compatibility graph (also called by some a maximal clique) is a completely connected set to which no additional node (stream) can be added while keeping it a completely connected set. Stoffers and others reasoned that if a certain set of streams have the right of way, and an additional stream could be added without introducing a conflict, it would be inefficient not to add it, so that therefore stages should consist only of cliques in the compatibility graph.

Zuzarte Tully and Murchland [7] developed a method, based on Stoffers, to exhaustively generate all possible sequences of stages, in which stages are compatibility cliques. The requirement that no stream have more than one start per cycle forces stages containing a given stream to be contiguous. However, the requirements that (a) stages always be compatibility cliques and that (b) streams have exactly one start per cycle can easily lead to a suboptimal solution, or to no feasible solution. The reason is clear; if there are streams with few
incompatibilities, requiring them to be green when no conflicting stream is green and requiring that all their green times be contiguous can lead to deadlock.

Nevertheless, the idea of the signal cycle as a sequence of stages of compatibility clique has persisted, and several researchers have tried to find optimal sequences using stage sequences as specified by Zuzarte Tully and Murchland [7], Heydecker [4], Moller [12]. Gallivan and Heydecker [9] recognized problems with the stage sequence approach, and called for a stream-based (rather than stage-based) approach; yet that paper continues to use Zuzarte-type stage sequences as a basic element of their solution procedure.

3.4 The Incompatibility Graph and Conflict Groups

A clique in the incompatibility graph, also known as a (maximal) conflict group or a blocking group, also plays an important role in signal timing. A conflict group is a group of streams that are mutually incompatible for which there is no other stream that is incompatible with all the streams in that group. Clearly, the splits of the members of a conflict group cannot overlap, and so the cycle must be at least as long as each conflict group, where the “length” of a conflict group is the sum of its member’s green plus yellow times, plus clearance times. Stoffers [6] points out this lower bound, and also offers some mathematical conditions related to the stage-stream incidence matrix under which the longest conflict group determines the minimum cycle time. (However, his analysis ignores clearance times, which can depend on the sequence, thus adding another dimension to the problem.)

Selecting stream sequence and determining cycle length based on the longest or critical conflict group is standard practice for many engineers. As far as sequencing streams within a conflict group, it is easy to manually find the sequence that minimizes total clearance time because conflict groups rarely have more that 4 streams, and the number of sequences within a conflict group of n members, excluding simple rotations, is (n – 1)!, i.e., only 6 for a conflict group of 4 streams. This approach appears to work for most intersections.
However, as Dickson and Murchland [11] point out in an unpublished note referenced in 1977, minimum cycle lengths can be longer than the critical conflict group. They give two clear and not-unrealistic counterexamples, both of which are further analyzed in this study. One, repeated in some length in both Moller [12] and Cantarella and Improta [10], is a 5 stream intersection in which there is a five-node circuit in the incompatibility graph (Figure 7). This intersection fails to meet Stoffers condition of unimodality. Cantarella and Improta express the problem as the presence of an “Odd-Hole”, a graph theory term indicating a circuit of more than three nodes with an odd number of nodes. In that example, if each stream requires an equal amount of green time, the conflict group approach underestimates minimum cycle length by 50%. The second counterexample (Figure 8), a 3-arm junction with six vehicular streams and a single all-way pedestrian stream, is repeated in Cantarella and Improta. Depending on the demand data, its minimum cycle length can be governed by a sequence that is not a conflict group, and in this example there is no “Odd-Hole” to blame.
Chapter 4

4.1 A Formulation on The Directed Incompatibility Graph

A still stronger analysis tool is the directed incompatibility graph, in which each node represents an event – in this case, a stream’s green start. Each node $i$ can be labeled with a value $u_i$, the (as yet unknown) start time of stream $i$. If two streams are specified to have a simultaneous start, a common node can represent the start time for both streams. Arcs represent the time elapsed between two events (two green starts), and carry minimum or maximum value constraints. The network has $N$ nodes and $A$ arcs. The incompatibility graph for the example intersection in Figure 1 is shown in Figure 4.

![Directed Incompatibility Graph](image)
4.2 Formulation

Let

\[ u_i = \text{start time of stream i’s green} \]

\[ (v/s)_i = \text{ratio of stream i’s arrival rate to its saturation flow rate} \]

\[ f_i = \frac{1}{\text{(maximum permitted degree of saturation for stream i) [e.g., if the maximum permitted degree of saturation for stream i is 0.8, } f_i = 1.25]} \]

\[ L_i = \text{lost time for stream i before its red period} \]

\[ L_{ij} = \text{clearance time between start of stream i’s red and start of stream j’s green} \]

\[ c = \text{cycle length} \]

In a cyclical operation, the starting point may be defined arbitrarily. Therefore let the cycle start be at the start of stream h’s green, where stream h may be any arbitrarily chosen stream. With the cycle start thus established, there is an unambiguous sequence for each pair of incompatible streams and for each pair of streams to which the staggered start tactic applies, which may be expressed in the variable

\[ F_{ij} = 1 \text{ if stream i follows stream j in the cycle, 0 otherwise} \]

The overall sequence array is \( F = \{ \{ F_{ij} \} \} \). The variable \( F_{ij} \) has no meaning for stream pairs that are compatible and have no staggered start constraint.

Of course, there is the reciprocity constraint

\[ F_{ij} = 1 - F_{ji} \quad (1) \]

By definition,

\[ u_h = 0 \quad (2) \]

\[ F_{hj} = 0 \text{ and } F_{jh} = 1 \text{ for all } j \quad (3) \]
If two streams $i$ and $j$ are incompatible, stream $j$ may not start its green until stream $i$’s service is ended and the clearance time has been observed. Therefore, if stream $j$ follows stream $i$, the necessary time between the start of stream $i$’s green and the start of stream $j$’s green is

$$ u_j - u_i \geq L_i + cf_i(v/s)_i + L_{ij}, \text{ if } F_{ij} = 0 $$

(4a)

The right hand side is the green plus yellow time necessary for stream $i$ (lost time plus an effective green time long enough to keep stream $i$’s degree of saturation below the specified limit), plus the clearance time. If, on the other hand, stream $j$ precedes stream $i$ in the cycle, it follows stream $i$’s green of the previous cycle, so that on the left hand side of (4a) $u_i$ is replaced with $(u_i - c)$, resulting in

$$ u_j - (u_i - c) \geq L_i + f_i c (v/s)_i + L_{ij}, \text{ if } F_{ij} = 1 $$

(4b)

Equations (4a) and (4b) can be combined thus:

$$ u_j - u_i \geq L_i + cf_i(v/s)_i + L_{ij} - c F_{ij}, \text{ streams } i \text{ and } j \text{ incompatible} $$

(4)

Constraints 1 – 4, together with the requirement that $F_{ij}$ is a binary variable and a non-negativity constraint on $c$, express (in a more compact form) the formulation of Improta and Cantarella [10]. Note that this formulation has no explicit dependence on stages. This formulation will also be extended to consider staggered start constraints, which can be specified by the parameters

$$ S_{\text{min}}_{ij} = \text{minimum time between stream } i \text{’s green start and stream } j \text{’s green start} $$

$$ S_{\text{max}}_{ij} = \text{maximum time between stream } i \text{’s green start and stream } j \text{’s green start} $$

where $S_{\text{max}}_{ij} \geq S_{\text{min}}_{ij}$. By setting $S_{\text{max}}_{ij} = S_{\text{min}}_{ij}$, the stagger will be fixed. In addition, $S_{\text{max}}_{ij}$ and $S_{\text{min}}_{ij}$ may be negative. An example in which negative values may be appropriate is when stream $j$ physically precedes stream $i$ in the intersection geometry such that through traffic first passes through stream $j$ and then through stream $i$. In such a case, it may be desired that stream $j$’s green not begin more than a few seconds before stream $i$’s green to prevent a blockage at
stream i’s stop-line. In a cyclical process, specifying the stagger from i’s start to j’s start also specifies the stagger from j’s start to i’s; therefore, one may either specify limits for j following i (\(S_{\text{min}ij}, S_{\text{max}ij}\)), or, for i following j (\(S_{\text{min}ji}, S_{\text{max}ji}\)), but not both. Therefore the constraints on the relative start of the two streams are:

\[
\begin{align*}
    u_j - u_i & \geq S_{\text{min}ij} - c F_{ij}, \text{ staggered start with stream j waiting for stream i} \\
    u_j - u_i & \leq S_{\text{max}ij} - c F_{ij}, \text{ staggered start with stream j waiting for stream i}
\end{align*}
\]

It will be helpful to rewrite inequality (6a) so that it has the same direction as constraints 4 and 5, and to replace \(F_{ij}\) with \(1 - F_{ji}\).

\[
\begin{align*}
    u_i - u_j & \geq - S_{\text{max}ij} + c(1 - F_{ji}), \\
    u_j - u_i & \geq - S_{\text{max}ji} + c(1 - F_{ij})
\end{align*}
\]

or exchanging i and j,

\[
\begin{align*}
    u_i - u_j & \geq - S_{\text{max}ji} + c(1 - F_{ji})
\end{align*}
\]

Relations 4 – 6 can be summarized in a general form

\[
\begin{align*}
    u_j - u_i & \geq y_a + z_a c - F_a c
\end{align*}
\]

where \(a\) is the index of the constraint that expresses a minimum interval from the green start of stream i to the green start of stream j and

\[
F_a = F_{ij}
\]
\[
y_a = \begin{cases} 
L_i + L_{ij} & \text{if stream i and j are incompatible} \\
S_{\text{min}}_{ij} & \text{if streams i and j have a staggered start specified for j following i} \\
-S_{\text{max}}_{ji} & \text{if streams i and j have a staggered start specified for i following j}
\end{cases}
\]

\[
z_a = \begin{cases} 
(f_i(V/S)_i & \text{if stream i and j are incompatible} \\
1 & \text{if streams i and j have a staggered start specified for j following i} \\
0 & \text{if streams i and j have a staggered start specified for i following j}
\end{cases}
\]

If it is specified that two streams i and j should have a simultaneous start,

\[u_i = u_j, \text{ if streams i and j must have a simultaneous start}\]

For minimizing the necessary cycle length subject to specified degrees of saturation, the optimization program is

\[
\text{Minimize } c \text{ with respect to the } u_i \text{'s, the } F_{ij} \text{'s, and } c, \text{ subject to} \\
(1) - (3), (7), (11) \\
F_{ij} = 0 / 1 \\
c > 0
\]

For maximizing reserve capacity, \( f_i \) in equation 4 is replaced with \( f \), and the optimization program is to maximize \( f \), subject to the same constraints as in (12), with \( c \) given as a parameter. (Naturally, there is a strong relationship between the minimum \( c \) and the maximum \( f \), which is explored in detail by Catarella and Improta [10].) Minimizing delay can be formulated by piecewise linear approximation, as done by Improta and Cantarella [10]. Because green start times are cyclical, there is no practical difference whether \( u_i \) equals a certain value, say, \( u_0 \), or \( u_0 + mc \), where \( m \) is any integer, positive or negative. Shifting start times by a multiple of \( c \) may be called a rotation. Depending on the search procedure used, it is possible to
generate several solutions that are rotations of each other, as pointed out by Heydecker [4], who presents a method for determining which sets of F are rotations of each other. However, while this approach is helpful when all the sequences are listed in advance (as they can be if stages are restricted to compatibility cliques, which is what Heydecker does), it is not practical when stages are not thus restricted because the number of possible sequences is exponentially large. Therefore, to avoid considering solutions that are merely rotations of each other, we define a canonical rotation as one that satisfies the relation

\[ 0 \leq u_i < c \]  

(12a)

This constraint is not included in program 12, but rather is used only to screen redundant solutions.

As long as offsets specified for staggered start constraints (Smin and Smax) remain smaller than c, as they normally are in what is generally considered an intersection problem, any feasible solution may be rotated into canonical form. If any offset exceeds c, non-canonical solutions cannot be excluded.

