Evaluation of Crossover Displaced Left-turn (XDL) Intersections and Real-time Signal Control Strategies with Artificial Intelligence Techniques

by

Ramanujan Jagannathan

Thesis submitted to the faculty of Virginia Polytechnic Institute and State University in partial fulfillment of the requirements for the degree of Master of Science in Civil Engineering

Dr. Dušan Teodorovic, Chair
Dr. John Collura
Dr. Sam Tignor

December 2003.
Falls Church, Virginia.

Keywords: Continuous Flow Intersections, CFI, XDL, Genetic Algorithms, Neural Networks, Real-time Signal Control.
DESIGN, OPERATIONAL PERFORMANCE AND REAL-TIME SIGNAL CONTROL STRATEGIES FOR XDL (CROSSOVER DISPLACED LEFT-TURN) INTERSECTIONS

Ramanujan Jagannathan

ABSTRACT

Although concepts of the XDL intersection or CFI (Continuous Flow Intersection) have been around for approximately four decades, users do not yet have a simplified procedure to evaluate its traffic performance and compare it with a conventional intersection. Several studies have shown qualitative and quantitative benefits of the XDL intersection without providing accessible tools for traffic engineers and planners to estimate average control delays, and queues. Modeling was conducted on typical geometries over a wide distribution of traffic flow conditions for three different design configurations or cases using VISSIM simulations with pre-timed signal settings. Some comparisons with similar conventional designs show considerable savings in average control delay, and average queue length and increase in intersection capacity. The statistical models provide an accessible tool for a practitioner to assess average delay and average queue length for three types of XDL intersections. Pre-timed signal controller settings are provided for each of the five intersections of the XDL network.

In this research, a “real-time” traffic signal control strategy is developed using genetic algorithms and neural networks to provide near-optimal traffic performance for XDL intersections. Knowing the traffic arrival pattern at an intersection in advance, it is
possible to come up with the best signal control strategy for the respective scenario. Hypothetical cases of traffic arrival patterns are generated and genetic algorithms are used to come up with near-optimal signal control strategy for the respective cases. The neural network controller is then trained and tested using pairs of hypothetical traffic scenarios and corresponding signal control strategies. The developed neural network controller produces near-optimal traffic signal control strategy in “real-time” for all varieties of traffic arrival patterns.
TABLE OF CONTENTS

ACKNOWLEDGEMENTS .................................................................................................................. v
LIST OF FIGURES .......................................................................................................................... vi
LIST OF TABLES ............................................................................................................................ viii
1 INTRODUCTION .......................................................................................................................... 1
2 RESEARCH OBJECTIVES ............................................................................................................. 3
3 MOTIVATION FOR RESEARCH ................................................................................................... 4
4 ORGANIZATION OF THESIS ..................................................................................................... 5
5 LITERATURE REVIEW ................................................................................................................. 6
  5.1 Design and Operational Performance of XDL Intersections ....................................................... 6
  5.2 Development of Real-time Signal Control Strategy of XDL Intersection ................................. 7
6 BRIEF DESCRIPTION OF XDL INTERSECTIONS ..................................................................... 11
7 DESIGN AND OPERATIONAL PERFORMANCE OF INTERSECTIONS ....................................... 13
  7.1 Cases Modeled ......................................................................................................................... 16
  7.2 Estimating average delay and queue for the XDL intersection ................................................. 29
  7.3 Conclusions and Recommendation ......................................................................................... 34
8 BRIEF OVERVIEW OF GENETIC ALGORITHMS .................................................................... 37
9 BRIEF OVERVIEW OF NEURAL NETWORKS ............................................................................. 42
10 DEVELOPMENT OF REAL-TIME SIGNAL CONTROL STRATEGY FOR XDL INTERSECTION .... 48
  10.1 Analysis Methodology .............................................................................................................. 51
  10.2 Testing Results ........................................................................................................................ 57
  10.3 Conclusions ............................................................................................................................. 59
11 CONCLUSIONS AND RECOMMENDATIONS ........................................................................... 61
12 REFERENCES .............................................................................................................................. 64
13 APPENDICES .............................................................................................................................. 66
  13.1 Appendix A – SAS CODE ......................................................................................................... 66
  13.2 Appendix B – MATLAB CODE .............................................................................................. 67
ACKNOWLEDGEMENTS

I am very thankful to my advisor, Dr. Dušan Teodorovic for his personal guidance, constant support and encouragement. I am especially indebted to him for providing me with financial assistance throughout my graduate education at Virginia Tech. My sincere thanks also go to Dr. Joe Bared at Federal Highway Administration for sponsoring and guiding me through part of this research. I am also highly indebted to my thesis advisory committee members, Dr. John Collura and Dr. Sam Tignor, for their support and encouragement.

I would also like to thank my parents and friends for all the sacrifices they have undertaken to make my academic career a success.

Finally, I am ever grateful to the Almighty God for bestowing me with the inner strength and peace of mind in this endeavor.
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1</td>
<td>Typical Geometry for XDL Intersection on four approaches</td>
<td>11</td>
</tr>
<tr>
<td>Figure 2</td>
<td>Differences in left-turn treatment between XDL intersection and conventional intersection</td>
<td>12</td>
</tr>
<tr>
<td>Figure 3</td>
<td>Pedestrian approach and refuge areas at an XDL Intersection</td>
<td>14</td>
</tr>
<tr>
<td>Figure 4</td>
<td>Typical Geometry for XDL intersection on four approaches and comparable conventional intersection: Case-A</td>
<td>18</td>
</tr>
<tr>
<td>Figure 5</td>
<td>Signal Phasing Scheme for XDL intersection: Case-A</td>
<td>19</td>
</tr>
<tr>
<td>Figure 6</td>
<td>Signal Cycle and Splits from VISSIM Phasing Scheme for XDL intersection: Case-A</td>
<td>20</td>
</tr>
<tr>
<td>Figure 7</td>
<td>Schematic diagram of five intersections in an intersection having XDL on all four approaches</td>
<td>20</td>
</tr>
<tr>
<td>Figure 8</td>
<td>Typical Geometry for XDL intersection and comparable conventional intersection: Case-B</td>
<td>23</td>
</tr>
<tr>
<td>Figure 9</td>
<td>Signal Phasing Scheme for XDL intersection: Case-B</td>
<td>24</td>
</tr>
<tr>
<td>Figure 10</td>
<td>Typical Geometry for XDL Intersection and comparable conventional intersection: Case-C</td>
<td>26</td>
</tr>
<tr>
<td>Figure 11</td>
<td>Signal Phasing Scheme for XDL intersection: Case-C</td>
<td>27</td>
</tr>
<tr>
<td>Figure 12</td>
<td>Overview of Modeling Process</td>
<td>28</td>
</tr>
<tr>
<td>Figure 13</td>
<td>Delay Estimation Tables for XDL intersection: Case – A</td>
<td>33</td>
</tr>
<tr>
<td>Figure 14</td>
<td>Delay Estimation Tables for XDL intersection: Case – B</td>
<td>34</td>
</tr>
<tr>
<td>Figure 15</td>
<td>Generalized optimization procedure using Genetic Algorithms</td>
<td>38</td>
</tr>
<tr>
<td>Figure 16</td>
<td>Roulette-wheel selection strategy</td>
<td>39</td>
</tr>
<tr>
<td>Figure 17</td>
<td>Stochastic universal sampling strategy</td>
<td>40</td>
</tr>
<tr>
<td>Figure 18</td>
<td>Single-point crossover strategy</td>
<td>41</td>
</tr>
<tr>
<td>Figure 19</td>
<td>Schematic diagram of an artificial neuron with activation function</td>
<td>43</td>
</tr>
<tr>
<td>Figure 20</td>
<td>Schematic diagram of a two-layered feedforward neural network</td>
<td>44</td>
</tr>
<tr>
<td>Figure 21</td>
<td>Taxonomy of training a multilayered perceptron</td>
<td>46</td>
</tr>
<tr>
<td>Figure 22</td>
<td>Sample signal control strategy for the central intersection of the XDL intersection</td>
<td>52</td>
</tr>
<tr>
<td>Figure 23</td>
<td>Sample traffic arrival pattern for the central intersection of the XDL intersection</td>
<td>53</td>
</tr>
<tr>
<td>Figure 24</td>
<td>Convergence achieved using genetic algorithms for obtaining near-optimal signal control strategy</td>
<td>55</td>
</tr>
<tr>
<td>Figure 25</td>
<td>Neural Network used to determine green light extension for XDL intersection. (Central intersection)</td>
<td>57</td>
</tr>
</tbody>
</table>
## LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1</td>
<td>Signal Timing Control Settings for XDL intersection : Case-A</td>
<td>21</td>
</tr>
<tr>
<td>Table 2</td>
<td>Comparison of Traffic Performance-XDL intersection and conventional intersection: Cases A, B &amp; C</td>
<td>31</td>
</tr>
<tr>
<td>Table 3</td>
<td>Model Statistics for XDL intersection : Cases A, B &amp; C</td>
<td>32</td>
</tr>
<tr>
<td>Table 4</td>
<td>Selection probability and fitness value of individuals</td>
<td>39</td>
</tr>
<tr>
<td>Table 5</td>
<td>Individual before and after binary mutation</td>
<td>41</td>
</tr>
<tr>
<td>Table 6</td>
<td>Traffic arrivals and the optimal signal control for each time interval (Central intersection)</td>
<td>56</td>
</tr>
<tr>
<td>Table 7</td>
<td>Desired and Calculated output values of green time extension (Central intersection)</td>
<td>58</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

High volume intersections (especially during peak hours) pose a particularly difficult challenge to traffic engineers and planners interested in alleviating delays and improving safety for passengers and pedestrians. The major side effects of traffic congestion include increased pollution, higher driver stress levels, and greater economic losses in terms of wasted time. Researchers, attempting to reduce congestion, delay, and crashes have suggested several innovative intersection designs for heavy traffic flow situations. These unconventional intersection designs include the quadrant roadway intersection, median U-turn, superstreet median, bowtie, jughandle, split intersection and XDL (crossover displaced left-turn) intersection (also called CFI). The most influential factor in the intersection performance for these heavy flow situations is achieved by reducing the number of phases in the signal cycle, although traffic diversion may also be employed.

This thesis analyzes the performance of an XDL intersection relative to a conventional intersection with detailed concentration on traffic engineering perspectives. Secondly with ITS initiatives, intelligent traffic control strategies have become an important research arena. Research on implementation techniques for better traffic signal control systems is currently being pursued. In wake of these developments, a real-time traffic signal control strategy for an intersection having XDL on all four approaches is developed and analyzed in detail.

This thesis is organized as follows. The first section comprises of a literature review on two aspects: 1) the XDL concept and other studies specific to design, operational
performance and implementation of XDL intersections, 2) “real-time” signal control strategies for intersections. Additionally, a brief overview of XDL intersections and related traffic operation principles is provided. The following section analyses the design and operational performance of XDL intersections and also presents comparison with conventional intersections. The next section provides a brief overview of genetic algorithms and neural network concepts that are used in the development of real-time signal strategy for XDL intersections. The following section deals with development of real-time signal control strategy for XDL intersections. Lastly, conclusions and recommendations for future research are presented.
2 RESEARCH OBJECTIVES

The main objectives of this research are:

- To conduct a literature review on the XDL concept and other studies specific to the design, operational performance and implementation of the XDL and “real-time” signal control strategies for intersections.
- To carry out an extensive analysis on geometric design and operational performance of XDL intersections and comparison with conventional intersections having similar geometries and characteristics using pre-timed traffic signal controllers.
- To develop a neural network controller producing near-optimal signal time settings for the XDL intersection using real-time traffic signal control strategies.
- To provide recommendations for future research in the areas mentioned above.
3 MOTIVATION FOR RESEARCH

Federal Highway Administration (FHWA) has an ongoing initiative to study the performance of unconventional intersections especially for high traffic volumes. The CFI or XDL Intersection is an unconventional intersection and few studies are available showing a detailed traffic performance analysis as well as geometrical information for implementation of such intersections. This study was undertaken to provide an in-depth analysis of XDL intersections in terms of design issues, operational performance and signal timing controls. Additionally, a “real-time traffic signal control strategy” was developed using genetic algorithms and neural networks for superior traffic control at the XDL intersection.
4 ORGANIZATION OF THESIS

The thesis is structured into the following sections:

- Literature Review
- Brief Overview of XDL Intersections.
- Analysis of Design and Operational Performance of XDL Intersections.
- Brief Overview of Genetic Algorithms and Neural Networks.
- Development of Real-time Signal Control Strategy of XDL Intersection.
- Conclusion and Recommendations.
5 LITERATURE REVIEW

