KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY, KUMASI

TRAFFIC SEQUENCING WITH GENETIC ALGORITHM

CASE STUDY: LA PAZ TRAFFIC INTERSECTION

By

FESTUS EYRAM KWASHIE ASHIGBIE

(INDEX NUMBER: PG 8143012)

A THESIS SUBMITTED TO THE DEPARTMENT OF MATHEMATICS, KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY IN PARTIAL FULFILMENT OF THE REQUIREMENT FOR THE DEGREE OF MSC INDUSTRIAL MATHEMATICS WITH THESIS

March 23, 2016

Declaration

I hereby declare that this submission is my own work towards the award of the MSc. degree and that, to the best of my knowledge, it contains no material previously published by another person nor material which had been accepted for the award of any other degree of the university, except where due acknowledgement had been made in the text.

FESTUS EYRAM KWASHIE ASHIGBIE	.....	.....
Student	Signature	Date

Certified by:

Dr. JOSEPH ACKORA-PRAH	.....	.....
Supervisor	Signature	Date

Certified by:

Prof. S. K. AMPONSAH	.....	.....
Head of Department	Signature	Date

Dedication

I dedicate this thesis to:

My Son Timothy

My Wife Theodora

My Mum Veronica

My Dad Livingstone

Abstract

Managing traffic flow at complex intersections is a challenging task. The intersection under study is situated at La Paz on the N1 highway. One can see wreckages of crushed vehicles at different locations of the highway. Within one month of its commissioning, the road registered fifteen (15) accidents with huge loses to life and property. Between February 2012 and December 2013, 489 accidents had been recorded on the road with 350 people sustaining severe injuries. Two hundred (200) pedestrians had also been knocked down by speeding cars while 70 pedestrians died on the spot. These accidents occur when pedestrians attempt to cross the road. Some identified causes of these accidents are inefficient traffic order sequence, inadequate green time allocation, and non-user friendly traffic installations. The breadth of the highway is about 50 metres with no islands for pedestrians use. There are no overhead footbridge or underground passage at the intersection for pedestrian use. These causes lots of chaos at the intersection and leads to road block and heavy traffic during rush hours. To save lives and properties, and improve traffic flow, the intersection is investigated to propose new ways of improving pedestrian safety and traffic flow sequence. Genetic Algorithm approach was used to find an optimal traffic flow sequence for the intersection. The intersection was simplified into a 'plus' intersection that has traffic streams moving in fourteen directions. The streams were encoded and put through the genetic algorithm process. The output was evaluated for both computational efficiency and accuracy. The program returned a flow sequence that is very efficient. It gives hope that soon, the chaos and accidents at the intersection will become history.

Acknowledgements

I wish to thank my supervisor, Dr. Joseph Ackora-Prah for guiding me in doing this thesis. His expertise, precious time, and willingness to provide feedback made the completion of this research an enjoyable experience. I owe a lot of thanks to all the lecturers and colleagues of the Mathematics department for their leadership throughout my study.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background of Study

The increase of traffic in the city of Accra is causing significant problems. The current road network cannot keep up with traffic growth. One may think of massive expansion of the road network as a solution to the problem. This solution of providing more streets is limited by economic and ecological factors and thus it is necessary to optimize traffic movements on the existing street network.

Firstly, we may approach this problem by Optimizing the network: New roads could be built and existing roads expanded, use of roads could be changed,clever and better routing introduced, or smoothing traffic flow by a system of coupled intersection signal operations.

We may also tackle the problem by Optimizing intersections: Efficient sequencing of traffic movements at intersections and improving traffic signal operation. The traffic signal assigns the right-of-way to a stream of vehicles and pedestrians in other to guarantee their safe crossings. The specification of the signal setting (traffic control strategy) has a major impact on intersection capacity and efficiency. Since intersection parameters are local, an intersection optimization can be based on local traffic attributes. As described in Schnabel and Lohse (1997), intersections mainly determine the capacity of urban traffic networks. Therefore optimizing the intersection is a quick and efficient way to cope with the characteristic problems of urban traffic. When permeability of intersections are improved, it leads to lower traffic densities and reduced travel times.

The work presents a model to improve sequencing at an intersection. The aim is to determine compatible groups of traffic flow or the order in which each phase of traffic moves to enhance traffic flow and reduce collision.

The intersection under consideration is situated at La Paz on the N1 highway. One can see wreckages of crushed vehicles at different locations of the motorway. Within one month of commissioning, the road registered fifteen (15) accidents with huge loses to life and property. There have also been reports of various degrees of pedestrian knock downs.

Some people argue that the accidents on the highway are due to the absence of U-turns and zebra crossings at vantage points on the road. The Association of Persons with Disabilities have also complained about the absence of facilities on the overhead foot bridges that they should have been using to cross the N1 Highway. The explanation given by the Millennium Challenge Account (MCA) for this phenomenon was that the foot bridges were too high to make room for such facilities. This explanation is a very strange one.The MCA should have made good our laws that mandates the provision of these disability friendly facilities on all public structures and institutions. What it means is that persons with disabilities would have to go to the daily hustle of moving to the nearest traffic light junction or crossing to enable them move across the broad road.

Another obvious risk on the road is the absence of reflectors at the Dimples Roundabout. Ideally, there should be reflectors on the aluminium rails at the roundabout. It is very difficult or almost impossible for a driver in a fast moving vehicle that might be using the road for the first time, to notice that sharp roundabout from a distance as the aluminium rails do not have signals to warn drivers of the hazard. It is therefore urgent for the Road Ministry to ensure that reflectors are posted on the rails to guide drivers and motorists alike to be forewarned of the hazards posed by that roundabout.

Drivers seem to have thrown away the rule that one cannot exceed the 50km/h speed limit imposed on drivers on roads passing through towns and cities. Even though the N1 Highway is an extension of the Kwame Nkrumah Motorway, the highway passes through busy communities towards Mallam Junction and drivers must not be allowed to use the highway as a race course. The imposition and

respect for this speed limit will enable any driver to have a good stopping and breaking distance even if met unexpectedly by any hazards on the road. The situation even becomes safer for long trucks and cargo vehicles that seem to have braking problems and longer stopping distance. Enforcing this speed limit will largely curtail the unnecessary accidents on the George Walker Bush N1 Highway. Another factor worth mentioning is the attitude of pedestrians on the road. It is not surprising that many pedestrians choose to jump over the short rails separating the speedways instead of using the overhead bridges and to the pelican crossings on the highway. Life seems to have suddenly changed for the people using the road all day to cross over to the other side of the road on transit. Many of them are uncomfortable about walking all the way to the overhead bridge or the traffic lights to have safe crossing to the other side of the road. It is amazing that pedestrians would take these risks for granted and opt to jump over the rails rather than use the approved areas and they should have themselves to blame if they get knocked down in the process that is if they are lucky enough to live to see another day. The use of mobile phones while driving or crossing the road at unapproved sections is another matter that must attract the attention of law makers.