Program 12 is a binary mixed integer programming problem that can be solved by branch and bound with linear programming relaxations, as proposed by Improta and Cantarella. By exploiting the problem structure, we are able to develop improvements to both the linear programming and branch and bound algorithms that make the problem tractable for complex junctions.

### 4.3 Circuit Analysis On The Directed Incompatibility Graph

Moller’s [12] analysis focuses on circuits in the directed incompatibility graph. The sections that follow build on this analysis because of the insight it offers into problem structure and solutions. A well-known theorem from graph theory [14] is that, for feasibility, every (directed) circuit in such a graph must have length less than or equal to zero. Because the length of every circuit is a
function of the cycle length, this constraint can be transformed into a minimum cycle length demanded by each circuit. The circuit demanding the greatest cycle time is the \textbf{critical circuit}. Often, the critical circuit passes through streams that follow one another within a single cycle, e.g., a conflict group circuit 1-5-4-3-1 on T junction as shown in Figure 7. However, the critical circuit will sometimes pass through a sequence of streams that covers two or more cycles, e.g., 1-2-3-4-5-1 in Figure 7, in which the critical circuit covers two cycles. (This is the case with the “Odd-Hole” example mentioned earlier.) Roughly speaking, the cycle length demanded by a circuit \( k \) is the circuit length divided by an integer \( n_k \) representing the number of cycles (i.e., intersection control cycles) covered by the circuit.

Using the form of constraint 7, the no-positive-circuit condition for feasibility can be expressed as:

\[
\sum_{a \in A_k} (y_a + z_a c - F_a c) \leq 0 \quad \text{for every circuit } k
\]

where \( A_k \) = set of arcs belonging to circuit \( k \). With the realistic restriction that \( c > 0 \), circuits can be divided into two sets, depending on whether relation 13 implies a lower or upper bound on \( c \). Let \( \text{KL} \) be the set of circuits \( k \) for which

\[
\sum_{a \in k} y_a > 0
\]

(14)

These circuits impose a lower bound on \( c \), and are dominated by incompatibility relationships. Let \( \text{KU} \) be the set of circuits that do not satisfy relation 14; they impose an upper bound on \( c \) and are dominated by staggered start relationships, since \( y_a < 0 \) only for arcs representing staggered starts. Note that the partition into sets \( \text{KL} \) and \( \text{KU} \) is independent of the values of the decision variables.

For circuits \( k \in \text{KL} \), relation 13 can be satisfied with \( c > 0 \) only if

\[
\sum_{a \in k} F_a > \sum_{a \in k} z_a
\]

(15a)
in which case relation 13 provides a lower bound $\text{clb}_k$ on $c$, which can be expressed as

$$c \geq \text{clb}_k = \frac{\sum_{a \in A_k} y_a}{\sum_{a \in A_k} F_a - \sum_{a \in A_k} z_a}$$ (16)

If for any circuit $k \in \text{KL}$ relation 15a is not satisfied, the problem is infeasible; we may write

$$\text{clb}_k = \infty \text{ if } \sum_{a \in k} F_a < \sum_{a \in k} z_a \text{ and } \sum_{a \in k} y_a > 0$$

For circuits $k \in \text{KU}$, if

$$\sum_{a \in k} F_a > \sum_{a \in k} z_a$$ (15b)

relation 13 provides an upper bound $\text{cub}_k$ on $c$, which can be expressed as

$$c \leq \text{cub}_k = \frac{-\sum_{a \in k} y_a}{\sum_{a \in k} z_a - \sum_{a \in k} F_a}$$ (17)

If relation 15b is not satisfied for any circuit $k \in \text{KU}$, relation 13 is satisfied for any $c > 0$, and so we may write

$$\text{cub}_k = \infty \text{ if } \sum_{a \in k} F_a \leq \sum_{a \in k} z_a \text{ and } \sum_{a \in k} y_a \leq 0$$

One special case of a circuit belonging to set KU is the simple circuit formed by the pair of arcs i-j and j-i when streams i and j have a staggered start relationship. An example is shown in Figure 5, in which stream i might be a crosswalk whose green should begin between $S_{\text{minij}}$ and $S_{\text{maxij}}$ seconds before parallel vehicular stream j. Because $S_{\text{maxij}} \geq S_{\text{minij}}$, such a circuit belongs to set KU. Equations 1, 8 and 10 guarantee that $\sum F_a = \sum z_a$ for this circuit, so that it imposes no restriction on feasibility.
Consider the more typical case of a circuit of arcs representing incompatibility relationships and, from equation 9, all have \( y_a \geq 0 \). Assume also that relation 15a holds (otherwise, there is no feasible solution). Such a circuit belongs to set \( KL \), and bounds the cycle length with the limit:

\[
clb_k = \left( \sum L_i + \sum L_{ij} \right) / \left( 1 - \sum f_i(v/s)_i \right)
\]  \hspace{1cm} (19)

This is the well known formula for minimum cycle length, when applied to the critical conflict group (the conflict group for which clb is greatest). With minor modification (setting \( f_i \) to 1, inflating lost times by 50%, and adding an extra 5 s to lost time), this is Webster’s empirical formula for the “best” (not minimal) cycle length for fixed time control.[15]
\( c_{\text{Webster}} = \left[ 5 + 1.5 \left( \sum L_i + \sum L_{ij} \right) \right] / \left[ 1 - \sum \left( \frac{v}{s_i} \right) \right] \) \hspace{1cm} (20)

The problem of finding the minimum cycle length thus can be stated in terms of circuits on the directed incompatibility graph as follows:

\[
\begin{align*}
\text{Min } c &= \max_k [ \text{clb}_k(F)] \\
\text{Subject to} & \\
F_{ij} &= 0/1 \\
F_{ji} &= 1 - F_{ij} \\
c &\leq \text{cub}_k (F) \text{ for all circuits } k \in \mathbf{KU}
\end{align*}
\] \hspace{1cm} (21)

where the only decision variable is \( F \), the array of \( F_{ij} \)'s. Comparing with program 12, this formulation has no explicit reference to the \( u_i \)'s. Together, the network structure and relations 16 and 17 ensure that constraints 7 and 11 satisfies start times. Constraints 2 and 3 are unnecessary, although they could be included to reduce the problem size.

The circuit in program 21 whose lower bound governs the value of \( c \) is called the critical circuit, much like the critical path in an activity-on-arc network in the Critical Path Method (CPM). For the critical circuit, relation 13 holds as an equality; that is, the length of the critical circuit will be zero.

Program 21, however, is not a practical solution method, because in a cyclical operation, the number of circuits is unlimited.
4.4 Formulation Examples

Three examples illustrating situations in which the minimum cycle time is governed by a critical circuit that is not a conflict group will be presented to illustrate the formulation of the problem, and to motivate the necessity of a solution method that goes beyond compatibility and incompatibility clique. In all three examples, the allowed degree of saturation is 1 (so \( f_i = 1 \)).

4.4.1 Example 1:

Identical Streams with Asymmetric Clearance Times

The first is a simple 4-stream junction (Figure 6) with identical demands and lost times, but asymmetric clearance times. For each stream, we shall take \( L_i = 3 \) s and \( (v/s)_i = 0.4 \). Clearance times are either 0 or 2 s, depending on whether the stop line nearer the conflict point belongs to stream whose green is starting or ending. Therefore the lost time for a conflict pair is either 3s or 5s.

![Figure 6: Circuit Analysis with a Simple 4-legged Junction](image)
The directed incompatibility graph (part b of the figure) shows the four conflict groups: NB-EB, NB-WB, SB-EB, and SB-WB. Consider circuit SB-EB-SB passing through conflict group SB-EB (part c). Because it has only two members, sequence within the group is meaningless; of the two sequence variables $F_{SE}$ and $F_{ES}$, one must equal 0 and the other 1. The sum of the lost times in this circuit is 8 s, and the sum of the v/s ratio is 0.8. From equation 16, the minimum cycle length for this conflict group is $8 / (1 - 0.8) = 40$ s. It can easily be verified that the other conflict groups all have the same minimum cycle time.

Yet, the minimum cycle length for the intersection is not 40 s, but 50 s, because it is not governed by a conflict group circuit, but rather by the counterclockwise circuit SB-EB-NB-WB-SB (part d). The length of this circuit depends on the values of the sequence variables ($F_{ij}$'s). Clearly the most efficient sequencing, shown in the figure, is for streams NB and SB to run together, followed by streams EB and WB together, and then returning to NB and SB. The circuit SB-EB-NB-WB-SB has components:

$$
\sum y_a \quad \text{(lost and clearance times)} : \quad 20 \text{ s}
$$
$$
\sum z_a \quad \text{(v/s ratios)} : \quad 1.6
$$
$$
\sum F_a \quad \text{(cycles covered)} : \quad 2
$$

The minimum cycle length (equation 14) for this circuit is therefore $20 / (2 - 1.6) = 50$ s. (The reader may verify that clockwise circuit SB-WB-NB-EB-SB has a cmin of only 30 s.)

### 4.4.2 Example 2:

#### The Five-Stream “Odd-Hole” Junction

The second example is the “Odd-Hole” example proposed by Dickson and Murchland [11] and analyzed by both Moller [12] and Cantarella and Improta [10], a three arm junction with three vehicular streams and two pedestrian streams (See Figure 7). The lost times and (v/s) ratios are consistent with Cantarella and Improta’s Fig. 8b. The conflict groups all consist of two streams: 1-2, 2-3, 3-4, 4-5, and 5-1. In the most efficient sequencing, as shown, there is no “barrier” in the signal cycle, i.e., no time at which all the streams are red. The reader can verify that while $c_{\text{min}}$ for the conflict group circuits range from 40 to 60, the critical circuit is 1-2-3-4-5-1, a circuit covering two cycles, whose $c_{\text{min}}$ equals 120 s.
4.4.3 Example 3:

The Three-Arm Junction With an All-Pedestrian Stage

This example, also proposed by Dickson and Marchland [11], is analyzed by Cantarella and Improta [10] as another example for which the minimum cycle length may be greater than that demanded by any conflict group. It has 6 vehicular streams, and a set of crosswalks that run together as an all-pedestrian phase. Each pair of crossing or merging streams (Figure 8) is assumed incompatible. For the sake of brevity, we treat the set of crosswalks as a single streams. For the six vehicular streams, lost time and v/s data (Fig.8) are taken from Cantarella and Improta’s Table 1, except that volumes have been factored by 42/55 (without modification, the necessary cycle length will be more than 300 s); the pedestrian stream has been given 18 s. Clearance times are all zero.
The assumed signal sequencing, taken from Cantarella and Improta (their Fig. 12, excluding the suggested second green period), is shown in Figure 9. In the directed incompatibility graph (Figure 9), green start times are shown in bold for nodes on the critical circuit; for nodes off the critical circuit, earliest and latest green start times are shown. One can verify that the conflict group circuit demanding the greatest cycle length is 1-3-5-P-1. It has a minimum cycle time of 93.6 s. However, circuit 1-3-6-P-1, which is not a conflict group circuit, has a greater minimum cycle time (100 s) and is the critical circuit. The critical in this example covers only one cycle (i.e. $\Sigma F_q = 1$), but is not a maximal conflict group.
Interestingly, and perhaps bearing witness to the need for a better method for determining optimal stream sequence, the sequence given by Cantarella and Improta does not minimize cycle length. A better sequence, shown in Figure 9, is to move stream 6 so that it runs together with stream 1. With this arrangement, the conflict group circuit 1-3-5-p-1 is critical, and the minimum cycle time is 93.6 s.

However, to further illustrate Dickson and Murchland’s proposition that conflict groups do not always determine the minimum cycle length, even without asymmetric clearance times or odd holes, the problem may be reanalyzed with the volume on stream 1 reduced to 150. The original sequence (Figure 9) now minimizes necessary cycle length, and the critical circuit is 1-3-6-P-1, a circuit that is not a conflict group. Circuit 1-3-6-P-1 requires a cycle length of 70.2 s, while the next most demanding conflict group (1-3-5-P-1) requires a cycle length of only 63.8 s.
Chapter 5: Optimization Algorithm Finding Minimum Cycle Length, Given Sequence

Unfortunately, examining circuits in the directed incompatibility graph is not a practical optimization approach for junctions that are not simple, because the number of circuits becomes too large for simple enumeration of circuits. This study’s optimization approach, therefore, follows program 12 rather than program 20, but takes advantage of the structure of the directed incompatibility graph.