5.1 Design and Operational Performance of XDL Intersections

Reid and Hummer (2001) comparing unconventional intersections to their conventional counterparts suggested “The continuous flow intersection always had the highest move-to-total-time-ratio of all designs, keeping traffic moving as its name implies.” They also suggested “The continuous flow intersection probably needs the smallest right-of-way of all the unconventional designs (quadrant roadway intersection, median U-turn, superstreet median, bowtie, jughandle, split intersection and continuous flow intersection) examined here.” In his thesis, Chick (2001) suggests, “The Displaced Right Turn junction (in Great Britain) is a multi-node intersection which improves overall junction capacity through the removal of conflicts at the center of the intersection.” The Traffic Control Systems Handbook (1996) suggests “A recent study comparing the performance of traffic operations at a CFI with that of operations at a similar conventional intersection indicated a 60 percent increase in capacity at the CFI.” Other benefits noted by the Traffic Control Systems Handbook included substantial reduction in auto emission pollutants and significant increase in average speed. KLD Associates, Inc. refers to the CFI as the Dispersed Movement Intersection (DMI) and concludes that this type of intersection “can provide comparable capacity at a fraction of the cost of a grade separation.” They also mention that it increases intersection capacity without compromising safety. Dowling College sponsored a human factors study of DMI operations to determine how the design affected the driving task. Their results suggest “…about 80% of the first time users of the DMI intersection expressed positive comments about the design. After about a week of use, 100% of the daily drivers samples
expressed positive comments about the design. The intersection is easily negotiated by drivers who are initially unfamiliar with the design and that after a short learning curve, nearly all drivers are familiar and comfortable with the DMI intersection.” Goldblatt et. al., (1994) conclude that “the advantages of a CFI over a conventional intersection are most pronounced when demand approaches or exceeds the capacity of conventional designs and when heavy left-turn movements require protected phases.” Hutchinson (1994) suggests in his paper comparing the CFI and the conventional intersection, “It is clear that when conflicting flows are heavy, the CFI design is greatly superior.” Hutchinson (1974) earlier noted that, “The results clearly support the claims of Al Salman and Salter, showing a great increase in capacity for right-turners (Great Britain) and a corresponding reduction in delay, particularly at high flows for right-turners.”

Of late there has been some interest in the USA to apply the XDL intersection design concept. Presently, there is an XDL intersection at the entrance of the Dowling College National Aviation and Transportation Center, Oakdale, New York and a T-intersection built at the crossing of routes MD 210 and MD 228 in Prince Georges County, Maryland. It is also known that more than a dozen sites have been built in Mexico in pursuance of this concept.

5.2 Development of Real-time Signal Control Strategy of XDL Intersection

The primary objective of any traffic engineer analyzing an intersection for traffic performance is to optimize the signal settings to minimize delay, stops and other concomitant effects like vehicular pollution, noise pollution etc. The two basic approaches for optimization of signal timing as summarized by Teodorovic et. al (2002) are listed below.
Maximizing bandwidth efficiency:

In this approach, unrestricted movement of traffic across a corridor is considered as the main criteria for traffic performance. To achieve this objective, the traffic platoons are formed at intersections and the signal timing is adjusted to facilitate platoon progression. Teodorovic et. al. (2002) state that “PASSER II and MAXBAND are two of the most advanced and versatile software programs that use the progression method for optimizing the arterial signal intersections”.

Minimizing a disutility index:

In most cases, presence of a traffic control primarily results in savings of time and fuel. In an engineer’s terminology the same can be classified as delays, stops, queue length, fuel consumptions, emissions, etc. The weighted combination of these factors is traditionally termed as disutility index. The main objective of this method is to adjust signal timings in order to minimize the disutility index. Teodorovic et. al (2002) state that TRANSYT-7F, SIGOP-III, SYNCHRO and SSTOP are examples of software using this model.

Types of signal control strategies:

Different signal control strategies commonly used at intersections are summarized below:

**Time of Day (TOD) Traffic Control Strategy** is implemented on a time of day, day of week schedule. The strategy consists of optimized splits, cycle lengths, and offsets developed offline using the traffic pattern variation over the years.

**Traffic Responsive Signal Control Strategy** does not work on a fixed schedule as in the case of TOD. Instead, traffic responsive control implements timing plans based on actual traffic conditions. As traffic reaches certain predefined thresholds, optimized timing plan corresponding to that particular traffic condition is implemented.
**Traffic Adaptive Control Strategy**: As opposed to the models outlined above, which use historical data to create one or more optimized timing plans, adaptive control strategies use real time data from detectors to perform constant optimization on the signal timing plan for an arterial or a network. This means that signals can adapt to non-recurring congestion, incidents, events, or traffic demand growth over time, without needing to be reset. As traffic arrival pattern varies, the optimization algorithm determines the best parameter values (green split, cycle length and offset) for the given real-time traffic conditions subject to the minimum and maximum green time constraints. OPAC, SCAT, SCOOT are some of the working real time traffic adaptive control systems.

**OPAC (Optimized Policies for Adaptive Control)**. OPAC is a distributed real-time traffic signal control system that continuously adapts signal timings to minimize a performance function of total intersection delay + stops over a pre-specified network. OPAC can operate as an independent smart controller, or as part of a coordinated system. OPAC also optimizes offset and cycle length.

**SCAT (Sydney Coordinated Adaptive Traffic System)**. Developed by the New South Wales Department of Main Roads, SCATS is a dynamic control system with a decentralized architecture. SCATS updates intersection cycle length using the detectors at the stop line. SCATS allows for phase skipping. Offsets between adjacent intersections are predetermined and adjusted with the cycle time and progression speed factors.

**SCOOT (Split Cycle Offset Optimization Technique)**. SCOOT is an off-the-shelf centralized computerized traffic control model developed at the Transportation Road
Research Laboratory in the U.K. It is an enhancement over first generation UTCS systems and provides real-time adaptive control. SCOOT uses system detectors to measure traffic flow profiles in real time, and along with predetermined travel times and the degree of saturation (the ratio of flow-to-capacity), predicts queues at intersections. Adjustments of cycle length, phase splits and offsets are made in small steps to operate at a preset degree of saturation (usually 90%). Tests have shown that SCOOT is most effective when demand almost approaches capacity without exceeding it, where demand is unpredictable, and when distances between intersections are short. Traffic control systems using SCOOT are prevalent in Australia, Asia, and recently in North America. ALLONS-D, UTOPIA, PRODYNE, UTOPIA, and RHODES are some of the other popular real time traffic adaptive control algorithms.

Teodorovic et al. (2002) used dynamic programming and neural networks to design an intelligent traffic control system for a four-approach intersection having four consecutive phases (one phase per approach). Comparisons showed that the output “green extension” of the neural network controller was nearly equal to the best solution obtained by using dynamic programming. Teodorovic et al. (2001) when studying real-time traffic control at the isolated “T” intersection used genetic algorithm to find the near-optimal solution for a given traffic arrival pattern. A fuzzy rule base was developed using the data generated. Fuzzy logic was used to build a traffic signal controller. Comparisons showed that the output “time until the next phase change” of the fuzzy logic controller was nearly equal to the best solution obtained by using genetic algorithms. In this thesis, we use the same methodological approach proposed by Teodorovic et al. (2001, 2002) to build a real-time signal control strategy for XDL intersections.
6 BRIEF DESCRIPTION OF XDL INTERSECTIONS

The XDL intersection is an at-grade intersection. The fundamental design principle of the XDL intersection involves displacement of the left turn lane to the other side of the opposing through lanes a few hundred feet before the intersection. The displaced left turn lanes are aligned parallel to the through lanes at the intersection. This geometric design results in a simultaneous movement of the left turning phases with the through movements thereby saving one phase per direction. This saving is significant especially when there are high percentages of left-turners at the intersection. Figure 1 illustrates the geometric design of an intersection using XDL on all four approaches.

Figure 1: Typical Geometry for XDL Intersection on four approaches
Figure 2 illustrates the differences in left-turn treatment at XDL intersection compared with conventional intersections.

Figure 2 : Differences in left-turn treatment between XDL intersection and conventional intersection
To study in detail the traffic performance of the XDL Intersection, VISSIM was chosen for traffic modeling. The choice of VISSIM was made considering the versatility of the software in allowing unconventional movements to be modeled in congruence with field operations. ITC-World, distributor of VISSIM in the US, defines it as a microscopic, time step and behavior based simulation model developed to analyze the full range of functionally classified roadways. In our study, three different geometric design configurations for the XDL were analyzed and compared to conventional intersections. The comparison intersections are conventional designs with similar geometric characteristics, dimensions and traffic flows to the XDL for a few parallel scenarios.

The first endeavor to calculate the signal timings was made using Synchro (signal optimization software). The modeling of the displaced left turn movement was not possible as simulation showed cars getting stuck in the short gaps between the hypothetical intersections used to model the actual movement of traffic through the XDL intersection. Transyt 7f was used but it had a limitation of having a minimum 50 feet distance between intersections and hence the XDL intersections could not be broken into a group of hypothetical intersections. When pedestrian presence is considered, the time taken for pedestrians to cross the roads in two cycles is the most dominating factor influencing signal timing for all modeled cases in this study. In order to accommodate pedestrian crossing of a XDL intersection in fewer phases, some phase lengths have to be increased and coordinated in favor of pedestrians. But it is also observed that longer cycle lengths defeat the underlying principle of the XDL intersection that operates best
with short cycles. Hence, the approach adopted in this study was to set the minimum phase length long enough to accommodate the pedestrian crossing of at least two through movements in a single phase. Medians 10 feet x 10 feet wide (light-gray colored areas) are provided as refuge areas for pedestrians and are illustrated in Figure 3 below.

![Figure 3: Pedestrian approach and refuge areas at an XDL Intersection](image)

Signalizing the right turn lanes can be adopted to handle heavy pedestrian flows safely. Pedestrians will have to cross the intersection, against one protected movement at a time, over 1.5 –2 cycles. However, it was observed that absence of pedestrian traffic at an intersection permits the reduction of cycle lengths to as low as 46 seconds further reducing the vehicular delay compared to longer cycle lengths accommodating pedestrian
traffic. Corresponding cases for the conventional intersection signal timings were made using signal timings optimized with Transyt7f for the modeled traffic flows. The signal phasing scheme had protected left followed by through movements, for both directions. In the cases modeled, the phase lengths were always long enough that pedestrian presence was not a dominating factor on the cycle time.

The signal controller used in modeling is a pre-timed controller. The amber and red times are calculated using the standard ITE formulae. The ITE formulae for determining the amber interval with a subsequent red clearance interval is as follows:

\[ y = t + \frac{v}{2a+2Gg} \]

where:

- \( y \) = Length of amber interval in seconds
- \( t \) = Driver perception/reaction time, generally assumed as 1.0 second
- \( v \) = Velocity of approaching vehicle (m/sec)
- \( a \) = Average deceleration (m/sec2)
- \( g \) = Acceleration due to gravity (9.81 m/sec2)
- \( G \) = Grade of approach in percent divided by 100 (downhill is negative)

\[ r = \frac{(w+l)}{V} \]

where:

- \( r \) = the length of the all red phase expressed in seconds, and follows the yellow change interval.
- \( w \) = width of intersection, curb to curb expressed in feet.
- \( l \) = vehicle length, taken as 20 feet.
V = posted speed in feet/second.

The traffic composition modeled is the default option in VISSIM with two percent trucks. The driver behavior is based on the Wiedemann model used in VISSIM. All other network parameters are as default in VISSIM. The average speed of the vehicles is 45 mph on the roads, 30 mph on the displaced left turn lane and 15 mph while executing the turn. The average pedestrian walking speed is assumed to be 4 feet/sec.

7.1 Cases Modeled
The three cases modeled are: A) four legged intersection with four corresponding displaced left-turn lanes, B) four legged intersection with only two opposing displaced left-turn lanes on the major road, and C) T intersection with one displaced left-turn lane.