As regards road signs and traffic regulations, the population is highly ignorant of what these road signs are. There are a number of "No Stopping for Any Reason" signs along the highway which are usually ignored.

(Amegashie, 2012)

1.2 Problem Statement

The N1 highway can be labelled as a good asset that has become a death trap. Many accidents occur on this highway and 80 per cent happens at the traffic intersection. Between February 2012 and December 2013, 489 accidents had been recorded on the road with 350 people sustaining severe injuries. Two

hundred (200) pedestrians had also been knocked down by speeding cars while 70 pedestrians died on the spot. These accidents occur when pedestrians attempt to cross the road. There have also been several car crashes. Some identified causes of these accidents are inefficient traffic order sequence, inadequate green time allocation, and non-user friendly traffic installations. The breadth of the highway is about 50 metres with no islands for pedestrians use. There are no overhead footbridge or underground passage at the intersection for pedestrian use. To save lives and properties, and improve traffic flow, the intersection is investigated to propose new ways of improving pedestrian safety and traffic flow sequence.

1.3 Objectives of the study

Traffic sequencing is usually a trial-and-error process. Based on heuristics and expert knowledge, the traffic engineer proposes a signal operation and tests its effectiveness by means of simulations. He may test one or two other configurations, but due to the duration of this process he has no possibility for exuberant experiments. This design process usually yields reasonable results. But it has a significant drawback. The design relies heavily on individual experience. Solutions beyond the traditional way usually will not be found, even though special problems often need special answers.

The objectives of this study is to use Genetic Algorithm (GA) to find an efficient vehicle passing sequence to maximize the traffic throughput at the La Paz intersections.

1.4 Scope of Study

Profile of Study Area

The study will be carried out on the La Paz traffic light intersection situated on the George Walker Bush Highway. The George Walker Bush Highway is a six-lane, 14-kilometre (8.7 mile) highway in Accra in the Greater Accra Region of Ghana. It is also known as the Mallam-Tetteh Quashie Highway. It links the urban town of Mallam to the Tetteh Quarshie Interchange and was officially opened to motorists on 15th February, 2012. It is known also as the N1 highway. The N1 is among the busiest roads in Accra as it links the Greater Accra region with the Central Region. Besides, it carries a lot of human traffic of which most is concentrated at La Paz. La Paz, before the construction of the N1 Highway was one community. The construction of the highway split the community into two. Relatives, friends, businesses were split, one on one side of the road, the other at the other side of the road. At La Paz, the highway was joined by two roads forming a four way junction. La Paz is a major business hub of Accra. It houses most of the well-established financial institutions, businesses, churches, health facilities and religious buildings. This makes the La Paz experience massive human traffic and hence much crossing at the interchange.

1.5 Justification

The traffic congestion at La Paz intersection is a daily occurrence. It is usually caused by the inability of the road network to cope with the volume of vehicular traffic using it. This results in loss of money and time and affects the economy adversely. Commuters have to spend longer travel time to and from work. Vehicles have to burn more fuel. The environment is also affected due to the heavy emission of gases from burning of gasoline. We will need more inter-urban roads to improve accessibility, connectivity and development. However,the desire to

build the roads to meet the demand for motor vehicles is unlikely to solve congestion and other traffic problems. In fact, the alternative highway capacity is likely to be

'choked' and the benefits offset unless the growth in traffic volume is restrained. Instead of looking only at road expansion, we are looking at optimising traffic flow at intersections as one of the ways to reduce road congestion.

1.6 Methodology

Traffic flow on the N1 highway has become more saturated. The traffic flow sequence at the La Paz intersection possesses danger to road users and need to be improved. Genetic Algorithm approach with MATLAB codes will be used in this investigation.

Chapter 2

Literature Review

In the 1950s and the 1960s several computer scientists independently studied evolutionary systems with the idea that evolution could be used as an optimization tool for engineering problems. The idea in all these systems was to evolve a population of candidate solutions to a given problem, using operators inspired by natural genetic variation and natural selection.

In the 1960s, Rechenberg (1965, 1973) introduced ”evolution strategies”, a method he used to optimize real-valued parameters for devices such as airfoils.. This idea was further developed by Schwefel (1975, 1977). The field of evolution strategies has remained an active area of research, mostly developing independently from the field of genetic algorithms. The two communities have begun to interact in recent times. Fogel and Walsh (1966) developed ”evolutionary programming”, a technique in which candidate solutions to given tasks were represented as finite-state machines, which were evolved by randomly mutating their state-transition diagrams and selecting the fittest. A somewhat broader formulation of evolutionary programming also remains an area of active research. Together, evolution strategies, evolutionary programming, and genetic algorithms form the backbone of the field of evolutionary computation.

Several other people working in the 1950s and the 1960s developed evolutioninspired algorithms for optimization and machine learning. Box (1957); Friedman (1959); Bledsoe (1961); Bremermann (1962); Reed and Barricelli (1967) all worked in this area, though their work has been given little or none of the kind of attention or follow up that evolution strategies, evolutionary programming, and genetic algorithms have seen. In addition, a number of evolutionary biologists used computers to simulate evolution for the purpose of controlled experiments (Baricelli, 1957, 1962; Fraser, 1957b,a; Martin and

Cockerham, 1960). Evolutionary computation was definitely in the air in the formative days of the electronic computer.

Genetic algorithms (GAs) were invented by John Holland in the 1960s and were developed by Holland and his students and colleagues at the University of Michigan in the 1960s and the 1970s. In contrast with evolution strategies and evolutionary programming, Holland's original goal was not to design algorithms to solve specific problems, but rather to formally study the phenomenon of adaptation as it occurs in nature and to develop ways in which the mechanisms of natural adaptation might be imported into computer systems. Holland's 1975 book Adaptation in Natural and Artificial Systems presented the genetic algorithm as an abstraction of biological evolution and gave a theoretical framework for adaptation under the GA. Holland's GA is a method for moving from one population of "chromosomes" (e.g., strings of ones and zeros, or "bits") to a new population by using a kind of "natural selection" together with the genetics-inspired operators of crossover, mutation, and inversion. Each chromosome consists of "genes" (e.g., bits), each gene being an instance of a particular "allele" (e.g., 0 or 1). The selection operator chooses those chromosomes in the population that will be allowed to reproduce, and on average the fitter chromosomes produce more offspring than the less fit ones. Crossover exchanges subparts of two chromosomes, roughly mimicking biological recombination between two single-chromosome ("haploid") organisms; mutation randomly changes the allele values of some locations in the chromosome; and inversion reverses the order of a contiguous section of the chromosome, thus rearranging the order in which genes are arrayed. (Here, as in most of the GA literature, "crossover" and "recombination" will mean the same thing.) Holland's introduction of a population-based algorithm with crossover, inversion, and mutation was a major innovation. (Rechenberg's evolution strategies started with a "population" of two individuals, one parent and one offspring, the offspring being a mutated version of the parent; many-individual populations and