The optimization approach has two levels. The outer problem, discussed later, is searching for the optimal sequence array F; the inner problem, discussed in this chapter, is to find the minimum feasible cycle length \( c \) for a given sequence array \( F \).

A Network Simplex Algorithm

With \( F \) fixed, program 12 is a linear optimization problem which can be solved using the simplex algorithm, as suggested by Improta and Cantarella [10]. However, as is often the case, the network structure of the problem can be exploited to develop an implementation of the simplex algorithm that is far more efficient than the general simplex algorithm, yet preserves its desirable properties such as the ability to easily conduct sensitivity analysis.

A compact formulation of program 12, given \( F \), is as follows:

Program 12a: Minimize \( c \), subject to

\[
\begin{align*}
  & u_j - u_i - s_a = y_a + Z_a c \quad \text{for every arc a going from a node i to a node j} \\
  & s_a > 0 \quad \text{for all a} \\
  & c > 0
\end{align*}
\]  

where

\[
Z_a = z_a - F_a,
\]

\( s_a = \text{arc-specific surplus variable associated with each constraint 7a}, \)
and \( u_h \) is taken as a parameter with value 0 (rather than as a variable). We assume that constraint 11 has been incorporated into the network by defining a common node for stream sets with a specified simultaneous green start. Let \( A = \) number of arcs, and \( N = \) number of nodes in the resulting network (note that, due to simultaneous starts, it is possible that there be more than one arc from a node \( i \) to a node \( j \)). The problem has \( N \) decision variables \((c \text{ and } u_i \text{ for } i \neq h)\) and \( A \) equations, defining each constrained such that \( u_j - u_i \geq d_a \text{ arc's minimal length:} \)

\[
d_a = y_a + Z_a c.
\]

Note that the \( u_i \)’s are unrestricted in sign.

We define a **longest path tree solution** as a solution in which there is a tree, directed outward from a root at node \( h \) and spanning every node, such that for arcs in the tree, constraint 7 is binding (i.e., there is no surplus), and that for non-tree arcs, constraint 7 is satisfied with non-negative surplus. The arcs in the tree determine the green start times \( u_i \), which can be calculated by fanning out along the tree beginning at node \( h \) (where \( u_h = 0 \)) and applying constraint 7 with zero surplus. The tree is a longest path tree because it defines the longest path in network from node \( h \) to every other node; it may be likened to a tree in the Critical Path Method (CPM) for finding the earliest start time for each activity.

**Proposition 1.** Given \( c \), if there is a finite solution to the longest path problem (finding the longest path from node \( h \) to every other node), it is also a solution to program 12a. If there is no finite solution to the longest path problem, there is no feasible solution to program 12a for the given \( c \).

**Proof:** With \( c \) given, the \( d_a \)’s (arc lengths) are fixed, and the objective function of program 12a becomes arbitrary. If we take as an objective function

\[
\text{Minimize } \sum_{i \neq h} u_i
\]

Then program 12a becomes the longest path problem. Therefore, any finite solution to the longest path problem is a feasible solution to program 12a, and an unbounded solution to the longest path problem implies that there is no feasible solution to program 12a. \( Q.E.D. \)
The longest path problem is solved very quickly, especially with the size of networks involved in modeling even the most complex intersection. An unbounded solution occurs only when there is a positive directed cycle. There are many longest path algorithms that will detect positive cycles; e.g., the label correcting algorithm with first-in-first-out node processing in Ahuja, Magnanti, Orlin and Reddy [15]. In the network we have formulated in which every conflict or staggered start constraint gives rise to a pair of arcs, no directed path can have more than A/2 arcs without containing a cycle. Therefore, by tracking the number of arcs in a path preceding each node, the longest path algorithm detects a positive cycle when the number of arcs preceding a node exceeds A/2. If a positive cycle is found, its length can be determined as a function of c as

\[ L_k = y_k + c Z_k \]

where \( y_k \) and \( Z_k \) are, respectively, the sums of \( y_a \) and \( Z_a \) over the arcs in the cycle. If \( y_k > 0 \) and \( Z_k > 0 \), there is no feasible solution to 12 for any value of c. Otherwise, the length of the circuit will go to zero if c is changed to \( -y_k/Z_k \).

Because in a longest path tree solution \( u_i \) is simply a sum of the arc lengths of the path on the tree from h to i, \( u_i \) can expressed as a function of c in the form

\[ u_i = \alpha_i + \beta_i c \]

The coefficients \( \alpha_i \) and \( \beta_i \) are also calculated by fanning out from node h along the tree, beginning with \( \alpha_h = \beta_h = 0 \), and the recursions

\[ \alpha_j = \alpha_i + y_a \]
\[ \beta_j = \beta_i + Z_a \]

where a is the spanning tree arc going from i to j. Likewise, the surplus (“float”, in CPM terms) for each arc can be expressed in terms of c in the form

\[ s_a = y'_a + c Z'_a \]
where

\[ y'_a = \alpha_j - \alpha_i - y_a \]
\[ Z'_a = \beta_j - \beta_i - Z_a \]  \hspace{1cm} (23)

are the coefficients of arc a’s surplus. For arcs in the spanning tree, \( y'_a = Z'_a = s_a = 0 \).

**Proposition 2.** To every longest path tree solution there is a corresponding augmented tree solution that is an extreme point of program 12a, unless the problem is ill-formed.

**Proof:** In an extreme point solution to program 12a, \( N \) surplus variable must be non basic variables, set to zero, leaving the \( u_i \)’s to be determined by equation (7a). A longest path solution is not in itself extreme point (basic feasible solution), since it fixes at 0 only the surplus variables of the \( N - 1 \) arcs in the tree. To convert a longest path tree solution to an extreme point, \( c \) must be lowered until it drives the surplus variable of a non-tree arc to zero. The resulting extreme point may be called an **augmented tree solution** ; its set of non-basic arcs (arcs with surplus fixed at zero) is the longest path tree plus the arc whose surplus was driven to zero.

With which arc should the longest path tree be augmented? Looking at equation (22), arcs for which \( Z'_p = 0 \), or for which \( Z'_p > 0 \) and \( y'_p \geq 0 \), put no limit on a change in \( c \) (since \( c > 0 \)). Arcs for which \( Z'_p < 0 \) limit only how much \( c \) can increase; however, because our objective is to minimize \( c \), we are not interested in them. On the remaining arcs (and if there are none, the problem is ill-formed, as \( c = 0 \) is a feasible solution), the ratio \( r_a = -y'_a / Z'_a \) is the lowest value of \( c \) that arc a will allow; any lower value and its surplus would become negative. Because the greatest lower bound controls, designate as arc \( p \) (it becomes the “**pivot**” arc) the arc in that group with the greatest \( r_a \). Setting \( c \) equal to \( r_p \), \( s_p \) is driven to zero, making arc \( p \) the final non-basic arc, while the other arc’s surplus remains non-positive. The result is a basic solution, and is feasible. **Q.E.D.**

**Proposition 3.** If the sub-graph of non-basic arcs in an augmented tree solution contains a directed cycle, that augmented tree solution is an optimal solution, and that directed cycle is the critical circuit.
Proof: Because the sub-graph of non-basic arcs at an extreme point is a longest path tree plus arc p, it will contain one circuit, which may or may not be a directed cycle. Let arc p’s initial and terminal vertices be called i and j, respectively, as shown in Figure 10a. If in the (un-augmented) longest path tree i is a successor of j, the circuit is a directed cycle consisting of two parts: a part along the longest path tree from node j to node i, and then arc p going from i to j. The distance along the tree from j to i is $u_i - u_j$, so the length of the directed cycle is

$$L = u_i - u_j + (y_p + Z_p \cdot c) = \alpha_i - \alpha_j + y_p + c(\beta_i - \beta_j + Z_p) = -y'_p - cZ'_p = -s_p$$

Because $s_p = 0$ in an augmented tree solution, the length of the directed cycle is zero; it is therefore a critical circuit. Because $Z'_p > 0$, c cannot be lowered further without making the solution infeasible ($s_p$ will become negative). Therefore the solution is a local optimum, and because it is a linear optimization problem, it is a global optimum. Q.E.D.
Proposition 4. If the cycle contained in the subgraph of non-basic arcs in an augmented tree solution is not a directed cycle, the neighboring extreme point in the direction of improvement may be found by adding arc p to the longest path tree, removing the arc that shares a terminal node (call it node j) with arc p, and finding the augmented tree solution corresponding to the resulting longest path tree.

Proof: Suppose the circuit caused by the addition of arc p (as before, running from node i to node j) to the longest path tree is not a directed cycle as in Figure 10b. Let the arc in the (un-augmented) longest path tree that terminates at j be called arc x (so called because this arc will later exit the augment tree), and let its initial node be called k, as shown in Figure 10b. There must be two directed paths from node h to node j along the subgraph of non-basic arcs, one following the longest path tree via node k and arc x, the other following the longest path tree only as far as node i and then reaching node j via arc p. Either of these paths will determine \( u_j \) as a function of c, yielding two equations for \( u_j \):

\[
\begin{align*}
\alpha_i + y_p + c(\beta_i + Z_p) &= \alpha_i + y_p + c(\beta_j - Z'_p) \\
\end{align*}
\]

Because \( u_j \) must be the same by either path, these two equations may be solved for c:

\[
c = \frac{\alpha_i + y_p - \alpha_k - y_x}{Z'_p}
\] (24)

From equation 24, the reduced costs of surplus variable \( s_p \) and \( s_x \) can be seen to be \( 1/Z'_p \) and \( -1/Z'_p \) respectively, because c changes with the surplus of an arc in the augmented tree at exactly the same rate with which it changes with that arc’s y. Therefore, since \( Z'_p > 0 \), a feasible direction of improvement is to increase the surplus of arc x, lowering c. As its surplus becomes non-zero, arc x leaves the augmented longest path tree.

How much can c decrease, with \( s_x \) rising simultaneously? The removal of arc x from the augmented tree results in a new longest path tree (un-augmented, in which arc p has replaced arc x). This new longest path tree has a corresponding augmented tree solution, as described earlier, formed when lowering c reaches the point at which another non-tree arc’s surplus is driven to zero. That new augmented tree solution sets the limit on how far c can be reduced. That new augmented tree solution is a neighboring extreme point, since it is reached by one basis entrance (\( s_x \) enters) and one basis exit (the surplus of a new pivot arc exits). Q.E.D.
In a pivot, calculation effort can be minimized by recognizing that coefficients of the u’s and s’s change value only in the part of the network downstream of the pivot arc.

The algorithm for minimizing c given F is summarized in Figure 11. It assumes that a feasible or incumbent value of c, c_{INC} (perhaps feasible for some other array F, not necessarily for the current F) is known. If a feasible solution is unknown, the largest permissible (from a traffic engineering viewpoint) c can be used.