Case A:
The intersection model has three through lanes per direction, two left turn lanes and one right turn lane per approach for all four approaches. The displaced left turn lane before the main intersection has a length of 325 feet. The right turn bay has a length of 250 feet. The left turn bay before the separation of the displaced left turn has a length of 350 feet. All acceleration lanes for right turning vehicles are 300 feet. The median between the through lanes in opposite directions is 10 feet wide, the median between the through lane and the displaced left turn lane is 10 feet wide while the median between the through and the right turn lane is about 6 feet. Please refer to Figure 4 for more details. The comparable conventional intersection has similar geometric features and dimensions as the XDL intersection described above on all four approaches. The traffic flows on the approaches of the XDL are randomly generated. A large number of cases modeled have directional flows to replicate peak-hour directional flows at intersections. The range used
for the left turns is between 100-750 vehicles/hour (vph) in one direction. The range for
the through traffic is 300-2,650 vph in one direction. The range for the right turning
vehicles is 50-350 vph in one direction. The green interval is 30 seconds for phase 1 and
2, and the corresponding yellow time + all-red intervals are 4+2 seconds. The green
intervals are 43 seconds and 17 seconds for phases 3, 4 and 5, 6 respectively and the
yellow time + all-red intervals are 4+2 seconds for all these movements (please refer to
Figure 5). The total cycle length for all scenarios is 72 seconds. Figure 6 is a good
representation of all the phases that are mainly under two phases 1 and 2. The gray
rectangle in each phase represent the green time while the gaps between the consecutive
phases represent the yellow + all-red time. The signal timing and the controller setting
required for case A is displayed in Table 1. The intersections illustrated in Table 1
correspond to the respective intersections shown in Figure 7. Table 1 shows the signal
setting required for the different signal heads and pre-timed phasing scheme
implemented. The signal setting values shown in Table 1 are derived from the signal
timings discussed previously in the section.
Figure 4: Typical Geometry for XDL Intersection on four approaches and comparable conventional intersection: Case – A
Figure 5: Signal Phasing Scheme for XDL Intersection: Case-A
Figure 6: Signal Cycle and Splits from VISSIM Phasing Scheme for XDL Intersection: Case-A

Figure 7: Schematic diagram of five intersections in an intersection having XDL on all four approaches
Table 1: Signal Timing Control Settings for XDL Intersection: Case-A

Case B:

The intersection model has three through lanes per direction, two-left turn lanes and one right turn lane per approach for the two major road approaches (Figure 8). The displaced left turn lane before the main intersection has a length of 325 feet. The right turn bay has a length of 275 feet. The left turn bay before the separation of the displaced left turn has a length of 350 feet. The acceleration lanes for the right turning vehicles are 300 feet. The other two legs have the conventional geometric design with two through lanes in each
direction, one left turn lane at the major crossing and one right turn lane. The right turn bay before the intersection has a length of 300 feet. The left turn bay before the intersection has a length of 350 feet on the major road. The median between the through lanes in opposite direction is 10 feet, the median between the through lane and the displaced left turn lane is 10 feet while the median between the through and the right turn lane is about 6 feet. Please refer to Figure 8 for more details. The comparable conventional intersection has similar geometric features and dimensions as the XDL intersection described above on all four approaches. The traffic flows on all the approaches are randomly generated. A large number of cases modeled have directional flows to replicate peak-hour directional flows at intersections. The range used for the left turns on the major roads is between 100-700 vph. The range for the through traffic on the major roads is 300-2,200 vph in one direction. The range for the right turning vehicles on the major roads is 50-350 vph in one direction. The range used for the left turns on the minor roads is between 50-200 vph in one direction. The range for the through traffic on the minor roads is 50-1,200 vph in one direction. The range for the right turning vehicles on the minor roads is 50-250 vph in one direction. The green times are 10 seconds, 25 seconds, 30 seconds and the yellow time + all-red time is 3+2 seconds, 4+2 seconds, 4+2 seconds for phases 1, 2, and 3 at the main intersection respectively (please refer to Figure 9). The green times are 53 seconds and 17 seconds for phases 4 and 5 respectively and the yellow time + all-red time is 4+2 seconds for all the movements at the junction of the displaced left turn and the through movement. The cycle length used for all scenarios is 82 seconds.
Figure 8: Typical Geometry for XDL intersection and comparable conventional intersection: Case-B
Case C:

This T intersection has three through lanes per direction on the major road and a displaced left turn double lane on the eastern approach, three through lanes per direction and one right lane on the western approach and two left lanes and one right lane on the T-
leg which is on the southern approach (Figure 10). The displaced left turn lane before the main intersection has a length of 325 feet. The right turn bay has a length of 300 feet on the western approach. The left turn bay before the separation of the displaced left turn has a length of 350 feet. The acceleration lanes for the right turning vehicles are 300 feet. The southern approach has the conventional geometric design with two left lanes and one right turn lane. The left turn bay before the intersection has a length of 350 feet. The median between the through lanes in the opposite directions is 10 feet, while the median between the through and the right turn lane is about 6 feet. The geometry can be further improved if a separate acceleration lane is provided for the right-turning vehicles on the southern approach. The comparable conventional intersection has similar geometric features and dimensions as the XDL intersection described above on all three approaches. The traffic flows on all the approaches are randomly generated. A large number of cases modeled have directional flows to replicate peak-hour directional flows at intersections. The range used for the displaced left turns on the eastern approach is 50-750 vph. The range for the through traffic on the eastern and western approaches is 300-2,650 vph per direction. The range for the right turning on the western approach vehicles is 50-350 vph. The range used for the left turns on the southern approach is between 100-1,450 vph. The range used for the right turns on the southern approach is between 50-750 vph. The green time is 30 seconds for phases 1 and 2, and the yellow time + all-red time is 4+2 seconds for all the movements at the main intersection (please refer to Figure 11). The green times are 43 seconds and 17 seconds for phases 3 and 4 respectively and the yellow time + all-red time is 4+2 seconds for all the movements at the junction of the displaced left turn and the through movement. The cycle length for all scenarios is 72 seconds.
Figure 10: Typical Geometry for XDL intersection and comparable conventional intersection: Case-C
A specialized program was developed to extract specified parameters from each of the output files generated by VISSIM. The list of files used as input for VISSIM was also used as input for the data extraction tool. The procedure is illustrated in Figure 12. The program processed each output file generated by VISSIM and returned a list with the following variables listed:
- Average control delay = deceleration, acceleration, move-up, and stop delay (second/vehicle)
- Average queue length = average queue length in the network (feet)

These output parameters provided the basis of the statistical analysis and subsequent model development.
7.2 Estimating average delay and queue for the XDL intersection

The comparison between cases A, B, C and their conventional counterparts is presented in Table 4 for four scenarios per case. The numbers of randomly generated and modeled scenarios are 743 for case A, 714 for case B and 262 for case C. The simulations were run for a time period of one hour for each scenario. A number of traffic characteristics for the network are available such as queues, number of stops, number of lane changes, speed on links, etc. However, only a few variables were chosen for representation in Table 2. The modeled entering flow refers to the total entering traffic volume that was modeled for the intersection. However, the actual number of vehicles entering the intersection is controlled by VISSIM and hence may vary slightly. Another notable fact is the significant variation between number of vehicles that can be serviced in reality and number of vehicles modeled especially for over-saturated traffic conditions at conventional intersections.

A general qualitative comparison may not be adequate to assist planners and traffic engineers in selecting the proper intersection configuration for various flow conditions. Statistical models are therefore developed using the data from all modeled scenarios for the three cases A, B and C to estimate two variables of interest commonly used in assessing intersection traffic performance. The models were developed using the non-linear regression technique readily available in the SAS software (Proc NLIN) to express an exponential form. After several trials and iterations of different variables and model forms, we have accepted the form given below for predicting average control delay (CD\textsubscript{CASE}) in seconds/vehicle and average queue (AQ\textsubscript{CASE}) in feet.
CDCASE-A = EXPO \[a_0 + (a_1XWN + a_2XES + a_3XNE + a_4XSW + a_5XNS + a_6XEW + a_7XWE + a_8XSN)/10000]\]

AQCASE-A = EXPO \[b_0 + (b_1XWN + b_2XES + b_3XNE + b_4XSW + b_5XNS + b_6XEW + b_7XWE + b_8XSN)/10000]\]

CDCASE-B = EXPO \[a_0 + (a_1XWN + a_2XES + a_3XNE + a_4XSW + a_5XNS + a_6XEW + a_7XWE + a_8XSN)/10000]\]

AQCASE-B = EXPO \[b_0 + (b_1XWN + b_2XES + b_3XNE + b_4XSW + b_5XNS + b_6XEW + b_7XWE + b_8XSN)/10000]\]

CDCASE-C = EXPO \[a_0 + (a_1XES + a_2XEW + a_3XWE + a_4XSW)/10000]\]

AQCASE-C = EXPO \[b_0 + (b_1XES + b_2XEW + b_3XWE + b_4XSW)/10000]\]

where,

\[a\text{ and } b\text{ are regression coefficients with corresponding measures of significance, and}
\]

\[\text{model goodness of fit given in Table 3.}\]

\[XWN = \text{flow from the western approach towards northern approach (vph)},\]
\[XES = \text{flow from the eastern approach towards southern approach (vph)},\]
\[XNE = \text{flow from the northern approach towards eastern approach (vph)},\]
\[XSW = \text{flow from the southern approach towards western approach (vph)},\]
\[XNS = \text{flow from the northern approach towards southern approach (vph)},\]
\[XEW = \text{flow from the eastern approach towards western approach (vph)},\]
\[XWE = \text{flow from the western approach towards eastern approach (vph)},\]
\[XSN = \text{flow from the southern approach towards northern approach (vph)}.\]

\[\text{EXPO (exponential)} = e = 2.716828\]
All variables are significant beyond the 95% confidence level. Goodness-of-fit measures (in terms of the conventional R-squared) are strong for almost all models, except control delay for the T intersection (case C). However, it is an acceptable model for a planning level estimation. Although the regression is based on a non-linear model, the R-squared statistic is acceptable according to Kvalseth (1985), when very few outliers are present.

<table>
<thead>
<tr>
<th>Intersection Type</th>
<th>Entering Flow (veh/hr)</th>
<th>Delay (s/veh)</th>
<th>Stops (#/veh)</th>
<th>Queues (feet)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CASE – A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conventional</td>
<td>3000</td>
<td>199</td>
<td>744</td>
<td>105</td>
</tr>
<tr>
<td>XDL</td>
<td>3000</td>
<td>215</td>
<td>749</td>
<td>106</td>
</tr>
<tr>
<td>Conventional</td>
<td>5000</td>
<td>594</td>
<td>1526</td>
<td>235</td>
</tr>
<tr>
<td>XDL</td>
<td>5000</td>
<td>618</td>
<td>1563</td>
<td>241</td>
</tr>
<tr>
<td>Conventional</td>
<td>7000</td>
<td>558</td>
<td>1998</td>
<td>287</td>
</tr>
<tr>
<td>XDL</td>
<td>7000</td>
<td>577</td>
<td>2002</td>
<td>293</td>
</tr>
</tbody>
</table>

For geometric characteristics and dimensions of the intersections modeled, please refer to the descriptions given under the headings Case A, Case B, Case C and corresponding Figures 4, 8 and 10.

Table 2: Comparison of Traffic Performance-XDL intersection & conventional intersection: Cases A, B & C
Table 3: Model Statistics for XDL Intersection: Cases A, B & C

- Blank field indicates that the variable is not included in the model for Case-C

For general estimations and comparisons, Figures 13, 14 and 14A offer estimates of average control delays at the XDL intersection for given traffic flows of Cases A, B and Case C respectively. The values for the comparison of the traffic performance at XDL intersection with the conventional intersection are obtained from Table 2.
Figure 13: Delay Estimation Tables for XDL intersection: Case-A
### Table 1: Delay Estimation Tables for XDL Intersection: Case-B

<table>
<thead>
<tr>
<th>Entering Flow (veh/hr)</th>
<th>XDL Intersection Delay (s/veh)</th>
<th>Conventional Intersection Delay (s/veh)</th>
<th>% Reduction in Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>22.2</td>
<td>53.1</td>
<td>58.2</td>
</tr>
<tr>
<td>5500</td>
<td>22.3</td>
<td>54.5</td>
<td>59.1</td>
</tr>
<tr>
<td>9000</td>
<td>27.7</td>
<td>165.0</td>
<td>83.2</td>
</tr>
<tr>
<td>10500</td>
<td>71.7</td>
<td>250.1</td>
<td>71.3</td>
</tr>
</tbody>
</table>

*Figure 14: Delay Estimation Tables for XDL intersection: Case-B*

### Figure 14A: Delay Estimation Tables for XDL Intersection: Case-C

### 7.3 Conclusions and Recommendation

Although the results and conclusions apply specifically to selected geometric dimensions and traffic characteristics in Cases A to C, the advantages still apply to the design concept where speeds, dimensions and signal settings can be varied to optimize performance, given the constraints of individual sites. For the cases modeled, the XDL Intersection or CFI consistently outperforms the conventional intersection even at low traffic volumes. The reduction in number of phases on approaches having XDL geometries results in
tremendous vehicular delay savings as well as a considerable increase in capacity of the intersection.

- The average delay and queue estimation models can help traffic engineers and planners compare the XDL intersection with other types of intersections to measure suitability of application, especially when traffic congestion at the intersection is a serious problem.

- From Table 2, we see that the percent reduction in average intersection delay due to the introduction of XDL on approaches of a conventional intersection is 48% to 85%, 58% to 71% and 19% to 90% for Cases A, B and C respectively.

- From Table 2, we see that the percent reduction in the average number of stops due to the introduction of XDL on approaches of a conventional intersection is 15% to 30% for under-saturated traffic flows and the 85% to 95% for saturated traffic flow conditions at the conventional intersection.

- From Table 2, we see that the percent reduction in average intersection queue length due to the introduction of XDL on approaches of a conventional intersection is 62% to 88%, 66% to 88% and 34% to 82% for Cases A, B and C respectively.