crossover were not incorporated until later. Fogel, Owens, and Walsh's evolutionary programming likewise used only mutation to provide variation.) Moreover, Holland (1975) was the first to attempt to put computational evolution on a firm theoretical footing. Until recently this theoretical foundation, based on the notion of "schemas," was the basis of almost all subsequent theoretical work on genetic algorithms In the last several years there has been widespread interaction among researchers studying various evolutionary computation methods, and the boundaries between GAs, evolution strategies, evolutionary programming, and other evolutionary approaches have broken down to some extent. Today, researchers often use the term "genetic algorithm" to describe something very far from Holland's original conception. In this book I adopt this flexibility. Most of the projects I will describe here were referred to by their originators as GAs; some were not, but they all have enough of a "family resemblance" that I include them under the rubric of genetic algorithms.

GAs was used to Evolve Strategies for the Prisoner's Dilemma. The Prisoner's Dilemma, a simple two-person game invented by Merrill Flood and Melvin Dresher in the 1950s, has been studied extensively in game theory, economics, and political science because it can be seen as an idealized model for real-world phenomena such as arms races (Axelrod and Kaufmann, 1987; Axelrod and Dion, 1988). It can be formulated as follows: Two individuals (call them Alice and Bob) are arrested for committing a crime together and are held in separate cells, with no communication possible between them. Alice is offered the following deal: If she confesses and agrees to testify against Bob, she will receive a suspended sentence with probation, and Bob will be put away for 5 years. However, if at the same time Bob confesses and agrees to testify against Alice, her testimony will be discredited, and each will receive 4 years for pleading guilty. Alice is told that Bob is being offered precisely the same deal. Both Alice and Bob know that if neither testify against the other they can be convicted only on a lesser charge for which they will each get 2 years in jail. Should Alice "defect" against Bob and hope for

the suspended sentence, risking a 4-year sentence if Bob defects? Or should she ”cooperate” with Bob (even though they cannot communicate), in the hope that he will also cooperate so each will get only 2 years, thereby risking a defection by Bob that will send her away for 5 years? The game can be described more abstractly. Each player independently decides which move to make-i.e., whether to cooperate or defect. A ”game” consists of each player’s making a decision (a ”move”). The possible results of a single game are summarized in a payoff matrix like the one shown in Figure 2.1. Here the goal is to get as many points (as opposed to as few years in prison) as possible. (In Figure 2.1, the payoff in each case can be interpreted as 5 minus the number of years in prison.) If both players cooperate, each gets 3 points. If player A defects and player B cooperates, then player A gets 5 points and player B gets 0 points, and vice versa if the situation is reversed. If both players defect, each gets 1 point. What is the best strategy to use in order to maximize one’s own payoff? If you suspect that your opponent is going to cooperate, then you should surely defect. If you suspect that your opponent is going to defect, then you should defect too. No matter what the other player does, it is always better to defect. The dilemma is that if both players defect each gets a worse score than if they cooperate. If the game is iterated (that is, if the two players play several games in a row), both players’ always defecting will lead to a much lower total payoff than the players would get if they cooperated. How can reciprocal cooperation be induced? This question takes on special significance when the notions of cooperating and defecting correspond to actions in, say, a real-world arms race (e.g., reducing or increasing one’s arsenal).

Robert Axelrod (1984) of the University of Michigan has studied the Prisoner’s Dilemma and related games extensively. His interest in determining what makes for a good strategy led him to organize two Prisoner’s Dilemma tournaments. He solicited strategies from researchers in a number of disciplines. Each participant submitted a computer program that implemented a particular

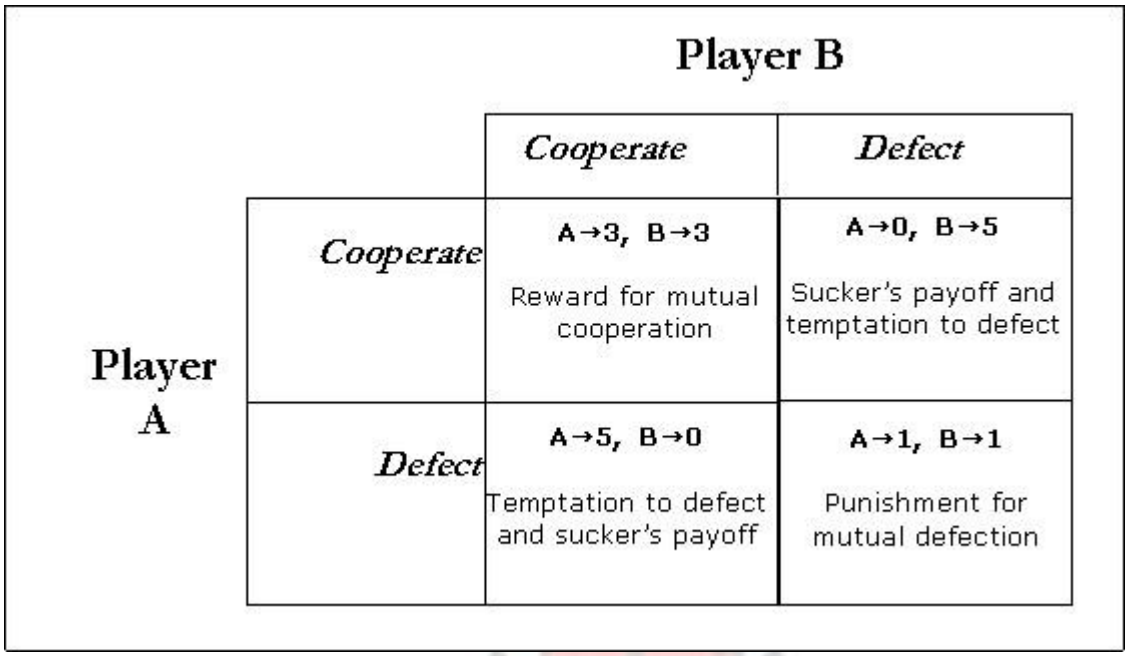


Figure 2.1: The payoff matrix for the Prisoner's Dilemma

strategy, and the various programs played iterated games with each other. During each game, each program remembered what move (i.e., cooperate or defect) both it and its opponent had made in each of the three previous games that they had played with each other, and its strategy was based on this memory. The programs were paired in a round-robin tournament in which each played with all the other programs over a number of games. The first tournament consisted of 14 different programs; the second consisted of 63 programs (including one that made random moves). Some of the strategies submitted were rather complicated, using techniques such as Markov processes and Bayesian inference to model the other players in order to determine the best move. However, in both tournaments the winner (the strategy with the highest average score) was the simplest of the submitted strategies: TIT FOR TAT. This strategy, submitted by Anatol Rapoport, cooperates in the first game and then, in subsequent games, does whatever the other player did in its move in the previous game with TIT FOR TAT. That is, it offers cooperation and reciprocates it. But if the other player defects, TIT FOR TAT punishes that defection with a defection of its own, and continues the punishment until the other player begins cooperating again. After the two tournaments,