Convergence of the algorithm is guaranteed because it traces the path of the simplex algorithm, albeit with far fewer computations. Our expectation, confirmed by computational experience, is that very few pivots will be required, as the longest path tree that applies for a given value of c is likely to apply to a considerable range of values around c. The algorithm is therefore extremely efficient, requiring one pass of the longest path algorithm and a few pivots, each of which require a relatively small number of simple calculations.
Figure 11: Network Simplex Algorithm for Minimizing c, given F

1. Find Feasible Longest Path Solution
   1.1 Set $c = c_{\text{INC}}$
   1.2 Determine the minimum arc lengths for the given $F$ and $c$
   1.3 Apply the longest path algorithm to find the longest path from node $h$ to every other node. If a finite solution is found, it is a feasible solution to the original problem; identify its tree, set $i = h$, and go to step 2
   1.4 Otherwise, a directed cycle $k$ of positive length has been found. Find $y_k$ and $Z_k$ for this circuit (see eq. 24). If $y_k \geq 0$, STOP; there is no feasible solution with $c \leq c_{\text{INC}}$ for the given $F$
   1.5 Otherwise, decrease $c$ to $c = -y_k/Z_k$ and go to Step 1.2

2. Find Minimal $c$ for Given Longest Path Tree
   2.1 Fanning outward on the tree from node $i$, find $\alpha_i$ and $\beta_i$ for each down-tree node (eq. 22)
   2.2 For each arc not in the tree, find $y'_a$ and $Z'_a$ (eq. 23)
   2.3 Find the non-tree arc that maximizes $r_a = (-y'_a/Z'_a)$, excluding arcs for which $y'_a \geq 0$. (Break ties arbitrarily.) Designate it as arc $p$, the pivot arc. (If there is no non-tree arc for which $y'_a < 0$, the problem is ill-formed; $c = 0$ is feasible and therefore optimal.)
   2.4 Let $c = r_p$

3. Test and Pivot
   3.1 If adding $p$ to the longest path tree creates a directed cycle, STOP; the minimal $c$ has been found
   3.2 Otherwise, add $p$ to the tree. Let $i =$ initial node of arc $p$. Remove from the tree the arc that has the same terminal node as $p$, and go to Step 2
Numerical Example:

Consider a four-leg, 12-stream intersection whose streams are sequenced in a standard dual ring (plus pedestrian phases) as shown in Figure 12. Parameters are given in Table 1. To achieve pedestrian’s safety, pedestrian’s phase green time may start earlier than parallel vehicular right turns to force turning cars yield on crossing pedestrians. This earlier green start for pedestrian is called “Leading Pedestrian Interval”. In this example, each crosswalk has 4s LPI against the compatible parallel right turns. For a $c_{INC} = 100$ s, the Initial longest path tree is calculated as shown in Figure 12c. In Figure 12c, the values above each node are stream starting time, based on the initial cycle length. Figures 12d to 12g shows several iterations needed to find the optimal augmented tree solution. Finding $r_a = - y_a'Z_a'$ and comparing feasible ratios, an arc with the greatest ratio will be the pivot arc. In this example, for each iteration, the entering arc is named “p” and exiting arc is called “x”. The augmented tree solution is found after three iterations, with a directed circuit of 5-P6-3-P4-4-5 which is the critical circuit.

Table 1: Network Simplex Algorithm Example

<table>
<thead>
<tr>
<th>Stream</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>P2</th>
<th>P4</th>
<th>P6</th>
<th>P8</th>
</tr>
</thead>
<tbody>
<tr>
<td>v/s</td>
<td>0.10</td>
<td>0.16</td>
<td>0.10</td>
<td>0.30</td>
<td>0.10</td>
<td>0.17</td>
<td>0.10</td>
<td>0.28</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>fixed</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>22</td>
<td>22</td>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>L</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LPI</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

The minimum cycle length is equal to the greatest ratio $r_a$ in that final iteration which is 76 s. This cycle length can also be found by summing lost times and v/s ratios over the critical circuit, as shown below:

Table 2: Numerical Example Critical Circuit

<table>
<thead>
<tr>
<th>Stream</th>
<th>5</th>
<th>P6</th>
<th>3</th>
<th>P4 (LPI only)</th>
<th>4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>v/s</td>
<td>0.10</td>
<td>0.00</td>
<td>0.10</td>
<td>0.00</td>
<td>0.30</td>
<td>0.50</td>
</tr>
<tr>
<td>Lost time</td>
<td>4</td>
<td>22</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>38</td>
</tr>
</tbody>
</table>

$$c = \frac{\sum \text{Lost time}}{1 - \sum(v/s)} = \frac{38}{1 - 0.5} = 76 \text{ s}$$
a. Layout

b. Stream Sequence,
Lost Time (y) and v/s Ratio (z) in Format (y, z).
Pedestrian Phases have 4 s leading start.

Figure 12: Numerical Example for a Network Simplex Algorithm

c. Initial Longest Path Tree, c = 100 s
Labels indicate (y, z) = u

d. First Iteration, c = 87.5 s
Arc p enters, arc x exits.

e. Second Iteration, c = 87.5 s
Arc p enters, arc x exits.

f. Third and Final Iteration, c = 76 s
Arc p enters, creates directed cycle.
g. Ring Diagram highlighting Critical Circuit

Figure 12: Numerical Example for a Network Simplex Algorithm
Chapter 6: Searching Over the Sequence Array

It is the integer array $F$ that makes the sequencing problem NP-hard. The number of possible values of $F$ is $2^M$, where $M$ is the number of conflicts. (In this section and in the remainder of the paper, a “conflict” can be either an incompatibility or a staggered start relationship. The term “conflict group”, however, remains equivalent to an incompatibility clique; it does not include streams with a staggered start relationship.)

However, the structure of the problem makes it possible to develop an efficient search procedure following the branch and bound approach. The main two elements of this approach are a rapidly solvable relaxation of the integer constraints, and an enumeration procedure based on conflict groups.

6.1 Searching Using the Conflict Group Permutation Array

To make the search for the optimal sequence array $F$ more efficient, our search procedure specifies $F$ implicitly by specifying conflict group sequences. During initialization, all (maximal) conflict groups are identified; let $G =$ number of conflict groups. If conflict group $g$ contains $n$ members, it contains within it $n \times (n - 1)/2$ pairwise conflicts whose sequence must be specified. However, within a conflict group, pairwise sequence relationships must obey the logic that if $A$ precedes $B$ ($F_{AB} = 0$) and $B$ precedes $C$ ($F_{BC} = 0$), $A$ must precede $C$ ($F_{AC} = 0$). (By way of example, if $F_{AC}$ were set to 1 instead, implying that $A$ follows $C$ without going to the next cycle, there would be logical deadlock. It would be detected in network analysis because circuit $A$-$B$-$C$-$A$ would have a positive length regardless of the value of $c$, because in this circuit all of the $F$ values are 0.) Thus, the pairwise variables $F_{ij}$ among the streams in a conflict group are not all independent. In the example just given of a 3-member conflict group, if pairwise $F$'s are set for two pairs, the $F$ for the third pair is also set. In a 4-member conflict group, once 3 pairwise sequence variables are fixed, the full set of 6 pairs contained in the conflict group will be fixed.

Therefore, searching over conflict groups rather than directly over the $F$ array saves the effort of analyzing “deadlock” infeasible solutions. While the number of possible values for the subarray of $F_{ij}$ variables contained in an $n$-member conflict group is $2^{n(n-1)/2}$, the number of permutations (ways of sequencing the streams in the conflict group) is only $n!$. For example, if $n = 4$, there are 6 pairwise conflicts contained in the conflict group which admit $2^6 = 64$ possible values for the $F_{ij}$ subarray, while there are only $6! = 24$ possible ways of sequencing the four streams; the other 40 sets of values are infeasible due to deadlock.

For conflict group $g$, we index its permutations with the variable $q_g = 1, \ldots, Q_g$ (where $Q_g = n!$, as mentioned before.) By definition, $q_g = 0$ indicates that no permutation has been specified for
conflict group \( g \). The array of conflict group permutations \( \{q_1, q_2, \ldots, q_G\} \) implicitly specifies all of the pairwise sequence variables, and is therefore used to search the solution space for \( F \). That is, nodes in the branch and bound tree are conflict groups, and branches emanating from the nodes are the possible permutations of that conflict group. In this manner, the branch and bound procedure enumerates (implicitly, when possible) every sequence of every conflict group in combination with every possible sequence of every other conflict group. (See Figure 13)

*Figure 13: Searching Using the Conflict Group Permutation Array*
6.2 Other Devices to Limit the Solution Space

Because many conflict groups share the same member streams, some permutations of a group are likely to be inconsistent with the permutations of other groups. When a new permutation of a conflict group is encountered in the branch and bound tree, the first check is to see whether any of the pairwise sequence variables it implies are inconsistent with those specified at earlier nodes in the tree; if so, that permutation is recognized as infeasible and pruned from the tree.

Conflict groups encountered late in the tree may have all of their pairwise sequence variables already specified by conflict groups appearing earlier in the tree. Such conflict groups are eliminated from the search list, reducing the effective value of $G$, the number of conflict groups.

Bounding is most likely to be successful when the sequence for a few of the longer conflict groups are specified, for it is the longer conflict groups that usually require the greatest cycle length, and poor coordination between two conflict groups can lead to grossly long cycle. To encourage bounding as early as possible, conflict groups are examined in order of decreasing length.

Two other devices are used to reduce the size of the solution space. First, in keeping with equation 2 and 3, one stream $h$ is arbitrarily chosen to start the cycle, fixing the $F$'s of all arcs incident to node $h$. By selecting as node $h$ the node representing the stream with the largest number of conflicts, the problem size is reduced the most. Any conflict group containing stream $h$ will then have only $(n-1)!$ permutations.

Second, if two streams $A$ and $B$ have been specified to have simultaneous start, they share a common node representing the start of their green. If both are in conflict with a third stream $C$, they must both have the same precedence relationship with $j$; that is, we require that $F_{AC} = F_{BC}$.


6.3  **Bounding Through Relaxation of the Reciprocity Constraint**

When evaluating a given branch in the search tree emanating from node g (i.e., conflict group g), the array of conflict group permutations will be specified as \{q_1, q_2, ..., q_g, 0, 0, ..., 0\}. The specified permutations of the first g conflict groups fix values for many of the pairwise sequence variable \(F_{ij}\). Other \(F_{ij}\)’s may be fixed because they relate to a start stream or a simultaneous start stream.

For i-j pairs not yet fixed, we relax the reciprocity constraint (equation 1), setting \(F_{ij} = F_{ji} = 1\) for such pairs. This relaxation makes the cycle minimization problem a linear optimization problem (program 12) that can be solved very efficiently with the network simplex algorithm described earlier. It remains only to be proven that this device is indeed a relaxation.

**Proof:** Consider two values of the \(F\) array, \(F_D\) and \(F_R\). \(F_R\) is a relaxation of \(F_D\) if, for every i-j pair in \(F_D\) for which \(F_{ji} = 1\), \(F_{ji} = 1\) in \(F_R\) as well; \(F_D\) is called a descendent of \(F_R\). Letting \(c^*(F)\) denote the minimum feasible cycle length for a given array \(F\), i.e.,

\[
c^*(F) = \max_k \left[ \text{cmin}_k(F) \right]
\]

It is clear from examining relation 16 and 17 that, for every circuit \(k\) in the directed incompatibility graph,

\[
\text{clb}_k(F_R) \leq \text{clb}_k(F_D) \quad \text{for} \quad k \in \text{KL}, \quad \text{and}
\]

\[
\text{cub}_k(F_R) \geq \text{cub}_k(F_D) \quad \text{for} \quad k \in \text{KU}
\]

therefore,

\[
c^*(F_R) \leq c^*(F_D) \quad (25)
\]

implying that \(F_R\) is indeed a relaxation of \(F_D\). Therefore, if cycle length minimization with \(F_R\) reveals that \(c^*(F_R) > c_{\text{INC}}\), all solutions that are descendents of \(F_R\) are bounded.

**Q.E.D.**
It is also evident that this relaxation puts a lower bound on every $u_i$, i.e.,

$$u_i(F_R) \leq u_i(F_D) \quad (26)$$

because non-zero elements of $F$ only make arc lengths shorter, and every $F_{ij}$ in $F_R$ is at least as great as the corresponding $F_{ij}$ in $F_D$. Therefore if $u_i \geq c$ for a relaxation (meaning a non-canonical rotation has been found), it can be concluded that its descendents will not be canonical rotations, and can therefore be bounded. (However, there is no similar bound for non-canonical rotations when $u_i<0$).