- From Table 2, we see that the percent increase in capacity of the intersection due to the introduction of XDL on approaches of a conventional intersection is 30%, 30% and 15% for Cases A, B and C respectively. However, it is important to state here that all the cases modeled had signal timings adjusted for pedestrian presence. In the absence of pedestrians, cycle lengths can be lowered resulting in
average intersection delay in the range of 14 seconds/veh to 19 seconds/veh at low and medium traffic volumes for Case A.

- Even for a single pre-timed signal setting, the XDL intersection works effectively for all combination of traffic flows (low, medium, heavy). This is unique and can be very useful for intersections not having sophisticated signal controllers.

The XDL intersection or CFI can be a very cost-effective solution for high traffic volume intersections needing conversion to grade-separated interchanges to improve the LOS. Additional right of way requirement for the XDL intersection is mostly due to the necessity of right-turn lanes with corresponding acceleration lanes. The reduction of delay due to implementation of XDL intersections will also have a positive impact on transit operations in the corridor. Finally, other recommended areas of research worth looking into would be the sensitivity of results to more efficient signalization at conventional intersections. Other helpful research is to study actuated signal control of XDL intersections, optimization of pedestrian service times, human factor issues for drivers and pedestrians, traffic performance with heavy truck volumes, and vehicular pollution effects.
8 BRIEF OVERVIEW OF GENETIC ALGORITHMS

The proposed real-time signal control model is based on genetic algorithms and neural networks. Hence, a brief introduction to genetic algorithms is provided in this section. Genetic algorithms are stochastic search methods that replicate behavior of proposed theories on natural biological evolution. The underlying principle in the proposed theory on natural biological evolution is “survival of the fittest”. The genetic algorithm operators operate on a population of individuals to produce results closer to the optimum solution. Each iteration involves the creation of new population by selecting the best individuals in the present population and applying operators similar to the proposed forces directing evolution in nature. Every consecutive iteration aims at creating a population of fitter individuals successfully adapting to their environment than their ancestors. The main genetic operators found in nature such as selection, recombination and mutation are successfully applied to the population to increase their fitness value. Figure 15 illustrates the process schematically.

The first step involves random generation of a population and evaluation of fitness of the generated population. The termination criteria are then checked. If the criteria are not satisfied, a new population is created. In the first step, the best individuals according to their fitness value are selected. The better the fitness value, the higher is the probability of an individual to become the parent. The genetic material of the parents is crossed over at one or multiple sites to produce offspring. The produced offspring are mutated with a certain probability. The offspring are ranked as per their fitness value. A new population is created where the better offspring replace their parents to increase quality of the population in the problem domain. This cycle is performed until the termination criteria
are reached. A brief description of the different genetic algorithm operators used in the analysis is given in the rest of the section.

**FIGURE 15. Generalized optimization procedure using Genetic Algorithms**

**Selection Strategy**

The different selection strategies commonly used in genetic algorithms are rank-based fitness assignment, roulette wheel selection, stochastic universal sampling, local selection, truncation selection and tournament selection. Pohlheim states, “The simplest selection scheme is roulette-wheel selection, also called stochastic sampling with replacement.” The technique applied in roulette-wheel selection strategy consists of mapping individuals to the contiguous segments of line where the fitness of an individual would govern the size of the individual’s segment. Similar to a roulette-wheel game, a
random number is selected and the individual whose segment includes that random number is selected for mating. The selection process is repeated to fill the population. An example is given below:

11 individuals are represented in Table 1 below with their fitness values and selection probability. The sum of all fitness values is $(3 + 2.7 + 2.4 + 2.1 + 1.8 + 1.5 + 1.2 + 0.9 + 0.6 + 0.3 + 0) = 16.5$. The selection probability of individual 1 is $3/16.5 = 0.18$. Similarly, the selection probability is calculated for all individuals.

<table>
<thead>
<tr>
<th>Number of individual</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>
</tr>
</thead>
<tbody>
<tr>
<td>Fitness value</td>
<td>3.0</td>
<td>2.7</td>
<td>2.4</td>
<td>2.1</td>
<td>1.8</td>
<td>1.5</td>
<td>1.2</td>
<td>0.9</td>
<td>0.6</td>
<td>0.3</td>
<td>0.0</td>
</tr>
<tr>
<td>Selection probability</td>
<td>0.18</td>
<td>0.16</td>
<td>0.15</td>
<td>0.13</td>
<td>0.11</td>
<td>0.09</td>
<td>0.07</td>
<td>0.05</td>
<td>0.04</td>
<td>0.02</td>
<td>0.00</td>
</tr>
</tbody>
</table>

**Table 4: Selection probability and fitness value of individuals**

Figure 16 below shows the individual segment occupancy. It can be observed that by virtue of its fitness value, Individual 1 has the largest segment length. Let us assume that we have to choose six individuals. For choosing six individuals, the corresponding random numbers are $0.81, 0.32, 0.96, 0.01, 0.65, 0.42$. The corresponding individuals chosen are 6, 2, 9, 1, 5, 3.

**Figure 16: Roulette-wheel selection strategy**

Thus it can be observed that the roulette-wheel selection algorithm provides a zero bias but does not guarantee minimum spread. Pohlheim defines bias as the absolute difference
between an individual's normalized fitness and its expected probability of reproduction and spread as the range of possible values for the number of offspring of an individual. To solve the problem of minimum spread, the stochastic universal sampling selection strategy is used. Similar to roulette-wheel selection strategy, the individuals occupy contiguous segments, having segment lengths based on their individual fitness. For the example presented above, the number of individuals is six. Hence the distance between the 6 individuals is 1/6 =0.167. The position of the first pointer is a random number chosen between the closed intervals of 0 and 0.167. Figure 17 illustrates this strategy. Pohlheim remarks, “Stochastic universal sampling ensures a selection of offspring which is closer to what is deserved then roulette wheel selection.”

![Figure 17: Stochastic universal sampling selection strategy](image)

**Recombination Strategy**

The different binary valued recombination strategies commonly used in genetic algorithms are single-point crossover, multi-point crossover, uniform crossover, shuffle crossover and crossover with reduced surrogate. The recombination strategy used in the analysis was single point crossover and is illustrated in Figure 18. Given two parent individuals, a random number is chosen within the length of the genetic string for the point of genetic crossover. At that point, the genetic material is exchanged between two parents to produce two offspring.
Figure 18: Single-point crossover strategy

The following example is given in this regard to illustrate this recombination strategy.

Considering two individuals binary strings of length 11, we have:

Individual 1  0 1 1 1 0 0 1 1 0 1 0
Individual 2  1 0 1 0 1 1 0 0 1 0 1

The offspring created by the crossover of genetic material at crossover point 7 are:

Offspring 1  0 1 1 1 0 0 1| 0 1 0 1
Offspring 2  1 0 1 0 1 1 0| 1 0 1 0

Mutation Strategy

For individuals having binary value representation, mutation means flipping of variable values. At any random location, the value of the variable is flipped to produce mutation. Mutation is one of the best ways to retain freshness of samples in a population. An example of a binary mutation for an individual with 11 variables having variable 6 mutated is shown in Table 5 below.

<table>
<thead>
<tr>
<th>Before mutation</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>After mutation</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5: Individual before and after binary mutation
9 BRIEF OVERVIEW OF NEURAL NETWORKS

The proposed real-time signal control model is based on genetic algorithms and neural networks. Hence, a brief introduction to neural networks is provided in this section.

The human brain and its extraordinary working have always been a subject of fascination for one and all. Its immaculate precision and astounding celerity in decision-making has been beyond the reach of any machined instrument. The field of artificial neural networks as the name suggests draws inspiration from the setup of the human brain. The fundamental component of the human brain is called a neuron.

The human brain is composed of roughly $10^{11}$ neurons. The communication system of these neurons is termed as neural networks. Inspired by the communication structure of neurons within the human brain with enhances parallel processing ability of the brain, scientists have come up with artificial neurons and communication links between them termed as artificial neural networks. Neural networks have been used extensively by researchers to approximate real-time phenomena occurring in nature or under controlled set of experimental conditions. This is primarily due to the ability of neural networks to generalize from previous experiences and apply the acquired knowledge to the scenario at hand. The theorem proved by Hornik et al. (1989) and Cybenko (1989) state that a multilayered feedforward neural network with one hidden layer can approximate any continuous function up to a desired degree of accuracy provided it contains a sufficient number of nodes in the hidden layer. The import of this theorem is that feedforward neural networks can approximate any function and hence work as universal approximators.
Characteristics of neural networks

A binary input, binary output, and fixed activation threshold were the components of the first model of an artificial neuron that was proposed by McCulloch and Pitts (1943). A schematic of the same is shown below in Figure 19.

![Schematic diagram of an artificial neuron with activation function](image)

**Figure 19:** Schematic diagram of an artificial neuron with activation function

The input signals \(i_1, i_2, i_3, \ldots, i_n\), representing the output signals of other neurons, are multiplied by associated weights, \(w_1, w_2, w_3, \ldots, w_n\). The final output value (NET), which is the weighted sum of input signals is compressed by an “S” curve such that the value of the output signal, OUT, never exceeds a relatively low level regardless of the value of NET. Various activation functions are used and the commonly used ones are step function, sigmoid function, hyper tangent function and identity function. Teodorovic et al. (1998) state that a neural network is primarily characterized by the following components: a) a set of processing elements, b) connectivity of those elements, c) the rule of signal propagation through the network, d) activation of transfer functions, e) training algorithms, and f) environment in which the network functions. A neural network contains three types of nodes: input, output, and hidden. Input nodes receive inputs from sources outside the network. Output signals transmit signals outside the network. The hidden nodes have their input and output signals within the network and they are not
“seen” from outside the network. Each node transmits signals of different weights to other connected nodes. The connectivity of the network nodes is very important in a neural network. It is basically of two types, partial and complete. In partial connectivity, all the network nodes of one layer are not connected to all the network nodes of the adjoining layer. Each link or connection in the network is associated with a weight (positive or negative) modifying the strength of the signal. The output signal of a node is the weighted sum of input signals modified by an activation function. An example of a multilayered feedforward neural network is given in Figure 20.

In a multilayered feedforward network the input signal extends forward through several layers and in every layer the signal is processed to estimate the network’s output signal. Each node is a processing element associated with the corresponding activation function by which the weighted sum of input values is transformed to determine the output value.

*Figure 20: Schematic diagram of a two-layered feedforward neural network*
Training of a neural network

Teodorovic et al. (1998) state that the most fascinating feature about neural networks is their ability to modify their behavior in response to the environment. Presented with a set of input and corresponding output data, they self-adapt to incite appropriate responses. There are various types of training algorithms that differ in their modeling, learning and validation properties. The modeling abilities of an algorithm determine the range of nonlinear functions that can be reproduced. The structure of a neural network model can influence the convergence rate of a training algorithm, determining the type of learning to be used. The multilayered neural networks became popular after the development of an error back propagation algorithm, which was used for training a network. The proposed back propagation algorithm is a gradient procedure where the activation functions of nodes are bounded, continuous, monotonously increasing, nonlinear, differential functions. The output function of the network is a continuous, differentiable weight function enabling the search of the extremum by the “gradient descent” algorithm.

The optimal weights, $w_{ij}$, are determined by the rule of gradient descent (delta rule, generalized delta rule) minimizing the criterion function or error. Each iteration of the algorithm (cycle or epoch defined as the process of transmission of one or a few training pairs through the network whereby the error is calculated) contains two passes as can be seen in Figure 21 (Teodorovic et al. (1998))

- Propagation of one or a set of input signals forward to the output layer (in the original algorithm input signals were brought to the network individually)
- Backward pass where the computed error extends backward in order to calculate the changes of parameters (weight of the network’s branches).
The procedure is performed in many iterations using the same training pairs until the error becomes “sufficiently” small.

![Taxonomy of training a multilayered perceptron](image)

**Figure 21 : Taxonomy of training a multilayered perceptron**

**Generalization**

The application of neural networks relies on the ability of an algorithm to generalize correctly from a limited number of samples, which means that the algorithm has to interpolate and locally extrapolate precisely. For an intended accuracy level of results, if the required minimum amount of sample data is provided for learning, the algorithm must draw the relevant knowledge from the set of data. After the neural network has been designed, to a certain extent, the reaction can be insensitive to minor variations in the input set of data. This ability to differentiate in the presence of noise and distortion of shapes is critical.