Axelrod (1984) decided to see if a GA could evolve strategies to play this game successfully. The first issue was figuring out how to encode a strategy as a string. Here is how Axelrod's encoding worked. Suppose the memory of each player is one previous game. There are four possibilities for the previous game; CC (case 1), CD (case 2), DC (case 3), and DD (case 4). C denotes "cooperate" and D denotes "defect". Case 1 is when both players cooperated in the previous game, case 2 is when player A cooperated and player B defected, and so on. A strategy is simply a rule that specifies an action in each of these cases. For example, TIT FOR TAT as played by player A is as follows:

- If CC (case 1), then C.
- If CD (case 2), then D. • If DC (case 3), then C.
- If DD (case 4), then D.

If the cases are ordered in this canonical way, this strategy can be expressed compactly as the string CDCD. To use the string as a strategy, the player records the moves made in the previous game (e.g., CD), finds the case number i by looking up that case in a table of ordered cases like that given above (for CD, i = 2), and selects the letter in the ith position of the string as its move in the next game (for i = 2, the move is D).

Axelrod's tournaments involved strategies that remembered three previous games.

There are 64 possibilities for the previous three games:

- CC CC CC (case 1),
- CC CC CD (case 2),
- CC CC DC (case 3),
- .....

- DD DD DC (case 63),
- DD DD DD (case 64).

Thus, a strategy can be encoded by a 64-letter string. Since using the strategy requires the results of the three previous games, Axelrod actually used a 70-letter string, where the six extra letters encoded three hypothetical previous games used by the strategy to decide how to move in the first actual game. Since each locus in the string has two possible alleles (C and D), the number of possible strategies is 270. The search space is thus far too big to be searched exhaustively. In Axelrod's first experiment, the GA had a population of 20 such strategies. The fitness of a strategy in the population was determined as follows: Axelrod had found that eight of the human-generated strategies from the second tournament were representative of the entire set of strategies, in the sense that a given strategy's score playing with these eight was a good predictor of the strategy's score playing with all 63 entries. This set of eight strategies (which did not include TIT FOR TAT) served as the "environment" for the evolving strategies in the population. Each individual in the population played iterated games with each of the eight fixed strategies, and the individual's fitness was taken to be its average score over all the games it played. Axelrod performed 40 different runs of 50 generations each, using different random-number seeds for each run. Most of the strategies that evolved were similar to TIT FOR TAT in that they reciprocated cooperation and punished defection (although not necessarily only on the basis of the immediately preceding move). However, the GA often found strategies that scored substantially higher than TIT FOR TAT. This is a striking result, especially in view of the fact that in a given run the GA is testing only 20 * 50 = 1000 individuals out of a huge search space of 270 possible individuals.

It would be wrong to conclude that the GA discovered strategies that are "better" than any human-designed strategy. The performance of a strategy depends very much on its environment-that is, on the strategies with which it is playing. Here the environment was fixed-it consisted of eight human-designed strategies that

did not change over the course of a run. The resulting fitness function is an example of a static (unchanging) fitness landscape. The highest-scoring strategies produced by the GA were designed to exploit specific weaknesses of several of the eight fixed strategies. It is not necessarily true that these high-scoring strategies would also score well in a different environment. TIT FOR TAT is a generalist, whereas the highest-scoring evolved strategies were more specialized to the given environment.

Axelrod concluded that the GA is good at doing what evolution often does: developing highly specialized adaptations to specific characteristics of the environment. To see the effects of a changing (as opposed to fixed) environment, Axelrod carried out another experiment in which the fitness of an individual was determined by allowing the individuals in the population to play with one another rather than with the fixed set of eight strategies. Now the environment changed from generation to generation because the opponents themselves were evolving. At every generation, each individual played iterated games with each of the 19 other members of the population and with itself, and its fitness was again taken to be its average score over all games. Here the fitness landscape was not static-it was a function of the particular individuals present in the population, and it changed as the population changed. In this second set of experiments, Axelrod observed that the GA initially evolved uncooperative strategies. In the first few generations strategies that tended to cooperate did not find reciprocation among their fellow population members and thus tended to die out, but after about 10-20 generations the trend started to reverse: the GA discovered strategies that reciprocated cooperation and that punished defection (i.e., variants of TIT FOR TAT). These strategies did well with one another and were not completely defeated by less cooperative strategies, as were the initial cooperative strategies. Because the reciprocators scored above average, they spread in the population; this resulted in increasing cooperation and thus increasing fitness. Axelrod's experiments illustrate how one might use a GA both to evolve solutions to an

interesting problem and to model evolution and coevolution in an idealized way. One can think of many additional possible experiments, such as running the GA with the probability of crossover set to 0-that is, using only the selection and mutation operators (Axelrod and Kaufmann, 1987) or allowing a more open-ended kind of evolution in which the amount of memory available to a given strategy is allowed to increase or decrease (Lindgren, 1992; Melanie, 1999).

Noland (1996) analyzed the signal timing based on travel time costs of both pedestrians and vehicles at isolated intersections with high pedestrian demand. From an economic perspective, he claimed that pedestrians should be favored when high ratio of pedestrians to automobiles by decreasing automobile green phase length and increasing pedestrian green phase or by alternative strategies such as reducing major road width and closing selected streets to vehicular traffic at certain peak hours. However, Noland didn't show difference of optimized signal timing between models considering pedestrian and vehicle delay and those considering vehicle delay.

Ishaque (2005) analyzed effect of signal cycle timing on both vehicle and pedestrian delay in a hypothesized network by a VISSIM microsimulation model. Aiming at minimizing the multimodal travel delay and travel costs, they found that optimal cycle lengths under light traffic conditions, 60 to 72 seconds, were shorter than optimal cycle lengths under heavy traffic conditions (90 seconds). However, they only discussed eight fixed-time noncoordinated signal plans with single or double exclusive pedestrian phases. In addition, pedestrian compliance effect was not considered in the research. Based on their research in 2005, Ishaque (2007) studied trade-offs between pedestrian and vehicle traffic in the same hypothetical network by a VISSIM microsimulation model. Aiming at optimizing average travel cost per person in all modes of the network, they found that shorter cycle lengths were beneficial for pedestrians, and that signal plans advantageous to vehicles might be disadvantageous to pedestrians. Based on different proportions of pedestrians to vehicle users and different pedestrian time values, suitability of

three different pedestrian phase types was analyzed, so that the optimal network performance could be achieved for all road users. Compared with their previous research, pedestrian compliance effect was considered in the research, and the variety of signal plans was improved. However, the variety was still limited to a two-phase vehicle signal plans with single exclusive, or double exclusive, or staggered pedestrian crossing phase(s).