The advantage of relaxing the reciprocity constraint instead of the 0/1 integer constraint is that the latter relaxation does not result in a linear optimization problem, because the objective function would then include the product of two decision variables, $cF_{ij}$.

### 6.4 Summary of the Branch and Bound Algorithm

The branch and bound algorithm is summarized in Figure 14. That algorithm is written to find the sequence that minimizes cycle length; if there is a tie, the first optimal sequence encountered will emerge as the solution. With a slight change, the algorithm is readily modified to find every sequence that yields a cycle length below some fixed value $c_o$. Because other objectives besides minimizing cycle length can enter into the sequencing decision, it is helpful to provide the traffic engineer with not only the optimal solution, but a list of “good” solutions which can later be examined in more detail.
Figure 14: Branch and Bound Algorithm

Algorithm to find the stream sequence requiring the minimum cycle length.

1. Initialize Search Order and Variables
   1.1 List conflict groups and order them in decreasing size
   1.2 Initialize the incumbent \( c_{\text{INC}} \). This may be done either by having the user simply enter a value known (or believed) to be feasible, or by having the user specify an initial value of \( F \) and applying the circuit analysis, setting \( c_{\text{INC}} \) to the resulting minimum cycle length.
   1.3 Set \( F_{ij} = F_{ji} = 1 \) for every conflicting stream pair \( i-j \)
   1.4 Choose a start stream (or set of start streams with simultaneous start), giving preference to the stream (or simultaneous start stream set) with the largest number of conflicts. If possible, let it also be a member of the longest conflict group.
   1.5 Call the start stream \( h \). For every conflicting stream pair \( h-j \), put arcs \( h-j \) and \( j-h \) into set \( CA_o \), the initial set of arcs whose \( F_{ij} \) values are fixed, and set \( F_{hj} = 0 \). (If there is a set of initial start streams, apply this step to each start stream, calling each start stream in turn \( h \).)
   1.6 If a stream \( i \) incident to an arc \( i-j \) belonging to \( CA_o \) has a simultaneous start relationship with another stream \( k \) and streams \( j \) and \( k \) are in conflict, set \( F_{ki} = F_{ij} \), set \( F_{ik} = 1 - F_{ki} \), and add arcs \( i-k \) and \( k-i \) to \( CA_o \)

2. Establish Correspondence Between Conflict group Order and Order in Which Pairwise Sequence Variables Are Set
   2.1 Let \( G \) be the number of conflict groups, and \( g \) the conflict group index (corresponding to the list initially created in Step 1.1). Loop, for \( g = 1 \) to \( G \):
   2.1.1 Let \( A_g \) be the set of all arcs \( a(i,j) \) for which streams \( i \) and \( j \) belong to conflict group \( g \), excluding arcs already in \( CA_{g-1} \). \( A_g \) represents the set of arcs whose \( F_{ij} \) will be set when conflict group \( g \) is processed
2.1.2. If a stream i incident to an arc i-j that belongs to $A_g$ has a simultaneous start relationship with another stream k and streams j and k are in conflict, add stream pairs k-j and j-k to $A_g$, unless they already belong to $A_g$.

2.1.3. If for any conflict group g, $A_g$ is an empty set, remove g from the conflict group list. Reduce G and update the conflict group indices accordingly.

2.1.4. Let $CA_g = CA_{g-1} \cup A_g$. $CA_g$ represents the cumulative set of arcs whose F values are specified by conflict groups 1, ..., g.

2.2. For every conflict group, determine the number of permutations $Q_g$. If a start stream is a member of the group, $Q_g = (n_g - 1)!$; otherwise, $Q_g = n_g!$, where $n_g =$ size of conflict group g. The permutations of conflict group g will be indexed by $q_g = 1, \ldots, Q_g$. Permutations have a natural “lexicographic” order, so that a particular permutation index is unambiguous.

3. Initialize Branch and Bound Tree

3.1. Set g = 1

3.2. Set $q_g = 1$

4. Evaluate and Test

4.1. For all arcs a(i,j) for which streams i and j belong to conflict group g, generate a set of trial pairwise sequence values $TF_{ij}$ that correspond to the current permutation $q_g$. If any such arc belongs to $CA_{g-1}$ and $TF_{ij} \neq F_{ij}$, this permutation is inconsistent with the predecessor branches and the current branch is pruned; go to Step 6.

4.2. For all arcs in $A_g$, set $F_{ij} = TF_{ij}$

4.3. Perform network optimization (see earlier algorithm). Let $c^* =$ minimum cycle length returned from the network analysis.

4.4. If $c^* \geq c_{INC}$, the current branch is pruned (suboptimal); go to Step 6.
4.5 If \( u_i \geq c \) for any node \( i \), the current branch is pruned (not a canonical rotation); go to Step 6

4.6 If \( g = G \) the current solution has been completely specified

4.6.1 If \( u_i < 0 \) for any node \( i \), the current branch is pruned (not a canonical rotation); go to Step 6

4.6.2 Otherwise, a new solution has been found; set \( c_{INC} = c \) and go to Step 6

5. **Advance to Next Conflict Group**

5.1 Set \( g = g + 1 \)

5.2 Set \( q_g = 1 \) and go to Step 4

6. **Advance to Next Permutation Within a Conflict Group**

6.1 Set \( q_g = q_g + 1 \)

6.2 If \( q_g \leq Q_g \), go to Step 4. Otherwise, continue; conflict group \( g \) has been exhausted

7. **Retreat to Previous Conflict Group; Test for Termination**

7.1 For all arcs \( a(i,j) \) belonging to \( A_g \), set \( F_{ij} = 1 \)

7.2 Set \( g = g - 1 \)

7.3 If \( g = 0 \), STOP; entire search tree has been exhausted. Otherwise, go to Step 6
Chapter 7    Model Extensions and Limitations

7.1    Finding the Sequence with the Smallest Webster Cycle Length

In recognition of randomness in arrival and departure processes, cycle length is rarely set to its minimum value with 100% saturation (i.e., \( f_i = 1 \) for all \( i \)). One common approach is to specify a smaller degree of saturation, say, 90%. Another is to use Webster’s well-known cycle length formula [15],

\[
c_W = \frac{1.5L + 5}{1 - \sum (v/s)}
\]

where \( c_W \) is the Webster cycle length, \( L \) is the lost time summed over the critical conflict group and the \( (v/s) \) sum is likewise over members of the critical conflict group. Compared to the minimum cycle length formula, Webster expands lost times by 50% and adds 5 more seconds to the lost time to give the cycle a buffer against overflow in case of demand fluctuations, resulting in a cycle length that, based on experience, performs well.

The Webster cycle length for an intersection depends on the way the streams are sequenced, if sequencing can affect lost time. Of the many possible stream sequences, it may be desirable to find the one with the minimum Webster cycle time, as this sequence is likely to perform best under the fluctuating demand conditions for which Webster’s formula was derived. We have generalized Webster’s formula to account for situations in which the cycle length is determined by any critical circuit in the incompatibility graph (not just a critical conflict group). Our generalization also treats lost time for normal vehicular streams, whose needed green time from cycle to cycle varies with demand and vehicle headways, differently from that of pedestrian and similar streams whose needed green time does not vary with demand. The generalized Webster cycle length for a given circuit \( k \) in the directed incompatibility graph is given by

\[
c_{W,k} = \frac{1.5L_V + L_p + 5 \sum F_{ij}}{\sum F_{ij} - \sum (v/s)_i}
\]

(27)
where,

\[ c_{W,k} = \text{generalized Webster cycle length for circuit } k \]
\[ L_v = \text{sum of the lost time on normal vehicular streams and clearance times, summed over circuit } k \]
\[ L_p = \text{sum of the lost time on pedestrian streams and other streams whose needed green time does not depend on demand, summed over circuit } k \]

and the sums in the formula are over arcs (for the double subscripted variables) or streams in circuit \( k \). Because a circuit in the directed incompatibility graph can cover more than one cycle – in fact, it covers \( \sum F_{ij} \) cycles – the 1 in the denominator of Webster’s formula is replaced with \( \sum F_{ij} \), and the 5 s additional lost time per cycle is also multiplied by the same quantity.

To ensure that every sequence of streams in the signal control program will have enough time to handle random fluctuations (i.e., will have the needed Webster cycle length), a variation of constraint 12 must be satisfied:

\[
\sum y_a + 5 \sum F_a + c_{W,k} \sum z_a - c_{W,k} \sum F_a \leq 0 \quad \text{for every circuit } k \quad (28)
\]

in which lost times included in the \( y_a \) terms have been inflated by 50%, as appropriate. Therefore the design problem of finding the stream sequence that results in the minimum Webster cycle length has the minimize objective

\[
\min c_W = \max_k [c_{W,k}(F)] \quad \text{with respect to } F \quad (29)
\]

The resulting optimization program is identical to program 17, and can therefore be solved by the same algorithm, by simply inflating lost times by 50% as appropriate, setting \( f_i \) to 1 for all streams, and replacing equation 14 with

\[
c_{W,k} = \frac{\sum_{a \in k} y_a + 5 \sum_{a \in k} F_a}{\sum_{a \in k} F_a - \sum_{a \in k} z_a} \quad (30)
\]
In the network analysis (Step 4.3 of the algorithm, Figure 14), $c_{\text{min}_k}$ is replaced with $c_{W,k}$. Because $c_{W,k}$ is calculated just as $c_{\text{min}_k}$ using accumulated values of $y_a$, $z_a$, and $F_a$, no change in the network analysis algorithm is needed except the application of equation 22 in place of equation 14. Because equation 23 is a generalization of equation 14 by adding a second term to the numerator, we use it in place of equation 14 in software, allowing the user to choose the multiplier (i.e., 5 for Webster cycle length, 0 for minimum cycle length) of that term.

As an example, the 4 – legged intersection shown in Figure 1, can be applied for Webster’s calculation. Simply minimizing cycle length leads to a solution with $c_{\text{min}} = 41.8$ s, as shown in chapter 9, example 2. With the objective function modified, the Webster cycle length is found to be 64.4 s, as shown in Figure 15.

\[ \text{Cycle} = 64.4 \text{ s} \]

*Figure 15: 4 - legged intersection - Webster's Formula Solution*
The solution using Webster’s formula has the same augmented tree as the regular calculation, shown in Figure 16.

The last arc added to the augmented tree (arc 8-3) gets the 5 seconds extra time that Webster’s formula considers for each cycle. That’s why the cycle length is 64.4 s, even though the last critical stream ends at time 59.4 s. For application, those extra 5 s should not all be concentrated between streams 8 and 3, but should instead be distributed across the critical streams.

*Figure 16: 4-legged intersection
Augmented Tree*
7.2 Minimum Green and Maximum Red Constraints

Our model explicitly accounts for minimum green time constraints (in seconds) on pedestrian streams and other streams whose needed green time does not depend on demand. Unless a stream’s v/s ratio is very small, no such constraint is usually needed for streams whose green time depends on demand. Likewise, maximum red time constraints (which may be related to a maximum desired queue length, for example) are not usually necessary, either, because minimizing cycle length (as well as maximizing reserve capacity) will usually result in the shortest red times possible.

Nevertheless, constraints of this sort can be added. We will sketch out how the network formulation can be extended to accommodate such constraints. Suppose stream i is constrained such that its green time may not exceed \( G_{\text{min}}^i \), and its red time may not exceed \( R_{\text{max}}^i \). The maximum red time constraint can be transformed into a minimum green time constraint, with a minimum green time of \( (c - R_{\text{max}}^i) \). Each supplementary minimum green time constraint on stream i leads to an additional constraint identical to constraint 4, but with the minimum green time embodied in constraint 4, \( (L_i + f_i c(v/s)_i) \), replaced with the supplementary minimum green time (i.e., \( F_{\text{min}}^i \) or \( c - R_{\text{max}}^i \)). On the directed incompatibility graph, this means adding, for each supplementary minimum green time constraint on stream i, an additional arc a going from stream i to every stream j that is incompatible with i. For a minimum green constraint, the new arc’s length components are \( z_a = 0 \), \( y_a = G_{\text{min}}^i + L_{ij} \), and \( F_a = F_{ij} \). For a maximum red constraint, the length components are \( z_a = 1 \), \( y_a = -R_{\text{max}}^i + L_{ij} \), and \( F_a = F_{ij} \). This can lead to up to three arcs going from stream i to stream j, only one of which (at most) will be binding in any solution.