**Validation of model**

The validation of a trained neural network model is done using testing data. Generally, training algorithms are able to successfully learn a set of training data but the ability has
to be verified. The general method used is to divide the data in two parts, $\frac{1}{3}^{rd}$ and $\frac{2}{3}^{rd}$. $\frac{2}{3}^{rd}$ of the data is used to train the network, and $\frac{1}{3}^{rd}$ of the data is used to test the model. Thus, if the model properly generalizes the testing data, then the model can be validated.
Fixed-time or traffic responsive control strategies are the most common types of signal control strategies implemented at intersections. In contrast to control strategies mentioned above, which use historical data to create one or more optimized timing plans, adaptive control strategies use real time data from detectors to perform constant optimizations on the signal timing plan for an arterial or a network. This means that signals can adapt to non-recurring congestion, incidents, events, or traffic demand growth over time, without needing to be reset. There is a significant interest in the development of real-time signal control strategies to control intersections and corridors. In this thesis, a real-time control strategy for XDL intersections is developed and tested. The methodology used to develop a real-time signal controller for XDL intersections is based on the approach proposed by Teodorovic et al. (2001, 2002). Teodorovic et al. (2002) used dynamic programming and neural networks to design an intelligent traffic control system for a four-approach intersection having four consecutive phases (one phase per approach). Comparisons showed that the output “green extension” of the neural network controller was nearly equal to the best solution obtained by using dynamic programming. Teodorovic et al. (2001) when studying real-time traffic control at the isolated “T” intersection used genetic algorithm to find the near-optimal solution for a given traffic arrival pattern. A fuzzy rule base was developed using the data generated. Fuzzy logic was used to build a traffic signal controller. Comparisons showed that the output “time until the next phase change” of the fuzzy logic controller was nearly equal to the best solution obtained by using genetic algorithms.
The basic assumptions for the proposed methodology as proposed by Teodorovic et al. (2001, 2002) are listed below. It is initially assumed that it is possible to develop a real-time signal control strategy that makes real-time decisions of high quality. The developed signal control system will be able to interpret different traffic situations and make appropriate decisions incognizant of the functional relationships between individual variables. The real-time signal control system deployed should be able to generalize, adapt, and learn based on new knowledge and new information.

The proposed real-time traffic control system comprises of the following components:

1) Traffic arrival strings
2) Traffic signal control strategies
3) Creation of a real-time traffic signal control system
4) Response testing for the developed system

Different traffic arrival pattern sets are produced using uniform and Poisson distributions. Every traffic arrival pattern represents a possible “traffic scenario” which the signal control system may be exposed to. In order to derive the optimal or “good” traffic control strategy for a unique traffic arrival pattern, different optimization and heuristic techniques can be employed. Meta-heuristic technique using genetic algorithms has been chosen to find the optimal or near-optimal signal control strategies. For different traffic arrival patterns, the best control strategies are generated. These control strategies are then used to develop a neural network model that is analogous to the heart of the real-time traffic signal control system. The primary objective is to train the neural network model to minimize the deviation between the optimal output and the neural network’s output for the control system. Using the traffic arrival data primarily recorded by the installed traffic
detectors as inputs, the neural network model is trained to determine the best signal control strategy for the arriving traffic pattern in terms of extension of the green time provided to the currently operating phase.

As shown in Figure 1 an isolated intersection having XDL on all four approaches can be said to be composed of 5 independent but coordinated intersections. There is one intersection in the middle and 4 intersections on the four arms of the XDL intersection. Each intersection primarily operates on two phases and the signal control strategy using pre-timed signal control is discussed at depth in Section 7.

An important assumption as stated by Teodorovic et al. (2001, 2002) is that the intersection is relatively “busy” and under-saturated with significant demand fluctuations on all approaches. Offline-optimized timing plans are generated relative to the traffic arrival patterns which have been historically collected over the years. The vehicular traffic changes on a day-by-day and hour-by-hour basis. Therefore, using a pre-timed signal setting is not the best solution since it does not adapt instantly to changing traffic arrival patterns.

Traffic detectors placed upstream of the signal controller would be source of traffic arrival information. The time lag is assumed to be 5 sec. The signal control strategy has therefore to be developed for changing traffic scenario is small steps of 5 seconds. The signal control strategy should be able to give appropriate green times to minimize a performance index of linear combination of total number of stops and the total delay. The functional form of the performance index, which has to be optimized, is the same used by Teodorovic et. al (2002):

\[ F = w_s S + w_d D \]
where:

\( w_s \) - the weight (the importance) given to the total number of stopped vehicles,

\( w_d \) - the weight (the importance) given to the total delay,

\( S \) - the total number of stopped vehicles in all the approaches during a cycle,

\( D \) - the total delay of all vehicles in all the approaches during a cycle.

\[ w_s + w_d = 1 \]

10.1 Analysis Methodology

The following algorithm illustrated the strategy adopted to develop the proposed real-time traffic signal control strategy:

Step 1: Develop a set of hypothetical vehicle arrival patterns for the XDL intersection, and determine the optimal green time for all the four phases for each pattern using genetic algorithms.

Step 2: Train a neural network, by learning from the set of vehicle arrival patterns and corresponding optimal green times obtained in the previous step.

Step 3: Test the effectiveness of the neural network by comparing the best results obtained by using genetic algorithms and the one obtained by the proposed method.

The time span for simulation has been chosen as 180s. The choice was made based on the complexity of the operations involved and the fact that each intersection operates on just two phases. The time increment used in the model was 5 sec. A mix of Uniform and Poisson arrival patterns was used to model traffic arrival at the XDL Intersection. 3700 scenarios were modeled altogether.
The time span of 180 seconds was divided into 5-second intervals to form a binary string of 36 genes in a chromosome. The presence of ‘1’ gene indicates the current status of a phase is either green or yellow and the presence of ‘0’ gene indicates that the phase is now showing all-red. The last ‘1’ in a series of consecutive ‘111….’ refers to the yellow interval. A sample string and its representation is shown below in Figure 22.

![Figure 22: Sample signal control strategy for the central intersection of the XDL intersection](image)

1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 1 1 0 0 0 0 0 0 0 1 1

Each of the five intersections works primarily on a two-phase cycle offset relative to the central signal controller. Hence the choice of binary representation was appropriate as the operation of one phase indicates the non-operation of the complimentary phase in the cycle. Each intersection has a binary string representing the corresponding signal timing.

The traffic arriving at the intersection was also a corresponding sequence of 36 entries depending on the movement type. Since the right turners arriving at the central intersection have separate lanes and merge with the thru traffic past the intersection on the arms of the XDL, the arrival pattern of right-turning vehicles was not considered in the modeling process. Based on results of traffic analysis carried out in Section 7, maximum flow rate for thru moving vehicles was taken to be 2650 veh/hr. Similarly, the maximum flow rate for left turning vehicles was taken to be 750 veh/hr.

A sample traffic pattern generated for thru moving vehicles is given below in Figure 23.
Figure 23: Sample traffic arrival pattern for the central intersection of the XDL intersection

At the central intersection, the eastbound and westbound thru traffic compete for green time against the northbound and southbound thru traffic. At the junctions of the XDL, the imminent left-turning vehicles compete with the thru moving vehicles and the vehicles that have executed the left-turn at the central intersection.

The traffic string generated and the signal phasing string generated for all the 5 intersections of the XDL intersection is an input for the genetic algorithms. Genetic algorithms used operate with the following parameters. Number of individuals per population set is 100. Generation Gap is 0.9. Generation gap is defined as the fraction of the population to be reproduced every generation. If Generation Gap is smaller than 1.0, less offspring than individuals in population are produced, thus, some individuals of the population survive. If Generation Gap is greater than 1.0 more offspring than individuals in population are produced. Not all offspring are inserted in the population. Maximum number of generations for termination is 200.

The initial population of 100 strings is generated for each of the 5 intersections. Alternatively, Webster’s formula can be used to create strings to be used as starting points thereby reducing the population size. Then each member of the population is evaluated for fitness using the fitness function. As per the fitness of the members, ranking
is performed on the members. Then the members to be used for crossover are selected using stochastic universal sampling technique. Crossover type used is single point crossover where genetic material is exchanged at exactly one point in the string. The probability of binary mutation used is based on a popular formula, $0.7/\text{length of the individual string} \approx 0.0194$. The fitness of the newly generated population is calculated using the fitness function. The reinsertion technique used is fitness-based reinsertion. The fitness-based reinsertion scheme implements a truncation selection between offspring before inserting them into the population. In truncation selection individuals are sorted according to their fitness and only the best individuals are selected. Some of the code used in programming was adapted from the “geatbx toolbox” developed by Pohlheim.

The minimum phase length for any phase is 10 seconds. The initial number of vehicles present at the intersection for each approach is a randomly generated. The number of vehicles arriving at the intersection for each approach is known from the arrival string. During the green phase for an approach, number of vehicles passing through the approach is calculated and remaining vehicles, if any, are added up to the sum of the arrivals in the red phase. For every 5 sec, the number of vehicles multiplied by 5 sec gives the vehicular delay and number of vehicles arriving during a red phase gives the count of stopped vehicles. An equal weight has been assigned to delays and stops in this study. The delay plus the stops for all the approaches constitute the disutility for the intersection that is minimized using genetic algorithms to search for the best signal setting. The changing of weights of stops and delays in the disutility function may change the extension time.
granted to an approach. However, these sensitivity tests have been not been carried out in this thesis.

Figure 24 is a Matlab screen shot showing process of obtaining the near-optimal signal control by using genetic algorithms for the central intersection. The figure shows how the convergence occurs quickly in the beginning and slowly stabilizes over each generation.

![Figure 24: Convergence achieved using genetic algorithms for obtaining near-optimal signal control strategy](image)

Thus by using genetic algorithms, the near-optimal solution containing the green times for all the 5 intersections are developed for many hypothetical traffic scenarios. The hypothetical scenarios generated and the corresponding optimal green time values constitute the input-output pairs for the neural network similar to the formulation suggested by Teodorovic et. al (2002). For the central intersection, at every time step of 5 seconds, cumulative traffic arrivals on all four approaches, amount of green time elapsed for the green-approach ($G_{\text{elapsed}}$) and the green time extension for the current phase (Extension), the data is organized as shown in Table 6.
Table 6: Traffic arrival and the optimal signal control for each time interval (Central Intersection)

<table>
<thead>
<tr>
<th>No</th>
<th>Time Period</th>
<th>Cumulative Traffic Arrivals (central intersection)</th>
<th>$G_{elapsed}$ (sec)</th>
<th>Extension (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Approach 1</td>
<td>Approach 2</td>
<td>Approach 3</td>
</tr>
<tr>
<td>1</td>
<td>0-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>5-10</td>
<td>1</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>10-15</td>
<td>1</td>
<td>15</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>15-20</td>
<td>3</td>
<td>24</td>
<td>14</td>
</tr>
</tbody>
</table>

... (Continued table)

As can be seen from the table, cumulative traffic values on approaches and the green time elapsed are inputs to the neural network model. The neural network was built and trained using the computer software “NeuroSolutions”. The neural network created has 1 input layer, two hidden layers, an output layer, five inputs (four cumulative traffic arrivals for four approaches, and the amount of green time elapsed for the green-approach), and one output (green light extension). A schematic of the neural network is shown in Figure 25.
10.2 Testing Results

The robustness of any neural network model is gauged by subjecting it to testing. As stated in the previous section, traffic arrival patterns and corresponding near-optimal traffic signal control strategy were evaluated using genetic algorithms. The solutions obtained using genetic algorithms were checked against the solutions produced by the neural network as the reference for evaluation of the robustness of the model. The neural network was trained with 2700 rows of data for each of the five intersections. The neural networks results are compared with the solutions produced by the genetic algorithm optimization for both the trained cases and the test cases. The test cases are cases that have been previously optimized using genetic algorithms but are not used in training the neural network. Thus, the testing data is foreign to the neural network. The accuracy of the neural network for each of the five intersections was tested with 1000 rows of data.
The testing of the neural network on foreign data makes sure that the comparison is not biased. The mean squared error obtained equals 4.5%. A sample of the genetic algorithm optimized values and neural network output values of green time extension for a few cases are given in Table 7.

<table>
<thead>
<tr>
<th>Genetic Algorithm output (steps of 5 sec)</th>
<th>Neural Network Output (steps of 5 sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.0000000</td>
<td>5.831082</td>
</tr>
<tr>
<td>5.0000000</td>
<td>4.856625</td>
</tr>
<tr>
<td>4.0000000</td>
<td>3.911463</td>
</tr>
<tr>
<td>1.0000000</td>
<td>0.928748</td>
</tr>
<tr>
<td>2.0000000</td>
<td>1.778258</td>
</tr>
<tr>
<td>1.0000000</td>
<td>1.016393</td>
</tr>
<tr>
<td>4.0000000</td>
<td>4.101101</td>
</tr>
<tr>
<td>1.0000000</td>
<td>0.990106</td>
</tr>
<tr>
<td>3.0000000</td>
<td>2.917038</td>
</tr>
<tr>
<td>2.0000000</td>
<td>1.973655</td>
</tr>
<tr>
<td>3.0000000</td>
<td>3.120036</td>
</tr>
</tbody>
</table>

*Table 7: Desired and Calculated output values of green time extension (central intersection)*

As can be seen from the results, the neural network outputs are very close to the genetic algorithm outputs. The discrepancy noticed is also because of the use of a continuous function in neural networks that produces continuous values as opposed to the discrete value outputs given by optimization using genetic algorithms. However, by appropriate rounding to the nearest decimal, this minor discrepancy can be successfully removed. Thus, the neural network developed can be used to build “real-time traffic signal control strategy” for field applications.
10.3 Conclusions

In this section an intelligent tool was developed capable of recognizing the traffic arrival patterns in real-time produce optimal or near optimal signal strategies for an XDL intersection. Extensive off-line data was accumulated using traffic arrivals patterns and corresponding near-optimal signal control strategies using genetic algorithms. This data was used for training the neural-network that for each new traffic scenario there was a corresponding new signal strategy. At the end of training, the neural network controller produces near-optimal signal control strategy for real-time traffic arrival patterns.