In 1998, Virkler completed four research projects about pedestrian traffic control;pedestrian travel time estimationVirkler (1998b), pedestrian signal coordination benefitsVirkler (1998d), pedestrian compliance effectVirkler (1998a), and pedestrian crossing timingVirkler (1998c):

- By referring to test vehicle technique for travel time, Virkler developed a method for pedestrian travel time, which could be viewed as a combination of average-car and floating-car techniques. The pedestrian travel time included walking time based on average pedestrian flow rate and queuing delay in signalized or unsignalized intersections based on random arrivals. However, his method ignored signal coordination effect. Therefore, it might overestimate or underestimate signal delay experienced by platoon pedestrians.

- Virkler studied signal coordination benefits for pedestrians through field data from 10 intersection approaches. He found that ideal offsets with a given cycle length tended to be shorter for longer green time.

- According to Virkler's research, 69 percent of pedestrians arriving at the curb during the flashing Don't Walk phase would enter the crosswalk. Thus, compared with complete signal compliance, delay would reduce 22 percent based on random arrivals.

- Virkler analyzed various methods developed to determine appropriate pedestrian crossing timing at signalized intersections. Based on field data,

relationships to describe pedestrian flow at signalized crossings were developed, and certain improvements of signal timing parameters were recommended under high-volume conditions and with two-way flow within a crosswalk.

A deterministic model (Bhattacharya, 2005) was proposed that incorporated both pedestrian and vehicle delay in a signal coordination plan. The author(s) analyzed the running results of the model on a hypothesized five-intersection arterial with various offsets and found that the best offsets for vehicles and pedestrians along the arterial were not necessarily the same. In order to minimize total pedestrian and vehicle user cost, an optimal signal coordination plan could be achieved by balancing between pedestrian and vehicular delay.

Li (2009) developed a traffic signal optimization strategy, programmed in Matlab, for an individual intersection to minimize weighted total vehicle and pedestrian delay. The total vehicle and pedestrian delay on sidewalk were calculated based on their deterministic queuing model respectively. Total pedestrian delay on crosswalk was calculated based on an empirical pedestrian speed model, which considered interactions between pedestrian platoons. According to a case study at a Japanese Intersection, the proposed model improved average person delay by 10% without changing existing cycle lengths, and the further improvement could reach 44% with additional cycle length optimization.

As discussed in Artificial Intelligence in Transportation (TRB, 2007), since the Genetic Algorithm (GA) was developed maturely in ’90 s, it has been employed to solve lots of complex transportation problems. Among all of them, traffic signal optimization is one of most popular areas where GA has been applied.

Foy and Goldberg. (1992) used GAs to optimize cycle lengths, green splits, and phase sequences in a four-intersection network by minimizing the total average wait time per car. Green splits were expressed by the percentage of cycle lengths, while phase sequences determined whether north-south or east-west direction

green signal displayed first at each intersection, but neither turning movements nor turning phases were considered in their research. According to test runs, the traffic GA always converged to reasonable timing strategies.

Park and Urbanik (1999) employed a GA optimizer with a mesoscopic simulator to optimize cycle length, green split, offset, and phase sequence of a hypothesized arterial system with low, medium, and high demand volume levels. The GA optimizer generated the first generation of individual signal plans randomly. The mesoscopic traffic simulator (an intermediate product of macroscopic and microscopic simulation with queue blocking effect modeling) evaluated average delay of each signal plan. Then the GA optimizer would evolve the next generation based on fitness values obtained from the simulator. The circulation process continued until the maximum generation number was reached. Compared with the solutions by TRANSYT-7F on the basis of a CORSIM simulation program, the solutions by GA had lower average delay under low and high demands and equivalent delay under medium demand.

In order to optimize signal control on mixed traffic arterials, Duerr (2000) used a GA with a microscopic traffic simulator as the fitness evaluator to minimize the performance index (PI) which considered vehicle behavior at intersections and transit stops. The optimization results of a seven-node arterial in WA˜ 14rzburg (Germany), temporal deviation of each phase duration from the standard setting at each node, showed that travel time dropped 25% and 5% for buses and cars respectively. Furthermore, so as to optimize signal control under oversaturated traffic condition, Girianna and Benekohal (2002, 2004) applied a GA to a grid network of arterials. The optimization results of a hypothesized twenty-node network, green time of each phase at an intersection, showed that queues were successfully distributed spatially over different intersections and temporarily over different signal cycles.

Genetic Algorithms is one of the most suitable methods to solve problems with complex objective functions, large number of variables, and mixed solution space. Therefore, in this thesis, it is chosen to realize signal timing optimization for both pedestrians and vehicles.

The first known attempt to apply fuzzy logic in traffic control was made by Pappis and Mamdani (1977). They simulated an isolated signalized intersection composed of two one-way streets with two lanes in each direction without turning traffic. The fuzzy controller reduced average vehicle delay compared to an actuated controller.

Kok Khiang Tan and Yusof (1996) describe a fuzzy logic controller for a single junction that mimics human intelligence. They used two sensors for each lane. The first sensor behind each traffic light counts the number of cars passing the traffic lights, and the second sensor behind the first sensor counts the number of cars coming to the intersection at distance from the lights. The fuzzy logic controller determines the time that the traffic light should stay in a certain state, before switching to the next state. The order of states is predetermined, but the controller can skip a state if there is no traffic in a certain direction. The amount of arriving and waiting vehicles are quantized into fuzzy variables, like many, medium and none. The activation of the variables in a certain situation is given by a membership function, so when there are 5 cars in the queue, this might result in an activation of 25% of 'many' and 75% of 'medium'. In their experiments, the fuzzy logic controller showed to be more flexible than fixed controllers and vehicle actuated controllers, allowing traffic to flow more smoothly, and reducing waiting time.

Foy and Goldberg. (1992) documents the use of genetic algorithm to optimize timing plans. The application object is an octothorpe-shaped traffic network with four intersections. Every intersection can run a two-phase plan. This method uses nine decision variables including the total green time of all phases, phase orders and splits. These nine decision variables are coded with 24 bits. The objective

function is the reciprocal of the total waiting time. A simulation model is used to evaluate the optimizing method. Results show that genetic algorithm is indeed a parallel optimizing method compared with traditional search methods. Zhiyong et al. propose an improved immunity genetic algorithms for an urban area coordinated traffic control system. The system adopts a two-level hierarchical distributed construction, with parameters that are hierarchically optimized at an interval of 5-30 minutes. In each interval, the cycles and offsets are optimized in the central controller while the splits are optimized in intersection controller. For a given performance index, such as minimizing the mean vehicle delay or number of stops etc., an improved immunity GA is used to optimize the cycle, offsets and splits. To ensure that the proposed method is plausible, simulations were conducted with positive results.