It is also possible to add nodes representing the start of a stream’s red (the moment \( v_i \)). Let node \( iG \) indicate stream i’s green start, and node \( iR \) stream i’s red start. The three types of minimum green time constraints would then be represented by arcs going from node \( iG \) to \( iR \), with lengths \( (L_i + f_i c(v/s)_i) \), \( G_{\text{min}}^i \), and \( c - R_{\text{max}}^i \). Clearance and non-overlap between incompatible streams i and j would then be represented by an arc from \( iR \) to \( jG \) with length \( (L_{ij} + cF_{ij}) \). Staggered start constraints remain as previously formulated, connecting green start nodes of the affected streams. The addition of nodes representing red starts nodes makes the network in some ways clearer, and can reduce the network size if there are a lot of minimum green and maximum red constraints. It also facilitates modeling any other constraints or control tactics that directly
deal with the start red event. Otherwise, the nature of the problem and its solution remains unchanged.

7.3 Limitations

The same formulation can be applied if there are stepwise varying saturation flow rates, as well at networks of several intersections (e.g., a two-way arterial, or a network of intersections in a downtown), connected by staggered start constraints that aim to provide progression to selected movements that cross several stoplines. The methods presented here will find the sequencing and timing that achieve the desired progression with the minimum cycle length, if a feasible solution exists. However, a large number of desirable staggered start constraints often makes such a problem infeasible, and the objective becomes something like “provide the best progression possible”, formulated, e.g., as maximizing bandwidth or minimizing delay; such an extension is beyond the scope of this study. In addition, our methodology has assumed that the offsets specified for staggered start constraints ($S_{\text{min}}$ and $S_{\text{max}}$) are less than $c$, as they normally are in what is generally considered an intersection problem, but they may not be for a problem over a wider area. If either $S_{\text{min}}$ or $S_{\text{max}}$ exceeds $c$, there may be no canonical rotation as we have defined it. Either a broader definition of a canonical rotation, or a modification of the algorithm to admit non-canonical rotations, will be needed.
Chapter 8: Program

The program which is used in this study was written by Professor P.G. Furth. In this chapter a brief introduction to program and its usage will be discussed.

The program is written in C and uses the described algorithms (see chapter 6) to find the optimal sequence in general intersection. Staggered start, Simultaneous start, unusual geometry, asymmetric clearance time, are features included in this program.

To start the program, you need to make a text input file based on your model. To create an input file, you need to:

1. Make a conflict matrix; its value for each (i,j) pair is: 1 for every i and j pair in conflict and gets no value for compatible pairs
2. Create a table with node names in first row, node indices in row 2, each stream’s fixed time in tenths of a second in row 3, each stream’s lost time in tenths of a second in row 4, stream’s volume in row 5, saturated flow rate in row 6, calculated v/s in row 7, streams names in row 8 and streams indices in row 9. In rows 5 and 6, volume and saturation flow rate must have common units.
3. Create a clearance matrix is similar to conflict matrix. Instead of conflict values, each cell (i,j) gets needed clearance time in tenths of a second for streams i to clear the intersection for stream j. As described earlier (see chapter 2) clearance time for (i,j) and (j,i) can get different values.
4. Create a list of conflict pairs
5. Call nodes indices and pair clearance time for each conflict pair
6. There are two types of conflict pair:
   - Physically in conflict streams, such as (2,9) in Figure 2
   - Staggered start conflict streams
   For each type there is a code type, which values are: 0 = physically in conflict and 1=staggered start conflict
7. If you want to define an initial sequence to start with, create a sequence matrix with values of:

\[
F_{ij} = \begin{cases} 
1 & \text{If stream } i \text{ follows stream } j \text{ in the cycle} \\
0 & \text{If stream } j \text{ follows stream } i \text{ in the cycle} \\
-999 & \text{for compatible pairs}
\end{cases}
\]
**Input file:**

- First 2 rows are used for the input file title and a blank line to separate it from later data
- Next comes a table created in step 2 (i.e., see Table 3)
- Blank line to separate input data from next information
- Clearance times matrix as shown on Table 4
- Another blank line
- In two rows, joint names and indices are repeated
- Blank line
- ARC List title
- Arc list header
- Arc list table as created in Step 4-6 (see Table 5)
- 3 line including # after arc list table, blank line and initial sequence title line
- Initial sequence matrix as shown in Table 6 (All tables in this part will be discussed later with the example.)

**Output files:**

There are four different output files for each run: CGECHO, ECHO, DUMMY and solution file.

**CGECHO:**

It includes conflict matrix (conM), with clearance time values for conflict pairs and -999 for compatible pairs. In pair matrix, searched conflict pairs get indices and rest will have -1 value. Due to logical proposition that if (i,j) is in conflict, (j,i) is in conflict, only the upper triangle will get values other than -1.

Next comes CG pairs and a parameter called sub which indicate that mentioned pair is a subgroup of a greater conflict group or not; 1 for being part of greater conflict group, 0 otherwise.

Conflict groups are created by increasing offsets until it can’t make any greater conflict group.

The last part of CGECHO is a list of all maximal conflict groups of input file.
**ECHO:**

Data read from input file are written in ECHO. In this file, the program writes all conflict groups and shows the progress of the search for minimum c for any given F. A record is written for each iteration as the program searches for a better F array to reach the optimal or best possible sequence for the problem. In this record, all possible permutations, including infeasible permutations, will appear.

**DUMMY:**

Contains node names presenting each stream of the intersection

**Solution File: (.Sol)**

Solution file contains program final data. An input file name used in program is written after solution file title. Next will be streams and nodes that presenting them.

In next part, maximal conflict groups that appear in the ECHO file are shown in order of tree-building.

User chosen parameters will appear next and based on input data the next table contains arc numbers (a), node numbers (nd), starting node for each arc (i), ending node for each arc (j), y_a and z_a for each presented arc.

The next output are the searched sequences for the optimal or sequences better than initial sequence. It shows streams in order to make timing plan diagram, minimum calculated cycle length, F_a for each arc, preceding node for each listed node (PRE), arc lengths (negative value for arcs with F_a=1), U_i; each node start of green time (0 for h stream; there would be more than one 0 if simultaneous start is defined), Reduced Length; which are non negative values with 0 presenting arcs on a longest path tree and the last parameter is the length of arcs ending between each node and its preceding node (PLn).
Chapter 9: Examples of Algorithm Performance

To illustrate the efficiency of the algorithm, three examples are used. The first one is Chapel Hill, Figure 17, described in Heydecker[4] and used in several papers such as Zuzarte Tully[7], Wong, Law and Tong[16], Improta and Cantarella[17], Heydecker and Dudgeon[8], etc. The second case is the 4-legged intersection, Figure 1, described earlier. It has a combination of vehicular streams, Pedestrian and trolley stream. The third example is a local complicated intersection, Charles Circle at Boston, MA.

Example 1: Chapel Hill, Huddersfield, England

This example is a 4-legged intersection with an unusual geometry layout (Figure 17). The intersection is Chapel Hill (Huddersfield, England). Vehicular arrival and saturation departure rates for each of the streams are given in Table 7. Each of the vehicular streams has a minimum running time of 8 seconds. There are two approaches to a main route from the same side. Because of the layout of the junction, the clearance time matrix, Table 7, is asymmetric. The maximum degree of saturation was taken to be 0.90 for each stream. In this example, English driving rule is applied (vehicles drive on the left side of street.) All inputs are exactly the same as in Heydecker and Dudgeon[8].
Data for the example are shown on Table 7. Table 8 presents defined clearance time based on geometric limits. The clearance matrix also shows the conflict matrix.

**Table 7: Chapel Hill input data**

<table>
<thead>
<tr>
<th>stream</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>fixed</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>130</td>
</tr>
<tr>
<td>lost</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>v</td>
<td>123</td>
<td>260</td>
<td>250</td>
<td>633</td>
<td>871</td>
<td>722</td>
<td>925</td>
<td>655</td>
<td>0</td>
</tr>
<tr>
<td>s</td>
<td>3763</td>
<td>3997</td>
<td>2622</td>
<td>3494</td>
<td>2978</td>
<td>1835</td>
<td>3360</td>
<td>2965</td>
<td>1</td>
</tr>
<tr>
<td>v/s</td>
<td>0.03</td>
<td>0.07</td>
<td>0.10</td>
<td>0.18</td>
<td>0.29</td>
<td>0.39</td>
<td>0.28</td>
<td>0.22</td>
<td>0.00</td>
</tr>
<tr>
<td>joint name</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>Jt_index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

**Table 8: Chapel Hill - Clearance Times Matrix (in tenths of a second)**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>60</td>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>50</td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>70</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>70</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>90</td>
</tr>
<tr>
<td>9</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
<td>80</td>
</tr>
</tbody>
</table>

In Heydecker and Dudgeon [8], the critical time of 44 seconds is considered as a minimum cycle time for this example. Calculations using the program in this thesis resulted in the minimum cycle length of 38.5 seconds, with a timing plan as shown in Figure 18.
Figure 18: Chapel Hill, Optimal Solution Timing Plan

Cycle ≈ 38.5 s

Figure 19 shows the incompatibility graph for the example and Figure 20 presents the augmented tree.

Figure 19: Chapel Hill, Directed Incompatibility Graph
Comparing results, these comments can be offered:

1) Our study finds a better solution with lower cycle length. The previous research best solution was 44 seconds, which is greater than our solution, 38.5 seconds.

2) Heydecker and Dudgeon [8], assume that a sequence of maximum compatible groups will be the optimal solution. In this study, the optimal solution is a sequence of maximum compatible groups; see Figure 21.

3) However, their algorithm fails to find this solution, because it doesn’t use exhaustive enumeration. Their enumeration method admits only sequences that meet criteria related to overlaps.

4) In addition, their solution is based on a critical cycle (1-8-4-9-1) in which the streams are not mutually conflicting (streams 1 and 9 are not in conflict)
Example 2: 4-legged Intersection With 6 Vehicular Streams, 2 Crosswalks and a Trolley

The second example is a regular 4-legged intersection with a great number of streams; 6 vehicular streams, 2 pedestrian streams and a trolley stream.

Input data for this example is given in Table 3 and its clearance time is presented on Table 4 as described earlier. For all vehicular streams 3 seconds lost time is considered and all conflict pairs are incompatible arcs.

The intersection layout with all its streams is shown in Figure 1, and the incompatibility graph is presented in Figure 4. Figure 16 shows an augmented tree for this example and the final timing plan according to program solution is available in Figure 22.
Table 3: 4-legged intersection input data

<table>
<thead>
<tr>
<th>stream</th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>8</th>
<th>9</th>
<th>11</th>
<th>35</th>
<th>36</th>
<th>46</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>fixed</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>100</td>
<td>100</td>
<td>70</td>
</tr>
<tr>
<td>lost</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>v</td>
<td>300</td>
<td>200</td>
<td>500</td>
<td>400</td>
<td>300</td>
<td>400</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s</td>
<td>1800</td>
<td>1700</td>
<td>1800</td>
<td>1700</td>
<td>1800</td>
<td>1700</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>v/s</td>
<td>0.17</td>
<td>0.12</td>
<td>0.28</td>
<td>0.24</td>
<td>0.17</td>
<td>0.24</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>joint name</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>9</td>
<td>11</td>
<td>35</td>
<td>36</td>
<td>46</td>
</tr>
<tr>
<td>Jt index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

Table 4: 4-legged intersection - Clearance Times Matrix (in tenths of a second)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>5</th>
<th>10</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>20</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>20</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td>10</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
</tr>
</tbody>
</table>
Table 5 and 6 shows an input format of conflict arcs and sequence matrix.