In summary, the steps involved in the development of a real-time signal control strategy based on the approach proposed by Teodorovic et al. (2001, 2002) are listed below:

Step 1 Basic structure of the neural network is formulated for the given objective by defining the variables to be used as inputs and outputs.

Step 2 A large number of vehicle arrival patterns are generated for each of the five intersections of the XDL intersection.

Step 3 For each pattern, using genetic algorithms, the near-optimal signal control strategy for all the approaches of an intersection resulting in the minimum value of disutility index is found for all the five intersections.

Step 4 Pairs of optimal signal settings and corresponding vehicle arrival patterns and grouped together in an input – output table.

Step 5 A neural network is developed and trained using NeuroExpert software for the defined set of input-output pairs.

Step 6 The neural network is tested with the real time traffic pattern fed to the neural network in steps.
The neural network controller developed produces real-time traffic signal control for XDL intersection. The performance of the controller is very close to the near-optimal solutions obtained using genetic algorithms. Thus with the inputs of traffic arrival patterns on the approaches, the real-time signal controller can successfully determine the extension which should be provided to the current green phase for optimal traffic performance of the intersection. The maximum phase length was 40 seconds and the minimum was 10 seconds that are in the same range of values obtained for best signal control strategies using pre-timed signal controllers for low, medium and high traffic flow scenarios in Section 7. Practically negligible CPU times were achieved, and were thus absolutely acceptable for the “real time” application of the developed neural network controller. Finally, other recommended areas of research worth looking into would be real-time signal control strategy for other unconventional intersection types.
In this thesis, an extensive analysis of XDL intersections is presented specific to design, operational performance and real-time signal control strategies. For the cases modeled, the XDL Intersection or CFI consistently outperforms the conventional intersection even at low traffic volumes. The reduction in number of phases on approaches having XDL geometries results in tremendous vehicular delay savings as well as a considerable increase in capacity of the intersection. The average delay and queue estimation models can help traffic engineers and planners compare the XDL intersection with other types of intersections to measure suitability of application, especially when traffic congestion at the intersection is a serious problem. From Table 2, we see that the percent reduction in average intersection delay due to the introduction of XDL on approaches of a conventional intersection is 48% to 85%, 58% to 71% and 19% to 90% for Cases A, B and C respectively. From Table 2, we see that the percent reduction in the average number of stops due to the introduction of XDL on approaches of a conventional intersection is 15% to 30% for under-saturated traffic flows and the 85% to 95% for saturated traffic flow conditions at the conventional intersection. From Table 2, we see that the percent reduction in average intersection queue length due to the introduction of XDL on approaches of a conventional intersection is 62% to 88%, 66% to 88% and 34% to 82% for Cases A, B and C respectively. From Table 2, we see that the percent increase in capacity of the intersection due to the introduction of XDL on approaches of a conventional intersection is 30%, 30% and 15% for Cases A, B and C respectively. However, it is important to state here that all the cases modeled had signal timings adjusted for pedestrian presence. In the absence of pedestrians, cycle lengths can be
lowered resulting in average intersection delay in the range of 14 seconds/veh to 19 seconds/veh at low and medium traffic volumes for Case A. Even for a single pre-timed signal setting, the XDL intersection works effectively for all combination of traffic flows (low, medium, heavy). This is unique and can be very useful for intersections not having sophisticated signal controllers.

The XDL intersection or CFI can be a very cost-effective solution for high traffic volume intersections needing conversion to grade-separated interchanges to improve the LOS. Additional right of way requirement for the XDL intersection is mostly due to the necessity of right-turn lanes with corresponding acceleration lanes.

To the best of my knowledge, this is the first attempt made to develop real-time control strategies for XDL intersections. The steps involved in the development of a real-time signal control strategy based on the approach proposed by Teodorovic et al. (2001, 2002) are as follows. In the first step, the basic structure of the neural network is formulated for the given objective by defining the variables to be used as inputs and outputs. In the second step, a large number of vehicle arrival patterns are generated for each of the five intersections of the XDL intersection. In the third step, genetic algorithms is used to obtain the near-optimal signal control strategy for all the approaches of an intersection minimizing value of disutility index for each traffic pattern generated. In the fourth step, pairs of optimal signal settings and corresponding vehicle arrival patterns and grouped together in an input – output table. In the fifth step, a neural network is developed and trained using NeuroExpert software for the defined set of input-output pairs. Lastly, the
neural network is tested with the real time traffic pattern fed to the neural network in steps.

The neural network controller developed produces real-time traffic signal control for XDL intersection. The performance of the controller is very close to the near-optimal solutions obtained using genetic algorithms. Thus with the inputs of traffic arrival patterns on the approaches, the real-time signal controller can successfully determine the extension which should be provided to the current green phase for optimal traffic performance of the intersection. The maximum phase length was 40 seconds and the minimum was 10 seconds that are in the same range of values obtained for best signal control strategies using pre-timed signal controllers for low, medium and high traffic flow scenarios in Section 7. Practically negligible CPU times (microseconds) were achieved, and were thus absolutely acceptable for the “real time” application of the developed neural network controller.

Finally, other recommended areas of research worth looking into would be the sensitivity of results to more efficient signalization at conventional intersections and real-time signal control strategy for other unconventional intersection types. Other helpful research is to study actuated signal control of XDL intersections, optimization of pedestrian service times, human factor issues for drivers and pedestrians concerning ease of navigation (adaptability) and safety through the XDL intersection, traffic performance with heavy truck volumes, and vehicular pollution effects.
12 REFERENCES


Sample SAS Code for Case-C

data Tsec.Xdl6x6out13;
set Tsec.Xdl6x613;

    XNS = N_Sveh;
    XNE = N_Eveh;
    XNW = N_Wveh;
    XWE = W_Eveh;
    XWN = W_Nveh;
    XWS = W_Sveh;
    XSW = S_Wveh;
    XSN = S_Nveh;
    XSE = S_Eveh;
    XEN = E_Nveh;
    XEW = E_Wveh;
    XES = E_Sveh;
    XMAXEW = maxEWflow;
    XMAXNS = maxNSflow;
    XTOTRT = totright;

if (XNS GE XSN)
    then XMAXNS = XNS;
else XMAXNS = XSN;

if (XEW GE XWE)
    then XMAXEW = XEW;
else XMAXEW = XWE;

proc nlin;

    parms c0=2.4 to 2.6 by 0.1 c1=-12 to -11 by 0.5 c4=46 to 47 by 0.5 c7 = 60 to 61 by 0.5
    c10 = 1 to 2 by 0.5 c15 = 29 to 30 by 0.5 c16 = 23 to 24 by 0.5 c17 = -85 to -84.5 by 0.5
    c8=42 to 43 by .5 c6 =16 to 17 by .5 c3 =-14 to -13 by .5 c12 =14.5 to 15 by .5;
    model Alldelay = exp(c0 + c4*XES/10000 + c6*XEW/10000 + c3*XWE/10000 + c12*XSW/10000);
    output out=p2 p2 =predicted;
    run;*/
13.2 Appendix B – MATLAB CODE

clear all;
clc;
duration=180;
min_int=5;
str_len=duration/min_int;
max_byte_lim=2^36;
max_traf_lim=2^52;
max_flow1=2650;
max_flow2=750;
max_flow3=750;
max_flow4=750;
max_flow5=750;

NIND = 100;           % Number of individuals per subpopulations
MAXGEN = 200;        % maximum Number of generations
GGAP = .9;           % Generation gap,
NVAR = 1;           % No. of variables
PRECI = 36;          % Precision of binary representation

for loop = 1:13:2080
  % Generate Random Flows
  flow11=round(max_flow1*rand);
  flow12=round(max_flow1*rand);
  flow13=round(max_flow1*rand);
  flow14=round(max_flow1*rand);
  flow2=round(max_flow2*rand);
  flow3=round(max_flow3*rand);
  flow4=round(max_flow4*rand);
  flow5=round(max_flow5*rand);
  flow_mat=[flow11 flow12 flow13 flow14 flow2 flow3 flow4 flow5];

  tf11=max(2,ceil(flow11/3600*min_int)*2+1);
  tf12=max(2,ceil(flow12/3600*min_int)*2+1);
  tf13=max(2,ceil(flow13/3600*min_int)*2+1);
  tf14=max(2,ceil(flow14/3600*min_int)*2+1);
  tf2=max(2,round(flow2/3600*min_int)*2+1);
  tf3=max(2,round(flow3/3600*min_int)*2+1);
  tf4=max(2,round(flow4/3600*min_int)*2+1);
  tf5=max(2,round(flow5/3600*min_int)*2+1);
  tf_mat=[tf11 tf12 tf13 tf14 tf2 tf3 tf4 tf5];

% Traffic Pattern Generation
int_traf11=trafgen1(tf11,PRECI);
int_traf12=trafgen1(tf12,PRECI);
int_traf13=trafgen1(tf13,PRECI);
int_traf14=trafgen1(tf14,PRECI);
int_traf2=trafgen1(tf2,PRECI);
int_traf3=trafgen1(tf3,PRECI);
int_traf4=trafgen1(tf4,PRECI);
int_traf5=trafgen1(tf5,PRECI);

store(loop,:)=int_traf11;
store(loop + 1,:)=int_traf12;
store(loop + 2,:)=int_traf13;
store(loop + 3,:)=int_traf14;
store(loop + 4,:)=int_traf2;
store(loop + 5,:)=int_traf3;
store(loop + 6,:)=int_traf4;
store(loop + 7,:)=int_traf5;

% Actual Generated Flows

sumtf11=sum(int_traf11);
sumtf12=sum(int_traf12);
sumtf13=sum(int_traf13);
sumtf14=sum(int_traf14);
sumtf2=sum(int_traf2);
sumtf3=sum(int_traf3);
sumtf4=sum(int_traf4);
sumtf5=sum(int_traf5);
acttf_mat=[sumtf11 sumtf12 sumtf13 sumtf14 sumtf2 sumtf3 sumtf4 sumtf5];
acttfhr_mat=acttf_mat/duration*3600;

% Primary String Generation

% Initialise population
    Chrom = crtbp(NIND, NVAR*PRECI);

% Reset counters
    Best = NaN*ones(MAXGEN,1); % best in current population
    gen = 0; % generational counter

% Accumulated Traffic
acc11=round(max_flow1/3600*15*rand);
acc12=round(max_flow1/3600*15*rand);
acc13=round(max_flow1/3600*15*rand);
acc14=round(max_flow1/3600*15*rand);
acc2=round(max_flow2/3600*15*rand);
acc3=round(max_flow3/3600*15*rand);
acc4=round(max_flow4/3600*15*rand);
acc5=round(max_flow5/3600*15*rand);
shift2=2;
shift3=2;
shift4=2;
shift5=2;

% Evaluate initial population
[sizem, sizen] = size(Chrom);
for temp1=1:sizem
    ObjV(temp1,1) = disutilitycalc(1,acc11,int_traf11,Chrom(temp1,:)) +
                       disutilitycalc(1,acc12,int_traf12,Chrom(temp1,:)) + ...
                       disutilitycalc(1,acc13,int_traf13,invert1(Chrom(temp1,:))) +
                       disutilitycalc(1,acc14,int_traf14,invert1(Chrom(temp1,:)));
end

% Track best individual and display convergence
Best(gen+1) = min(ObjV);
%  plot((Best),'ro');xlabel('generation'); ylabel('disutility');
%  text(0.5,0.95,['Best = ', num2str(Best(gen+1))],'Units','normalized');
%  drawnow;

% Generational loop
while gen < MAXGEN,

% Assign fitness-value to entire population
FitnV = ranking(ObjV);

% Select individuals for breeding
SelCh = select('sus', Chrom, FitnV, GGAP);

% Recombine selected individuals (crossover)
SelCh = recombin('xovsp',SelCh,0.7);

% Perform mutation on offspring
SelCh = mut(SelCh);