In a mutation operation of genetic algorithm has a logistic chaotic mapping applied upon and then a chaotic mutation is implemented, thus building a chaotic genetic algorithm. From this population, 5% of the individuals are selected randomly to search according to chaotic dynamic searching process. The results of this search replace the previous individuals who were part of this population. It shows that the new algorithm converges more quickly and avoids local optimization and premature convergence when simulated in CORSIM (CORridor SIMulation).

Andrea Vogel, Christian Goerick and Werner von Seelen, of the Institute of Neuroinformatik, Ruhr-University of Bochum, Germany, used Evolution Algorithm (a variance of Genetic Algorithm) to optimize traffic signal operation (Andrea Vogel and von Seelen, 2000).

In their work they investigated the Structure, Parameters, Boundary conditions and the Optimization process. Finding an adequate structure means to decide which traffic movements should get the right-of-way simultaneously, i.e. building phases, and to put them into a sequence. They also try to determine how far the signal operation should adapt to short term changes in the traffic situation. A high

amount of pedestrians for instance can introduce the necessity of an extra phase in which the complete motorized traffic is stopped and the intersection is reserved for walking. Similar situations can emerge from public transportation that takes priority over the rest of traffic, or, in case of railway traffic, new conflicts can be caused that enforce a regularization differing from the above mentioned standard solutions. The team did not restrict themselves to a given set of phase operations. Other restrictions, as from driving physics and law can be implemented as boundary conditions. Traffic Dependence: The search space gets additional dimensions by taking into account the possibility of traffic responsive signal operation. Furthermore this decision has consequences that surpass optimization: A well-adapted traffic responsive signal operation can certainly enhance the degree of service, but the need of traffic supervision and complex control software increases the costs of the signal installation.

In the framework of signal operation, optimizing parameters means to choose a timing plan adequate to the traffic situation. The dimension of the search space for parameter optimization, i.e. the number of parameters to optimize is determined by the chosen structure. The minimum is one green time per phase to determine. Further parameters occur in case of traffic actuated signalization: The green time adaption due to time gaps between vehicles for example needs a maximum green time and a traffic density threshold as condition when to switch to the next phase. Their approach is not restricted to the above mentioned parameters and can cope with qualitatively different parameter sets as well.

Classical boundary conditions are facts that are, if not explicitly stated, invisible to the optimization process. Typical boundary conditions for signal operations are restrictions by law, e.g. a minimum green time per phase or a maximum cycle length. Other boundary conditions can be given by the vehicle's physics: For a given speed, the time gap between two cars has a minimum value determined by reaction time and deceleration time. These structure or parameter restrictions are rather obvious and can easily be implemented as part of the operators. Less

obvious are those boundary conditions that are part of the fitness function, such as the actual traffic density. In order to gain results of the most possible significance, it can be very useful to clearly identify those hidden boundary conditions. In many cases, an adaptation to changing boundary conditions can be done extraordinarily fast HA˜ 14sken and Vogel (2000).

Since the design and optimization of signal operations involves combinatorial as well as continuous-valued problems, Andrea et al, decided to use evolutionary algorithms. These techniques have proven to be powerful tools for optimization of both parameters and structure and are known to efficiently search higher-dimensional data spaces. Evolutionary algorithms are able to handle nonanalytical fitness functions (such as simulation models). They can be forced to a global search or a hill climbing behaviour towards the nearest (possibly local) optimum. The idea of evolutionary algorithms is to describe a number of solutions as a population. Each individual of this population represents a solution, i.e. a phase system with its parameters. Each individual has its own quality, its fitness. The fittest individuals will become parents and reproduce: they create offsprings, the next generation of individuals. The reproduction is superimposed by mutations, i.e. a systematic error. So each offspring differs from its parent. Since the fittest survives and generates similar but not identical children, the search converges towards a global optimum. Structure evolution and parameter adaption are processes which extract information from the environment in an iterative process: In every time step (generation) the algorithm proposes a quantity of solutions. These solutions are tested in the environment, i.e. in this case the intersection traffic model. In terms of evolutionary algorithms, the solutions quality is called fitness; the environment is described by the fitness function. Structure evolution and parameter adaptation proceed on different time scales. Parameter adaptation is a fast process adapting a given structure fast to the environment and boundary conditions. Fast parameter adaptation means

flexibility and thus the ability to adapt quickly to changing conditions. Structure evolution provides the basis for parameter adaptation. It takes place on a long time scale. Which structure is stated as good depends strongly on the optimization's aim. Adaptability in mind, we require a structure on which the given parameter adaptation is fast and therefore yields in good fitness values for different boundary conditions. An alternative approach could be a specialized structure that gives the maximum fitness for a single condition. In this case, changing boundary conditions, such as fluctuating traffic densities, enforce a new structure evolution. The problem was adequately encoded. Structure and parameters have to be represented in such a way that they are accessible for the optimization algorithm. The design of such a representation or encoding is an important task, because it determines the properties of the whole search process. The operators for the search in structure and parameter space were choses. Evolutionary algorithms enhance the population's average fitness step by step by creating new individuals in the neighbourhood of the last generation and selecting the fittest of them as next generation. The operators for creating these new individuals determine the way through the solution space and therefore determine which solutions can be found and how fast they are reached. Since in this case a time consuming simulation run has to be done for each individual (i.e. for each proposed solution) the interest is in a very fast search. An adequate fitness function was developed.

The optimization process consisted of several parts. First is to distinguish between a system that generates a signal operation. This contains the evolutionary algorithm with the structure and parameter encoding, the search operators and the coupling of parameter adaptation and structure evolution. Second is to distinguish between a system that tests the proposed signal operation. This contains the environment model that yields the fitness function.

Since we want to be able to use different traffic models as well as different generating algorithms, we demand a modular structure for the optimization

process. Thus our process gets a third, passive element: the intersection container. The optimization process gets the following structure:

- To remain as independent of special models as possible, we implement a container for the intersection data. This allows an easy changing of the fitness function as well as the optimizing procedure.
- The search algorithm extracts intersection data such as traffic densities, former signal operations and their quality from the intersection container. Based on this information it generates new signal operations and writes them back into the container.
- The traffic model extracts intersection data (including a new signal operation) from the intersection container. Based on this information it evaluates the solution's quality and writes it back into the container.