Figure 22: 4 - legged intersection (optimal solution timing plan)

Cycle = 41.8 s
### Table 5: 4-legged intersection Arc List

<table>
<thead>
<tr>
<th>index</th>
<th>i name</th>
<th>i index</th>
<th>j name</th>
<th>j index</th>
<th>type</th>
<th>type code</th>
<th>joint i</th>
<th>index</th>
<th>joint j</th>
<th>index</th>
<th>clearance</th>
<th>stagger</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>con</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>con</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>con</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>36</td>
<td>7</td>
<td>con</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>36</td>
<td>7</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>con</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>con</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>3</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>5</td>
<td>con</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>46</td>
<td>8</td>
<td>con</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>46</td>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>con</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>con</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>con</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>con</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>con</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>con</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>8</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>con</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>8</td>
<td>3</td>
<td>35</td>
<td>6</td>
<td>con</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>35</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>8</td>
<td>3</td>
<td>46</td>
<td>8</td>
<td>con</td>
<td>0</td>
<td>8</td>
<td>3</td>
<td>46</td>
<td>8</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>con</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>con</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>9</td>
<td>4</td>
<td>11</td>
<td>5</td>
<td>con</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>11</td>
<td>5</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>9</td>
<td>4</td>
<td>35</td>
<td>6</td>
<td>con</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>35</td>
<td>6</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>9</td>
<td>4</td>
<td>46</td>
<td>8</td>
<td>con</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>46</td>
<td>8</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>11</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>con</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>11</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>con</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>11</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>con</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>11</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>con</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>9</td>
<td>4</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>11</td>
<td>5</td>
<td>46</td>
<td>8</td>
<td>con</td>
<td>0</td>
<td>11</td>
<td>5</td>
<td>46</td>
<td>8</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>35</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>con</td>
<td>0</td>
<td>35</td>
<td>6</td>
<td>8</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>35</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>con</td>
<td>0</td>
<td>35</td>
<td>6</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>36</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>con</td>
<td>0</td>
<td>36</td>
<td>7</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>46</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>con</td>
<td>0</td>
<td>46</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>46</td>
<td>8</td>
<td>8</td>
<td>3</td>
<td>con</td>
<td>0</td>
<td>46</td>
<td>8</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>46</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>con</td>
<td>0</td>
<td>46</td>
<td>8</td>
<td>9</td>
<td>4</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>46</td>
<td>8</td>
<td>11</td>
<td>5</td>
<td>con</td>
<td>0</td>
<td>46</td>
<td>8</td>
<td>11</td>
<td>5</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

### Table 6: 4-legged initial sequence matrix

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>3</th>
<th>5</th>
<th>8</th>
<th>9</th>
<th>11</th>
<th>35</th>
<th>36</th>
<th>46</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>-999</td>
<td>1</td>
<td>-999</td>
<td>1</td>
<td>1</td>
<td>-999</td>
<td>1</td>
<td>-999</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>-999</td>
<td>1</td>
<td>-999</td>
<td>-999</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>-999</td>
<td>-999</td>
<td>-999</td>
<td>-999</td>
<td>-999</td>
<td>-999</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>-999</td>
<td>1</td>
<td>1</td>
<td>-999</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>-999</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>-999</td>
<td>-999</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td>-999</td>
<td>-999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td>-999</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>46</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-999</td>
</tr>
</tbody>
</table>
Example 3: Charles Circle, Boston, MA

The third example is a local case with a large number of streams and unusual geometry. There is a subway station at this intersection and it is a crowded place for both vehicular and pedestrian modes.

In this intersection 2 different situations were studied:

Case 1:

The current situation of Charles Circle expect that for the Cambridge Street WB approach, only one lane goes through and two lanes go right, as shown in Figure 23. Right turns from stream 1 and 2 are permitted to conflict with crosswalk 33.

Traffic on Storrow drive – WB (off-ramp) turning on to the Longfellow Bridge have no traffic signal; they are under yield control, and are omitted from the analysis.

An island divides streams 1 and 2 going through to Cambridge Street. Because of right turns from stream 1 and left turns from stream 2, streams 1 and 2 are conflicting.
In the current case, streams 3 and 4 run at the same time. In this model, stream 3 and 4 are separated, with a single lane feeding Longfellow Bridge (WB). Stream 96 is compatible with stream 2, but to evacuate the standing queue at stream 96’s stop line, it is forced by staggered start to start 12 s earlier than stream 2. This will create space for stream 2 traffic flow. Treating stream 5 in conflict with streams 1 and 2, forces stream 95 to stay green until 5 ends its green time, preventing spillback due small storage space for stream 95’s. This space is needed for the few left turners from stream 3 during their green time. (This constraint could have been done by expanding the model to include simultaneous or offset stop constraint.)

Input data are shown in Table 9 and 10. Saturation flow rates are based on the number of lanes and expected flow reductions due to turns, conflicting pedestrians, and short lanes. Lost times for crosswalks are based on a walking speed of 3.5 ft/s. Volumes are based on counts done by a consultant, shown in Figure 24. Clearance times are based on distances to conflict points.
Based on the degree of saturation of 0.92, the optimal solution found by program has cycle length of 93.8 seconds. The calculation result is close to the current cycle length running at this intersection which is about 100 s. In this example, according to site studies and intersection geometry, greater clearance times were used, such as 6 s clearance time for vehicles on stream 3 to clear the crosswalk 99.

### Table 9: Charles Circle - Case 1 - input data

<table>
<thead>
<tr>
<th>stream</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>95</th>
<th>96</th>
<th>11</th>
<th>22</th>
<th>33</th>
<th>44</th>
<th>55</th>
<th>66</th>
<th>88</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>fixed</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>lost</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>v</td>
<td>891</td>
<td>1147</td>
<td>329</td>
<td>375</td>
<td>252</td>
<td>380</td>
<td>2038</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>s</td>
<td>4500</td>
<td>3400</td>
<td>1800</td>
<td>2800</td>
<td>2700</td>
<td>3800</td>
<td>7200</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>v/s</td>
<td>0.20</td>
<td>0.34</td>
<td>0.13</td>
<td>0.15</td>
<td>0.09</td>
<td>0.10</td>
<td>0.28</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>joint name</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>95</td>
<td>96</td>
<td>11</td>
<td>22</td>
<td>33</td>
<td>44</td>
<td>55</td>
<td>66</td>
<td>88</td>
<td>99</td>
</tr>
<tr>
<td>j_index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
</tr>
</tbody>
</table>

### Table 10: Charles Circle - Case 1 - Clearance Times Matrix (in tenths of a second)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>95</th>
<th>96</th>
<th>11</th>
<th>22</th>
<th>33</th>
<th>44</th>
<th>55</th>
<th>66</th>
<th>88</th>
<th>99</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>20</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>60</td>
<td>0</td>
<td>0</td>
<td>15</td>
<td></td>
<td></td>
<td>20</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>120</td>
<td>0</td>
<td>5</td>
<td>0</td>
<td></td>
<td></td>
<td>20</td>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>95</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>96</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>55</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>66</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>88</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>99</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Solution timing plan is shown in Figure 25. Solid rectangle borders represent the necessary part of a stream’s split, while dashed borders indicate additional (slack) time. Earliest possible ending time is shown along with extended ending time (in parentheses). Figure 26 is the augmented tree for this example.
Figure 26: Charles Circle - Case 1
Augmented tree
**Case 2:**

The next example considers a few changes in geometric conditions as shown in Figure 27. By continuing the island, the left turn for stream 2 is restricted and vehicles that used to turn left from this approach should now make a U-turn on Cambridge street and join the Cambridge street–WB approach. Streams 1 and 2 are still in conflict, however, due to right turns from stream 1. Stream 0 is added to separate left turn movement on Longfellow Bridge – EB (using one lane) from through and right turn traffic (using 2 lanes, as stream 1). Crosswalk 77, not included in Case 1, is added. Stream 96 will be required to run concurrently with stream 0, making a continuous movement for left turns, eliminating the need for queuing at 96; therefore stream 96 has been reduced from 6 lanes to 2, as shown in figure 27. Stream P is added to give pedestrians a more direct way to get between the bridge and the subway station. For pedestrian and bicyclist safety, stream 1 is in conflict with crosswalk 33 and P.

*Figure 27: Charles Circle, Case 2 (Layout)*
On Cambridge Street – WB, a presignal is considered just before the U-turn. This presignal is not part of the analysis but it is needed to prevent arriving vehicles on Cambridge Street – WB from blocking vehicles on stream 2 making a U-turn.

Data for the analysis are shown in tables 11 and 12. Volumes for stream 3, 4 and 96 are modified to account for the vehicles which used to turn left from stream 2. Clearance time from stream 1’s end to stream2’s start was reduced from 1.5 s to 1.0 s because the new layout avoids stream 2’s left turn conflicts. Degree of saturation was set at 0.92 just as Case 1.

<table>
<thead>
<tr>
<th>stream index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>lost</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>170</td>
<td>140</td>
<td>170</td>
<td>140</td>
<td>140</td>
<td>170</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td>140</td>
<td></td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>199</td>
<td>692</td>
<td>1147</td>
<td>446</td>
<td>497</td>
<td>252</td>
<td>360</td>
<td>199</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>1500</td>
<td>2100</td>
<td>2100</td>
<td>1700</td>
<td>2000</td>
<td>2700</td>
<td>2900</td>
<td>2700</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>v/s</td>
<td>0.13</td>
<td>0.22</td>
<td>0.85</td>
<td>0.26</td>
<td>0.18</td>
<td>0.09</td>
<td>0.10</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>current name</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>t_index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>stream index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
<th>17</th>
</tr>
</thead>
<tbody>
<tr>
<td>flow</td>
<td>0</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td></td>
</tr>
<tr>
<td>lost</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>v</td>
<td>199</td>
<td>692</td>
<td>1147</td>
<td>446</td>
<td>497</td>
<td>252</td>
<td>360</td>
<td>199</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>s</td>
<td>1500</td>
<td>2100</td>
<td>2100</td>
<td>1700</td>
<td>2000</td>
<td>2700</td>
<td>2900</td>
<td>2700</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>v/s</td>
<td>0.13</td>
<td>0.22</td>
<td>0.85</td>
<td>0.26</td>
<td>0.18</td>
<td>0.09</td>
<td>0.10</td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td></td>
</tr>
<tr>
<td>current name</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
<tr>
<td>t_index</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
</tr>
</tbody>
</table>

Figure 28 is the optimal solution timing plan and Figure 29 shows the augmented tree for the optimal sequence with the cycle length of 88.4 s.
Figure 28: Charles Circle, Case 2 (Solution Timing Plan)

Cycle = 88.4 s
Comparing case 1 and case 2, in both cases, the critical circuit is $1 \rightarrow 95 \rightarrow 2 \rightarrow 1$. Same critical circuit means same cycle length, unless a stream has changed. Change is to stream 1, in which the left turn flow has been removed, along with one of the 3 lanes; however, with these compensating changes v/s scarcely changes, and so cycle length is almost the same.

The main difference is the better service for pedestrian, with two new (and critical) crossings, and a reduction in stream 96 from 6 to 2 lanes.
Chapter 10: Conclusion and Recommendations

Many researchers tried to search for the optimal sequences in complicate intersection. Most of these studies were done in 80’s. In that time they were concentrated on finding the algorithms. Previous research failed to find the optimal sequence in complicate intersections. This could be because computers were not very popular at that time or programming was not very simple. So they made the problems simpler to have a smaller model and tried to solve them and search for the optimal sequence. In this thesis, based on previous research theories, an algorithm was defined to search all possible sequences and the program was written to search for the optimal sequence.

By searching all sequences, optimal sequence won’t be missed but by using some assumptions to make the problem smaller, the optimal solution was missed in previous studies.

In new cities, grid networks were suggested to be easier for traffic planning, but from the urban planning aspect, it is not very nice to have a simple 4 leg intersections all around the city. Therefore, by presenting a program that is able to find the optimal solution for every complicate intersection, there will be more possible ideas in urban designs too.