% Evaluate offspring, call objective function

[sizem1, sizen1] = size(SelCh);
for temp2=1:sizem1
    ObjVSel(temp2,1) = disutilitycalc(1,acc11,int_traf11,SelCh(temp2,:)) +
                       disutilitycalc(1,acc12,int_traf12,SelCh(temp2,:)) + ...
                       disutilitycalc(1,acc13,int_traf13,SelCh(temp2,:)) +
                       disutilitycalc(1,acc14,int_traf14,SelCh(temp2,:)) + ...

end

end

69
disutilitycalc(1,acc13,int_traf13,invert1(SelCh(temp2,:))) +
disutilitycalc(1,acc14,int_traf14,invert1(SelCh(temp2,:))); end

% Reinsert offspring into current population
[Chrom ObjV]=reins(Chrom,SelCh,1,1,ObjV,ObjVSel);

% Increment generational counter
   gen = gen+1;

% Update display and record current best individual
   Best(gen+1) = min(ObjV);
   plot((Best),'ro'); xlabel('generation'); ylabel('Center disutility');
   text(0.5,0.95,['Best = ', num2str(Best(gen+1))], 'Units','normalized');
   drawnow;
end
% End of Primary GA String
[tempu1, tempu2]=min(ObjV);
store(loop + 8,:)=Chrom(tempu2,:);

% pause;
%Begin Secondary East String Generation

NIND = 100;  % Number of individuals per subpopulations
MAXGEN = 200;  % maximum Number of generations
GGAP = .9; % Generation gap,
NVAR = 1;  % No. of variables
PRECI = 36;  % Precision of binary representation

% Initialise population
   Chromsec = crtbp(NIND, NVAR*PRECI);

% Reset counters
   Bestsec = NaN*ones(MAXGEN,1);  % best in current population
   gensec = 0;  % generational counter

% Evaluate initial population
   [sizem, sizen] = size(Chromsec);
   for temp1=1:sizem
       ObjVsec(temp1,1) = disutilitycalc2(2,acc2,int_traf2,Chromsec(temp1,:)) + ...
       disutilitycalc2(1,acc11,int_traf11(1:length(int_traf11)-shift2),invert1(Chromsec(temp1,1+shift2:length(int_traf11))));
   end
% Track best individual and display convergence
Bestsec(gensec+1) = min(ObjVsec);

% Generational loop
while gensec < MAXGEN,

% Assign fitness-value to entire population
FitnVsec = ranking(ObjVsec);

% Select individuals for breeding
SelChsec = select('sus', Chromsec, FitnVsec, GGAP);

% Recombine selected individuals (crossover)
SelChsec = recombin('xovsp',SelCh,0.7);

% Perform mutation on offspring
SelChsec = mut(SelChsec);

% Evaluate offspring, call objective function

[sizem1, sizen1] = size(SelChsec);
for temp2=1:sizem1
   ObjVSelsec(temp2,1) = disutilitycalc2(2,acc2,int_traf2,SelChsec(temp2,:)) + ...
   disutilitycalc2(1,acc11,int_traf11(1:length(int_traf11)-shift2),invert1(SelChsec(temp2,1+shift2:length(int_traf11))));
end

% Reinsert offspring into current population
[Chromsec ObjVsec]=reins(Chromsec,SelChsec,1,1,ObjVsec,ObjVSelsec);

% Increment generational counter
   gensec = gensec+1;

% Update display and record current best individual
Bestsec(gensec+1) = min(ObjVsec);

end

% End of Secondary GA String
[tempu1, tempu2]=min(ObjVsec);
store(loop + 9,:)=Chromsec(tempu2,:);
% pause;

%Begin Secondary West String Generation

NIND = 100; % Number of individuals per subpopulations
MAXGEN = 200; % maximum Number of generations
GGAP = .9; % Generation gap,
NVAR = 1; % No. of variables
PRECI = 36; % Precision of binary representation

% Initialise population
Chromsec2 = crtbp(NIND, NVAR*PRECI);

% Reset counters
Bestsec2 = NaN*ones(MAXGEN,1); % best in current population
gensec2 = 0; % generational counter

% Evaluate initial population
[sizem, sizen] = size(Chromsec2);
for temp1=1:sizem
    ObjVsec2(temp1,1) = disutilitycalc2(2,acc3,int_traf3,Chromsec2(temp1,:)) + ...
    disutilitycalc2(1,acc12,int_traf12(1:length(int_traf12)-
    shift3),invert1(Chromsec2(temp1,1+shift3:length(int_traf12))));
end

% Track best individual and display convergence
Bestsec2(gensec2+1) = min(ObjVsec2);
% plot((Bestsec2),'ro');xlabel('generation'); ylabel('East Junc. disutility');
% text(0.5,0.95,['Best = ', num2str(Bestsec2(gensec2+1))],'Units','normalized');
% drawnow;

% Generational loop
while gensec2 < MAXGEN,

    % Assign fitness-value to entire population
    FitnVsec2 = ranking(ObjVsec2);

    % Select individuals for breeding
    SelChsec2 = select('sus', Chromsec2, FitnVsec2, GGAP);

    % Recombine selected individuals (crossover)
    SelChsec2 = recombin('xovsp',SelCh,0.7);

    % Perform mutation on offspring

end
SelChsec2 = mut(SelChsec2);

% Evaluate offspring, call objective function
[sizem1, sizen1] = size(SelChsec2);
for temp2=1:sizem1
    ObjVSelec2(temp2,1) = disutilitycalc2(2,acc3,int_traf3,SelChsec2(temp2,:)) + ... 
    disutilitycalc2(1,acc12,int_traf12(1:length(int_traf12)-
    shift3),invert1(SelChsec2(temp2,1+shift3:length(int_traf12))));
end

% Reinsert offspring into current population
[Chromsec2 ObjVsec2]=reins(Chromsec2,SelChsec2,1,1,ObjVsec2,ObjVSelec2);

% Increment generational counter
    gensec2 = gensec2+1;

% Update display and record current best individual
    Bestsec2(gensec2+1) = min(ObjVsec2);
    plot((Bestsec2),',ro'); xlabel('generation'); ylabel('East Junc. disutility');
    text(0.5,0.95,['Best = ', num2str(Bestsec2(gensec2+1))],'Units','normalized');
    drawnow;
end

% End of Secondary West GA String
[tempu1, tempu2]=min(ObjVsec2);
store(loop + 10,:)=Chromsec2(tempu2,:);
% pause;

%Begin Secondary North String Generation

NIND = 100;           % Number of individuals per subpopulations
MAXGEN = 200;        % maximum Number of generations
GGAP = .9;           % Generation gap,
NVAR = 1;           % No. of variables
PRECI = 36;          % Precision of binary representation

% Initialise population
    Chromsec3 = crtbp(NIND, NVAR*PRECI);

% Reset counters
    Bestsec3 = NaN*ones(MAXGEN,1);   % best in current population
gensec3 = 0;            % generational counter

% Evaluate initial population
[sizem, sizen] = size(Chromsec3);
for temp1=1:sizem
    ObjVsec3(temp1,1) = disutilitycalc2(2,acc4,int_traf4,Chromsec3(temp1,:)) + ...
    disutilitycalc2(1,acc13,int_traf13(1:length(int_traf13)-
    shift4),invert1(Chromsec3(temp1,1+shift4:length(int_traf13))));
end

% Track best individual and display convergence
    Bestsec3(gensec3+1) = min(ObjVsec3);
    plot((Bestsec3),'ro');xlabel('generation'); ylabel('East Junc. disutility');
    text(0.5,0.95,['Best = ', num2str(Bestsec3(gensec3+1))],'Units','normalized');
    drawnow;

% Generational loop
    while gensec3 < MAXGEN,
        % Assign fitness-value to entire population
            FitnVsec3 = ranking(ObjVsec3);
        % Select individuals for breeding
            SelChsec3 = select('sus', Chromsec3, FitnVsec3, GGAP);
        % Recombine selected individuals (crossover)
            SelChsec3 = recombin('xovsp',SelCh,0.7);
        % Perform mutation on offspring
            SelChsec3 = mut(SelChsec3);
        % Evaluate offspring, call objective function
            [sizem1, sizen1] = size(SelChsec3);
            for temp2=1:sizem1
                ObjVSelsec3(temp2,1) = disutilitycalc2(2,acc4,int_traf4,SelChsec3(temp2,:)) + ...
                disutilitycalc2(1,acc13,int_traf13(1:length(int_traf13)-
                shift4),invert1(SelChsec3(temp2,1+shift4:length(int_traf13))));
            end
        % Reinsert offspring into current population
            [Chromsec3 ObjVsec3] = reins(Chromsec3,SelChsec3,1,1,ObjVsec3,ObjVSelsec3);
        % Increment generational counter
            gensec3 = gensec3+1;
        % Update display and record current best individual
            Bestsec3(gensec3+1) = min(ObjVsec3);
            plot((Bestsec3),'ro'); xlabel('generation'); ylabel('East Junc. disutility');
% End of Secondary North GA String
[tempu1, tempu2]=min(ObjVsec3);
store(loop + 11,:)=Chromsec3(tempu2,:);
% pause;

%Begin Secondary South String Generation
NIND = 100;           % Number of individuals per subpopulations
MAXGEN = 200;        % maximum Number of generations
GGAP = .9;           % Generation gap,
NVAR = 1;           % No. of variables
PRECI = 36;          % Precision of binary representation

% Initialise population
Chromsec4 = crtbp(NIND, NVAR*PRECI);

% Reset counters
Bestsec4 = NaN*ones(MAXGEN,1);  % best in current population
gensec4 = 0;   % generational counter

% Evaluate initial population
[sizem, sizen] = size(Chromsec4);
for temp1=1:sizem
    ObjVsec4(temp1,1) = disutilitycalc2(2,acc5,int_traf5,Chromsec4(temp1,:)) + ...
                       disutilitycalc2(1,acc14,int_traf14(1:length(int_traf14)-
                           shift5),invert1(Chromsec4(temp1,1+shift5:length(int_traf14))));
end

% Track best individual and display convergence
Bestsec4(gensec4+1) = min(ObjVsec4);
%     plot((Bestsec4),'ro');xlabel('generation'); ylabel('East Junc. disutility');
%     text(0.5,0.95,['Best = ', num2str(Bestsec4(gensec4+1))],'Units','normalized');
%     drawnow;

% Generational loop
while gensec4 < MAXGEN,

    % Assign fitness-value to entire population
    FitnVsec4 = ranking(ObjVsec4);

    % Select individuals for breeding
SelChsec4 = select(‘sus’, Chromsec4, FitnVsec4, GGAP);

% Recombine selected individuals (crossover)
SelChsec4 = recombin(‘xovsp’, SelCh, 0.7);

% Perform mutation on offspring
SelChsec4 = mut(SelChsec4);

% Evaluate offspring, call objective function
[sizem1, sizen1] = size(SelChsec4);
for temp2 = 1:sizem1
    ObjVSelsec4(temp2,1) = disutilitycalc2(2,acc5,int_traf5,SelChsec4(temp2,:)) + ... 
    disutilitycalc2(1,acc14,int_traf14(1:length(int_traf14)-shift5),invert1(SelChsec4(temp2,1+shift5:length(int_traf14))));
end

% Reinsert offspring into current population
[Chromsec4 ObjVsec4] = reins(Chromsec4, SelChsec4, 1, 1, ObjVsec4, ObjVSelsec4);

% Increment generational counter
gensec4 = gensec4+1;

% Update display and record current best individual
Bestsec4(gensec4+1) = min(ObjVsec4);
plot((Bestsec4),’ro’); xlabel(‘generation’); ylabel(‘East Junc. disutility’);

end

% End of Secondary South GA String
[tempu1, tempu2]=min(ObjVsec4);
store(loop + 12,:)=Chromsec4(tempu2,:);
save storeval.txt store -ASCII
end

trafgen1.m

function temp2 = trafgen1 (base, PRECI)
for i = 1:PRECI
    temp2(i)=max(0,floor(rand*base));
end

% CRTP.m - Create an initial population
%
function [Chrom, Lind, BaseV] = crtbp(Nind, Lind, Base)
nargs = nargin;

% Check parameter consistency
if nargs >= 1, [mN, nN] = size(Nind); end
if nargs >= 2, [mL, nL] = size(Lind); end
if nargs == 3, [mB, nB] = size(Base); end

if nN == 2
    if (nargs == 1)
        Lind = Nind(2); Nind = Nind(1); BaseV = crtbase(Lind);
    elseif (nargs == 2 & nL == 1)
        BaseV = crtbase(Nind(2),Lind); Lind = Nind(2); Nind = Nind(1);
    elseif (nargs == 2 & nL > 1)
        if Lind ~= length(Lind), error('Lind and Base disagree'); end
        BaseV = Lind; Lind = Nind(2); Nind = Nind(1);
    end
else if nN == 1
    if nargs == 2
        if nL == 1, BaseV = crtbase(Lind);
        else, BaseV = Lind; Lind = nL; end
    elseif nargs == 3
        if nB == 1, BaseV = crtbase(Lind,Base);
        elseif nB ~= Lind, error('Lind and Base disagree');
        else BaseV = Base; end
    end
else
    error('Input parameters inconsistent');
end