An intersection container was used to guarantee the independence of the optimization and evaluation. It contains the fixed conditions under which traffic in this intersection takes place: The number of lanes per direction, maximum traffic densities, information about the installed traffic lights etc. Additionally, the actual traffic situation, which has a main influence on the optimization task, is filed in the intersection container.

Andrea et al decided to implement it as a group of vectors of fixed size. Each vector element is assigned to one traffic movement: The first three elements (0 to 2) represent traffic coming from north and turning left, straight on or right.

Elements 3 to 5 describe traffic from west, 6 to 8 cars from south, and elements 9 to 11 the ones coming from east. Three vectors contain the information about the number of lanes per direction (including 0), the actual and maximum traffic density. Possible extensions are e.g. a traffic density array instead of the vector to encode more than one traffic situation or model specific parameters.

The team decided to encode structure and parameters directly. This was done in the following way. The first chromosomes contain the phase operation. Each phase is represented by a single chromosome. The phase chromosomes have the same size as the container vectors. If a direction is part of the phase, the corresponding allel (i.e. the vector element) is marked with a 1, otherwise is set to zero. The number of phases is variable. The minimum is 2, the maximum corresponds to the number of directions, when each movement has its own phase. To speed up search, the number of phases can be restricted. The next chromosome contains the order of phases. The green time chromosome contains one allel per phase. Chromosomes contain information for traffic responsive signal operation, detector parameters for example. The experiments are based on a model intersection with four arms. Each arm contains three lines: one line for turn-left movements, one for straight on traffic and one for turn-right-movements. The main road traffic density is twice as high as on the crossing road. To obtain significant conflicts between straight-on and turn-left traffic we choose an asymmetric arrangement of the traffic density per movement: while the main direction from east and south is straight on, from west and north it is turn left. For a first demonstration of the feasibility of our approach, the team restricted their first experiments on nontraffic responsive intersections.

In the first experiments, the efficiency of different parameter adaptation strategies was tested. A (1 + 1) - strategie was implemented, also known as hillclimbing algorithm. This means that the population consists of one single parent producing one single offspring. Selection takes place among these two individuals. The hillclimber is known to be the most efficient parameter adaptation algorithm in evolutionary optimization (Rechenberg, 1994) Comparisons with other population sizes and mutation schemes confirm these results.

A binomial distribution with zero expectation was used as a mutation operator. The experiment was carried different variances and different initializations, i.e.

starting points for the parameter adaptation. Other experiments dealt with different evaluation schemes. Since the fitness function was noisy, it was useful to evaluate the parent's fitness each time it competes with a new offspring. As fitness function, the event based simulation model SIMVAS++, the simulation system of the Technische Universita¨t Dresden, Fakulta¨t fA˜ 14r Verkehrswissenschaften (Ringel, 1995) was used. Immanent to this model (as well as to reality) is a certain fluctuation of the simulation results. Therefore the noise in the adaptation process had to be dealth with.

The influence of initialization, adaptation step size and the fitness function's inherent fluctuations on the adaptation process and result were tested.

A first result is that the gain of using expert knowledge for initialization is minimal: Started from small random values, the algorithm reaches a good parameter set very fast. In cases of high traffic densities, traditional heuristic green time estimation methods overestimate the cycle length significantly and lead to bad fitness values. The small random initialization may even be worse at the beginning. But since the decay towards the optimum for growing cycle lengths is much steeper than for shrinking cycle lengths, particular at the start of the adaptation, the hillclimber is significantly faster. Furthermore, the results are less varying: while from five heuristically initialized runs only one reached a reasonable fitness, the randomly initialized runs nearly all ended up in the same minimum.

An adequate mutation step size problem was chosen. In the literature about evolutionary algorithms there exists a wide range of methods to deal with this task automatically. In these methods, step sizes become part of the optimization process. All these methods are oversized for this task. We compared two nonadaptive strategies (small and large step size) with a simple simulated annealing, where the step size is systematically reduced during the adaptation process. In some experiments we could observe a small speed gain for simulated

annealing versus large step sizes. Small step sizes can result in a slow adaptation when starting far from the optimum. The step size does not seem to be a critical parameter.

The fitness function is inherently noisy, i.e. several evaluations with the same parameter set lead to slightly different results. These fluctuations stem from the random number generator of SIMVAS++. The qualitative results are the same with and without multiple fitness evaluations. Compared over generations, the fitness development with noise is nearly the same as without noise. When we compare the run time, taking the noise into account becomes a drawback. Instead of one fitness evaluation (i.e. one simulation run) per generation, two are needed and thus the double run time of the parameter adaptation. Parameter adaptation is a task that can be easily solved with evolutionary methods. To optimize the parameter set for a given structure, it might prove useful to use simulated annealing to reduce the mutation step size during the adaptation. All the proposed methods lead to similar results: differences in the fitness values are more often caused by the fluctuations of the fitness function than by ending up with different parameter sets.

Combined with structure evolution, the aim of parameter adaptation is not to find the optimal parameter set for each structure. As described in Keesing and Stokes (1991), a too exhaustive parameter adaptation reduces the performance of the complete optimization process. Thus it is uncritical wether we coose a constant parameter step size or simulated annealing or whatever clever strategy. Besides the mixture of parameter and structure steps, good structure mutation operators are crucial for structure optimization. While a rarely changed structure benefits from the so far reached parameter adaptation, a structure that is changed too much in a single mutation loses all these benefits. Other points to keep in mind in structure optimization are the population sizes and the selection scheme. The evaluation of the structure optimization is subject of current work. Our first results show a positive tendency.

Starting from randomly selected four phase operation, the algorithm reduces either conflicts between traffic movements or reduces the number of phases.

The adaption of signal operation systems to special intersections and traffic situations is a high dimensional optimization problem that consists of combinatorial as well as continuous valued problems.A homogeneous approach to such problems are evolutionary algorithms. In this paper we presented a decomposition of the optimization process for using different signal operation systems and traffic models as well as a variety of optimization tools. Since the problem encoding and the optimization operators determine mainly the optimization results, they have to be choosen very carefully.

In their first experiments, their approach shows very promising results. The parameter optimization process seems to be rather robust against different operators and initialization and reliably finds good solutions. Structure optimization tends to reduce complexity in reducing the number of phases.

Chapter 3

Methodology

3.1 Overview of Genetic Algorithm

Every organism has a set of rules, a blueprint, describing how that organism is built up from the tiny building blocks of life. These rules are encoded in the genes of an organism, which in turn are connected together into long strings called chromosomes. Each gene represents a specific trait of the organism, like eye colour or hair colour, and has several different settings. For example, the settings for a hair colour gene may be blonde, black or auburn. These genes and their settings are usually referred to as an organism's genotype. The physical expression of the genotype - the organism itself - is called the phenotype.