The program is successful to find the optimal sequence, but it can be expanded to include more specific problems such as intersections with the simultaneous ending. The program is fast and accurate, so it can’t be improved more in this area but it can be improved to be easier to use. It might include graphs in output or have a nice input face to enter data.
References


Appendix: Full-Scale Figures
Figure 1: Traffic Streams at a 4-legged intersection
Figure 2: Asymmetric Location of the Conflict Point Between Streams 2 and 9
Figure 3: Leading Left versus Lagging Left Phase Sequences for a 4-leg, 8-phase Intersection
Figure 4: Directed Incompatibility Graph for a 4-legged intersection
If the defined cycle start is here, $F_{ij} = 0$

If the defined cycle start is here, $F_{ij} = 1$

\[ u_j - u_i \geq S_{\min ij} - c F_{ij} \]

So

\[ y_a = S_{\min ij} \]
\[ z_a = 0 \]
\[ F_a = F_{ij} \]

\[ u_i + F_{ji} \leq c - u_j \geq c - S_{\max ij} \]

or

\[ u_i - u_j \geq -S_{\max ij} + c - c F_{ji} \]

So

\[ y_a = -S_{\max ij} \]
\[ z_a = 1 \]
\[ F_a = F_{ji} = 1 - F_{ij} \]

*Figure 5: Formulating Staggered Start Constraints*
Figure 6: Circuit Analysis with a Simple 4-legged Junction
a. Directed Incompatibility Graph
   (Lost time, v/s ratio)

b. Intersection Layout

c. Maximal Conflict Group

d. Critical Circuit

e. Optimal Stream Sequence

Figure 7: The Five Stream - Odd-Hole Junction
Three Arm Junction with Three Vehicular Streams and Two Pedestrian Stream
Figure 8: Streams at a Three Arm Junction with Six Vehicular Streams and Single All Pedestrian Stream
Sequence Variables Equal to One

\[
\begin{align*}
F_{31} &= 1 & F_{51} &= 1 \\
F_{41} &= 1 & F_{52} &= 1 \\
F_{53} &= 1 & F_{Pi} &= 1 & \text{for } i = 1, \ldots, 6 \\
F_{63} &= 1
\end{align*}
\]

Flow parameters for test junction (Table 1 [9])

<table>
<thead>
<tr>
<th>Stream</th>
<th>(v)</th>
<th>(s)</th>
<th>(L)</th>
<th>(g_{\text{min}})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>687</td>
<td>1800</td>
<td>4</td>
<td>42.2</td>
</tr>
<tr>
<td>2</td>
<td>458</td>
<td>1800</td>
<td>3</td>
<td>28.5</td>
</tr>
<tr>
<td>3</td>
<td>229</td>
<td>1800</td>
<td>4</td>
<td>16.7</td>
</tr>
<tr>
<td>4</td>
<td>458</td>
<td>1800</td>
<td>3</td>
<td>28.5</td>
</tr>
<tr>
<td>5</td>
<td>229</td>
<td>1800</td>
<td>4</td>
<td>16.7</td>
</tr>
<tr>
<td>6</td>
<td>344</td>
<td>1800</td>
<td>4</td>
<td>23.1</td>
</tr>
<tr>
<td>(p)</td>
<td>-</td>
<td>-</td>
<td>18</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 9: Signal setting for a three-arm junction with an all-pedestrian phase
a. pivot arc (p) creating a directed circuit

b. entering (p) and exiting (x) arcs

Figure 10: Augmented tree (Ch. 5's example)
Figure 11: Network Simplex Algorithm for Minimizing c, given F

1. **Find Feasible Longest Path Solution**
   1.1 Set $c = c_{INC}$
   1.2 Determine the arc lengths for the given $F$ and $c$
   1.3 Apply the longest path algorithm to find the longest path from node $h$ to every other node. If a finite solution is found, it is a feasible solution to the original problem; identify its tree, set $i = h$, and go to step 2
   1.4 Otherwise, a directed cycle $k$ of positive length has been found. Find $y_k$ and $Z_k$ for this circuit (see eq. 24). If $y_k \geq 0$, STOP; there is no feasible solution with $c \leq c_{INC}$ for the given $F$
   1.5 Otherwise, decrease $c$ to $c = -y_k/Z_k$ and go to Step 1.2

2. **Find Minimal c for Given Longest Path Tree**
   2.1 Fanning outward on the tree from node $i$, find $\alpha_i$ and $\beta_i$ for each down-tree node (eq. 22)
   2.2 For each arc not in the tree, find $y'_a$ and $Z'_a$ (eq. 23)
   2.3 Find the non-tree arc that maximizes $r_a = (-y'_a/Z'_a)$, excluding arcs for which $y'_a \geq 0$. (Break ties arbitrarily.) Designate it as arc $p$, the pivot arc. (If there is no non-tree arc for which $y'_a < 0$, the problem is ill-formed; $c = 0$ is feasible and therefore optimal.)
   2.4 Let $c = r_p$

3. **Test and Pivot**
   3.1 If adding $p$ to the longest path tree creates a directed cycle, STOP; the minimal $c$ has been found
   3.2 Otherwise, add $p$ to the tree. Let $i =$ initial node of arc $p$. Remove from the tree the arc that has the same terminal node as $p$, and go to Step 2
Figure 12: Numerical Example for a Network Simplex Algorithm

a. Layout

b. Stream Sequence, 
Lost Time \((y)\) and v/s Ratio \((z)\) in Format \((y, z)\). Pedestrian Phases have 4 s leading start.
c. Initial Longest Path Tree, $c = 100 \ s$
Labels indicate $(y_i, z_i) = u_i$

d. First Iteration, $c = 87.5 \ s$
Arc $p$ enters, arc $x$ exits.

e. Second Iteration, $c = 87.5 \ s$
Arc $p$ enters, arc $x$ exits.

f. Third and Final Iteration, $c_{\text{exit}} = 76 \ s$
Arc $p$ enters, creates directed circuit.

Figure 12: Numerical Example for a Network Simplex Algorithm
Figure 12: Numerical Example for a Network Simplex Algorithm
Figure 13: Searching Using the Conflict Group Permutation Array
Figure 14: Branch and Bound Algorithm

Algorithm to find the stream sequence requiring the minimum cycle length.

1. Initialize Search Order and Variables

1.1 List conflict groups and order them in decreasing size

1.2 Initialize the incumbent $c_{\text{INC}}$. This may be done either by having the user simply enter a value known (or believed) to be feasible, or by having the user specify an initial value of $F$ and applying the circuit analysis, setting $c_{\text{INC}}$ to the resulting minimum cycle length.

1.3 Set $F_{ij} = F_{ji} = 1$ for every conflicting stream pair i-j

1.4 Choose a start stream (or set of start streams with simultaneous start), giving preference to the stream (or simultaneous start stream set) with the largest number of conflict. If possible, let it also be a member of the longest conflict group.

1.5 Call the start stream $h$. For every conflicting stream pair $h$-$j$, put arcs $h$-$j$ and $j$-$h$ into set $CA_o$, the initial set of arcs whose $F_{ij}$ values are fixed, and set $F_{hj} = 0$. (If there is a set of initial start streams, apply this step to each start stream, calling each start stream in turn $h$.)

1.6 If a stream $i$ incident to an arc $i$-$j$ belonging to $CA_o$ has a simultaneous start relationship with another stream $k$ and streams $j$ and $k$ are in conflict, set $F_{ki} = F_{ij}$, set $F_{ki} = 1 - F_{ki}$, and add arcs $i$-$k$ and $k$-$i$ to $CA_o$. 
2. Establish Correspondence Between Conflict group Order and Order in Which Pairwise Sequence Variables Are Set

2.1 Let G be the number of conflict groups, and g the conflict group index (corresponding to the list initially created in Step 1.1). Loop, for g = 1 to G:

2.1.1. Let $A_g$ be the set of all arcs $a(i,j)$ for which streams $i$ and $j$ belong to conflict group $g$, excluding arcs already in $CA_{g-1}$. $A_g$ represents the set of arcs whose $F_{ij}$ will be set when conflict group $g$ is processed.

2.1.2. If a stream $i$ incident to an arc $i-j$ that belongs to $A_g$ has a simultaneous start relationship with another stream $k$ and streams $j$ and $k$ are in conflict, add stream pairs $k-j$ and $j-k$ to $A_g$, unless they already belong to $A_g$.

2.1.3. If for any conflict group $g$, $A_g$ is an empty set, remove $g$ from the conflict group list. Reduce $G$ and update the conflict group indices accordingly.

2.1.4 Let $CA_g = CA_{g-1} \cup A_g$. $CA_g$ represents the cumulative set of arcs whose $F$ values are specified by conflict groups 1, …, $g$.

2.2 For every conflict group, determine the number of permutation $Q_g$. If a start stream is a member of the group, $Q_g = (n_g - 1)!$; otherwise, $Q_g = n_g!$, where $n_g = $ size of conflict group $g$. The permutations of conflict group $g$ will be indexed by $q_g = 1, …, Q_g$. Permutations have a natural “alphabetical” order, so that a particular permutation index is unambiguous.
3. Initialize Branch and Bound Tree

3.1 Set $g = 1$

3.2 Set $q_g = 1$

4. Evaluate and Test

4.1 For all arcs $a(i,j)$ for which streams $i$ and $j$ belong to conflict group $g$, generate a set of trial pairwise sequence values $TF_{ij}$ that correspond to the current permutation $q_g$. If any such arc belongs to $CA_{g-1}$ and $TF_{ij} \neq F_{ij}$, this permutation is inconsistent with the predecessor branches and the current branch is pruned; go to Step 6

4.2 For all arcs in $A_g$, set $F_{ij} = TF_{ij}$

4.3 Perform network optimization (see earlier algorithm). Let $c^* =$ minimum cycle length returned from the network analysis

4.4 If $c^* \geq c_{INC}$, the current branch is pruned (suboptimal); go to Step 6

4.5 If $u_i \geq c$ for any node $i$, the current branch is pruned (not a canonical rotation); go to Step 6

4.6 If $g = G$ the current solution has been completely specified

4.6.1 If $u_i < 0$ for any node $i$, the current branch is pruned (not a canonical rotation); go to Step 6

4.6.2 Otherwise, a new solution has been found; set $c_{INC} = c$ and go to Step 6
5. **Advance to Next Conflict Group**
   
   5.1 Set \( g = g + 1 \)
   
   5.2 Set \( q_g = 1 \) and go to Step 4

6. **Advance to Next Permutation Within a Conflict Group**
   
   6.1 Set \( q_g = q_g + 1 \)
   
   6.2 If \( q_g \leq Q_g \), go to Step 4. Otherwise, continue; conflict group \( g \) has been exhausted

7. **Retreat to Previous Conflict Group; Test for Termination**
   
   7.1 For all arcs \( a(i,j) \) belonging to \( A_g \), set \( F_{ij} = 1 \)
   
   7.2 Set \( g = g - 1 \)
   
   7.3 If \( g = 0 \), STOP; entire search tree has been exhausted. Otherwise, go to Step 6
Cycle = 64.4 s

Figure 15: 4 - legged intersection - Webster's Formula Solution
Figure 16: 4-legged intersection
Augmented Tree
Figure 17: Chapel Hill, Layout
Cycle = 38.5 s

Figure 18: Chapel Hill, Optimal Solution Timing Plan
Figure 21: Stage-base result comparison

a. Heydecker & Dudgeon

b. Our result
Cycle = 41.8 s

Figure 22: 4-legged intersection (optimal solution timing plan)
Figure 23: Charles Circle, Case 1 (Layout)
Figure 24: Charles Circle, Volume Counts (PM-peak)
Figure 25: Charles Circle - Case 1 (Solution Timing Plan)

Cycle = 93.8 s
Figure 28: Charles Circle, Case2 (Solution Timing Plan)

Cycle = 88.4 s
Figure 29: Charles Circle - Case 2
Augmented tree