Chrom = floor(rand(Nind,Lind).*BaseV(ones(Nind,1),:));

% End of file

invert1.m

function temp3 = invert1 (temp2)
temp3=temp2;
for i=1:length(temp2)
    if(temp2(i)==0)
        temp3(i)=1;
    else
        temp3(i)=0;
    end
function FitnV = ranking(ObjV, RFun, SUBPOP);

% Identify the vector size (Nind)
[Nind,ans] = size(ObjV);

if nargin < 2, RFun = []; end
if nargin > 1, if isnan(RFun), RFun = []; end, end
if prod(size(RFun)) == 2,
    if RFun(2) == 1, NonLin = 1;
    elseif RFun(2) == 0, NonLin = 0;
    else error('Parameter for ranking method must be 0 or 1'); end
    RFun = RFun(1);
    if isnan(RFun), RFun = 2; end
else prod(size(RFun)) > 2,
    if prod(size(RFun)) ~= Nind, error('ObjV and RFun disagree'); end
end

if nargin < 3, SUBPOP = 1; end
if nargin > 2,
    if isempty(SUBPOP), SUBPOP = 1;
    elseif isnan(SUBPOP), SUBPOP = 1;
    elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
end
if (Nind/SUBPOP) ~= fix(Nind/SUBPOP), error('ObjV and SUBPOP disagree'); end
Nind = Nind/SUBPOP;  % Compute number of individuals per subpopulation

% Check ranking function and use default values if necessary
if isempty(RFun),
    % linear ranking with selective pressure 2
    RFun = 2*[0:Nind-1]’/(Nind-1);
elseif prod(size(RFun)) == 1
    if NonLin == 1,
        % non-linear ranking
        if RFun(1) < 1, error('Selective pressure must be greater than 1');
        elseif RFun(1) > Nind-2, error('Selective pressure too big'); end
        Root1 = roots([RFun(1)-Nind [RFun(1)*ones(1,Nind-1)]]);
        RFun = (abs(Root1(1)) * ones(Nind,1)) .^ (0:Nind-1)’;
        RFun = RFun / sum(RFun) * Nind;
    else
        % linear ranking with SP between 1 and 2
        if (RFun(1) < 1 | RFun(1) > 2),
            error('Selective pressure for linear ranking must be between 1 and 2');
        end
    end
end
RFun = 2-RFun + 2*(RFun-1)*[0:Nind-1]’/(Nind-1);
end
end;

FitnV = [];

% loop over all subpopulations
for irun = 1:SUBPOP,
% Copy objective values of actual subpopulation
  ObjVSub = ObjV((irun-1)*Nind+1:irun*Nind);
% Sort does not handle NaN values as required. So, find those...
  NaNix = isnan(ObjVSub);
  Validix = find(~NaNix);
  % ... and sort only numeric values (smaller is better).
  [ans,ix] = sort(-ObjVSub(Validix));

  % Now build indexing vector assuming NaN are worse than numbers,
  % (including Inf!)...
  ix = [find(NaNix) ; Validix(ix)];
  % ... and obtain a sorted version of ObjV
  Sorted = ObjVSub(ix);

  % Assign fitness according to RFun.
  i = 1;
  FitnVSub = zeros(Nind,1);
  for j = [find(Sorted(1:Nind-1) ~= Sorted(2:Nind)); Nind],
    FitnVSub(i:j) = sum(RFun(i:j)) * ones(j-i+1,1) / (j-i+1);
    i =j+1;
  end

  % Finally, return unsorted vector.
  [ans,uix] = sort(ix);
  FitnVSub = FitnVSub(uix);

  % Add FitnVSub to FitnV
  FitnV = [FitnV; FitnVSub];
end

% End of function

% SELECT.M          (universal SELECTion)
%
function SelCh = select(SEL_F, Chrom, FitnV, GGAP, SUBPOP);
% Check parameter consistency
  if nargin < 3, error('Not enough input parameter'); end

% Identify the population size (Nind)
[NindCh,Nvar] = size(Chrom);
[NindF,VarF] = size(FitnV);
if NindCh ~= NindF, error('Chrom and FitnV disagree'); end
if VarF ~= 1, error('FitnV must be a column vector'); end

if nargin < 5, SUBPOP = 1; end
if nargin > 4,
  if isempty(SUBPOP), SUBPOP = 1;
  elseif isnan(SUBPOP), SUBPOP = 1;
  elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
end

if (NindCh/SUBPOP) ~= fix(NindCh/SUBPOP), error('Chrom and SUBPOP disagree');
end
Nind = NindCh/SUBPOP;  % Compute number of individuals per subpopulation

if nargin < 4, GGAP = 1; end
if nargin > 3,
  if isempty(GGAP), GGAP = 1;
  elseif isnan(GGAP), GGAP = 1;
  elseif length(GGAP) ~= 1, error('GGAP must be a scalar');
  elseif (GGAP < 0), error('GGAP must be a scalar bigger than 0'); end
end

% Compute number of new individuals (to select)
  NSel=max(floor(Nind*GGAP+.5),2);

% Select individuals from population
  SelCh = [];
  for irun = 1:SUBPOP,
    FitnVSub = FitnV((irun-1)*Nind+1:irun*Nind);
    ChrIx=feval(SEL_F, FitnVSub, NSel)+(irun-1)*Nind;
    SelCh=[SelCh; Chrom(ChrIx,:)];
  end

% End of function

% RECOMBIN.M (RECOMBINation high-level function)
function NewChrom = recombin(REC_F, Chrom, RecOpt, SUBPOP);

% Check parameter consistency
if nargin < 2, error('Not enough input parameter'); end

% Identify the population size (Nind)
[Nind,Nvar] = size(Chrom);

if nargin < 4, SUBPOP = 1; end
if nargin > 3,
    if isempty(SUBPOP), SUBPOP = 1;
    elseif isnan(SUBPOP), SUBPOP = 1;
    elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
end

if (Nind/SUBPOP) ~= fix(Nind/SUBPOP), error('Chrom and SUBPOP disagree'); end
Nind = Nind/SUBPOP; % Compute number of individuals per subpopulation

if nargin < 3, RecOpt = 0.7; end
if nargin > 2,
    if isempty(RecOpt), RecOpt = 0.7;
    elseif isnan(RecOpt), RecOpt = 0.7;
    elseif length(RecOpt) ~= 1, error('RecOpt must be a scalar');
    elseif (RecOpt < 0 | RecOpt > 1), error('RecOpt must be a scalar in [0, 1]'); end
end

% Select individuals of one subpopulation and call low level function
NewChrom = [];
for irun = 1:SUBPOP,
    ChromSub = Chrom((irun-1)*Nind+1:irun*Nind,:);
    NewChromSub = feval(REC_F, ChromSub, RecOpt);
    NewChrom=[NewChrom; NewChromSub];
end

% End of function

%MUT.m
function NewChrom = mut(OldChrom,Pm,BaseV)

% get population size (Nind) and chromosome length (Lind)
[Nind, Lind] = size(OldChrom);
% check input parameters
if nargin < 2, Pm = 0.7/Lind ; end
if isnan(Pm), Pm = 0.7/Lind; end

if (nargin < 3), BaseV = crtbase(Lind); end
if (isnan(BaseV)), BaseV = crtbase(Lind); end
if (isempty(BaseV)), BaseV = crtbase(Lind); end

if (nargin == 3) & (Lind ~= length(BaseV))
    error('OldChrom and BaseV are incompatible'), end

% create mutation mask matrix
BaseM = BaseV(ones(Nind,1),:);

% perform mutation on chromosome structure
NewChrom = rem(OldChrom+(rand(Nind,Lind)<Pm).*ceil(rand(Nind,Lind).*(BaseM-
1)),BaseM);

disutilitycalc2.m

function disutility1 = disutilitycalc2 (type, acctraffic, trafseq, phseq)
act=sum(acctraffic);
disutility1=0;
i=1;
while(i<length(trafseq))
    if ((phseq(i))==1)
        m=0;
        j=i;
        count = 0;
        while (~m & j<length(trafseq))
            if ((phseq(j))==1)
                count = count + 1;
            else
                m=1;
            end
            j=j+1;
        end
        start = i;
        last = i+count;
        prog = vehint(type, last-start);
        newtraf = sum(trafseq(start:last));
        act = max(0, act + newtraf - prog);
        disutility1 = disutility1 + (act-prog)*5 + act;
        i=last;
    end
end
else
    m=0;
    j=i;
    count = 0;
    while (~m & j<length(trafseq))
        if ((phseq(j))==0)
            count = count + 1;
        else
            m=1;
        end
        j=j+1;
    end
    start = i;
    last = i+count;
    prog = 0;
    newtraf = sum(trafseq(start:last));
    act = max(0, act + newtraf - prog);
    disutility1 = disutility1 + (act-prog)*5 + act;
    i=last;
end
i=i+1;
end

% REINS.M       (RE-INSertion of offspring in population replacing parents)

function [Chrom, ObjVCh] = reins(Chrom, SelCh, SUBPOP, InsOpt, ObjVCh, ObjVSel);

% Check parameter consistency
if nargin < 2, error('Not enough input parameter'); end
if (nargout == 2 & nargin < 6), error('Input parameter missing: ObjVCh and/or ObjVSel'); end

[NindP, NvarP] = size(Chrom);
[NindO, NvarO] = size(SelCh);

if nargin == 2, SUBPOP = 1; end
if nargin > 2,
    if isempty(SUBPOP), SUBPOP = 1;
    elseif isnan(SUBPOP), SUBPOP = 1;
    elseif length(SUBPOP) ~= 1, error('SUBPOP must be a scalar'); end
end

if (NindP/SUBPOP) ~= fix(NindP/SUBPOP), error('Chrom and SUBPOP disagree'); end
end
if (NindO/SUBPOP) ~= fix(NindO/SUBPOP), error('SelCh and SUBPOP disagree');
end
NIND = NindP/SUBPOP;  % Compute number of individuals per subpopulation
NSEL = NindO/SUBPOP;  % Compute number of offspring per subpopulation

IsObjVCh = 0; IsObjVSel = 0;
if nargin > 4,
    [mO, nO] = size(ObjVCh);
    if nO ~= 1, error('ObjVCh must be a column vector'); end
    if NindP ~= mO, error('Chrom and ObjVCh disagree'); end
    IsObjVCh = 1;
end
if nargin > 5,
    [mO, nO] = size(ObjVSel);
    if nO ~= 1, error('ObjVSel must be a column vector'); end
    if NindO ~= mO, error('SelCh and ObjVSel disagree'); end
    IsObjVSel = 1;
end
if nargin < 4, INSR = 1.0; Select = 0; end
if nargin >= 4,
    if isempty(InsOpt), INSR = 1.0; Select = 0;
    elseif isnan(InsOpt), INSR = 1.0; Select = 0;
    else
        INSR = NaN; Select = NaN;
        if (length(InsOpt) > 2), error('Parameter InsOpt too long'); end
        if (length(InsOpt) >= 1), Select = InsOpt(1); end
        if (length(InsOpt) >= 2), INSR = InsOpt(2); end
        if isnan(Select), Select = 0; end
        if isnan(INSR), INSR =1.0; end
    end
end
if (INSR < 0 | INSR > 1), error('Parameter for insertion rate must be a scalar in [0, 1]');
end
if (INSR < 1 & IsObjVSel ~= 1), error('For selection of offspring ObjVSel is needed');
end
if (Select ~= 0 & Select ~= 1), error('Parameter for selection method must be 0 or 1');
end
if (Select == 1 & IsObjVCh == 0), error('ObjVCh for fitness-based exchange needed');
end
if INSR == 0, return; end
NIns = min(max(floor(INSR*NSEL+.5),1),NIND);   % Number of offspring to insert

% perform insertion for each subpopulation
for irun = 1:SUBPOP,
    % Calculate positions in old subpopulation, where offspring are inserted
    if Select == 1,  % fitness-based reinsertion
        [Dummy, ChIx] = sort(-ObjVCh((irun-1)*NIND+1:irun*NIND));
    else             % uniform reinsertion
        [Dummy, ChIx] = sort(rand(NIND,1));
    end
    PopIx = ChIx((1:NIns)')+ (irun-1)*NIND;
    % Calculate position of Nins-% best offspring
    if (NIns < NSEL),  % select best offspring
        [Dummy,OffIx] = sort(ObjVSel((irun-1)*NSEL+1:irun*NSEL));
    else
        OffIx = (1:NIns)';
    end
    SelIx = OffIx((1:NIns)')+(irun-1)*NSEL;
    % Insert offspring in subpopulation -> new subpopulation
    Chrom(PopIx,:) = SelCh(SelIx,:);
    if (IsObjVCh == 1 & IsObjVSel == 1), ObjVCh(PopIx) = ObjVSel(SelIx); end
end

% End of function