When two organisms mate they share their genes. The resultant offspring may end up having half the genes from one parent and half from the other. This process is called recombination. Very occasionally a gene may be mutated. Normally this mutated gene will not affect the development of the phenotype but very occasionally it will be expressed in the organism as a completely new trait. Genetic Algorithms are a way of solving problems by mimicking the same processes mother nature uses. They use the same combination of selection, recombination and mutation to evolve a solution to a problem (Ai-Junkie, 2015).

Genetic Algorithm (GA) was developed by John Holland in 1975. It is a method for solving both constrained and unconstrained optimization problems that is based on natural selection. Natural selection is the process that drives biological evolution. The GA repeatedly modifies a population of individual solutions. At each step, the genetic algorithm selects individuals at random from the current population to be parents and uses them to produce the children for the next generation. Over successive generations, the population "evolves" toward an

optimal solution. The strong tend to adapt and survive while the weak tend to die out. That is, optimization is based on evolution, and the ”Survival of the fittest” concept. GAs have the ability to create an initial population of feasible solutions, and then recombine them in a way to guide their search to only the most promising areas of the state space. Each feasible solution is encoded as a chromosome (string) also called a genotype, and each chromosome is given a measure of fitness via a fitness (evaluation or objective) function.

Genetic algorithm maybe applied to solve a variety of optimization problems that are not well suited for standard optimization algorithms, including problems in which the objective function is discontinuous, non-differentiable, stochastic, or highly non-linear. The genetic algorithm can address problems of mixed integer programming, where some components are restricted to be integer-valued.

GA is a robust search technique. It will produce ”close” to optimal results in a ”reasonable” amount of time. It is suitable for parallel processing. GAs are blind without the fitness function.

GA can be used when an acceptable solution representation is available, when a good fitness function is available, when it is feasible to evaluate each potential solution, when a near-optimal, but not optimal solution is acceptable, and when the state-space is too large for other methods.

3.2 Genetic Algorithm Concept

GA process goes through Initialization, Fitness Evaluation, Selection, Recombination (Crossover), Mutation. The process is repeated till a stopping criteria is attained.

3.2.1 Initial Population

Before GA can be used to solve a problem, some preliminary considerations must be made. How the feasible solution will be represented must be determined. This

could be done by a choice of alphabet or number that permits a natural expression of the problem. The length of the alphabet or number must be determined.

Some factors that could influence the initial population are the search space, the fitness function, the diversity, the problem difficulty, the selection pressure, and the number of individuals (population size). The population size will remain constant throughout the algorithm. It has been recognized that if the initial population to the GA is good, then the algorithm has a better possibility of finding a good solution. If the initial supply of building blocks is not large enough or good enough, then it would be difficult for the algorithm to find a good solution. Sometimes, if the problem is quite difficult and some information regarding the possible solution is available, then it is good to seed the GA with that information. Objective Function to be used in the algorithm need to be determined. An objective function is an equation to be optimized given certain constraints and with variables that need to be minimized or maximized.

A measure of diversity plays a role here in the sense that, when we have no information regarding a possible solution, then we could expect, that the more diverse the initial population is, the greater the possibility to find a solution is, and of course, the number of individuals in the population to get a good degree of diversity becomes important.

Diversity could be good in terms of performance of the algorithm and diversity has been used not only to generate the initial population but also as a way to guide the algorithm to avoid premature convergence.

The selection pressure should be taken into account in the initial population size. If a selection pressure SP1 is greater than a selection pressure SP2, then, when using selection pressure SP1 the population size should be larger than when using selection pressure SP2, because a higher selection pressure can cause a decrease in diversity of the population at a greater rate, perhaps causing the algorithm to converge prematurely.

The fitness function can be taken into account, in the sense that it could be better to generate the initial population in a pseudo-random way than letting it go only with randomness or that there could be a degree of correlation between diversity and the fitness function. Besides that, the fitness evaluation of the initial population can be used as a metric of diversity, looking, for example, at the initial standard deviation of fitness values and evaluating the dispersion of such values.

Population Representations

Chromosomes can be represented as:

- Bit Strings (1011 ... 0100)
- Reals (19.3, 45.1, -12.9, ... 6.2)
- Integers (1,4,2,7,5,9,3,6,8) Usually Permutations of 1..n
- Characters (A, G, Q, ... F) Usually Permutations
- List of rules (R1, R2, ... R20)
- Chromosomes are all of the same type (Bit Strings)
- Chromosomes are all the same length
- The population size remains constant from generation to generation

3.2.2 Encoding

Before you can use a genetic algorithm to solve a problem, a way must be found to encode any potential solution to the problem. This could be as a string of real numbers or, as is more typically the case, a binary bit string(chromosome). A typical chromosome may look like this:

10010101110101001010011101101111111101

Chromosome 1	1 1 0 1 0 1 1 1 0 0 1 0
Chromosome 2	0 1 1 0 1 0 0 1 1 1 0 1

Table 3.1: Binary Encoding

Chromosome 1	0 6 2 5 4 5 2 4
Chromosome 2	6 3 7 2 6 4 2 5

Table 3.2: Octal Encoding

At the beginning of a run of a genetic algorithm a large population of random chromosomes is created. Each one, when decoded will represent a different solution to the problem at hand.

Binary Encoding

This is the most common form of encoding. In this encoding each chromosome is represented using a binary string. In binary encoding every chromosome is a string of bits, 0 or 1. Figure 3.1 shows an example of binary encoding.

In this encoding each bit show some characteristic of solution. On the other side each binary string represents a value. With smaller number of alleles, a number of chromosomes can be represented.

Octal Encoding

In this encoding chromosome is represented using octal numbers (0-7). Figure 3.2 shows the Octal encoding.

Hexadecimal Encoding

In this encoding chromosome is represented using Hexadecimal numbers (0-9, A-F).Figure 3.3 shows the hexadecimal encoding.

Chromosome 1	9 7 A E 0

Chromosome 2	A 2 C 6

Table 3.3: Hexadecimal Encoding

Chromosome 1	1 5 2 3 5 2 6 4 6 9 8
Chromosome 2	8 6 3 6 3 9 6 3 1 5 8

Table 3.4: Permutation Encoding

Chromosome 1	1.23 2.12 3.14 0.34 4.62
Chromosome 2	ABDJEIFJDHDDLDFLFEGT

Table 3.5: Value Encoding

Permutation Encoding

Permutation Encoding is used in ordering problems. In this, each chromosome represents position in a sequence e.g. in travelling salesman problem, the string of numbers represent the sequence of cities visited by salesman. Sometimes corrections have to be done after genetic operation is completed. Figure 3.4 shows the Permutation encoding.Permutation encoding is only useful for problems that have specific order. Some types of crossover and mutation corrections must be made to leave the chromosome consistent (i.e.,have real sequence in it) for such kind of problems.

Value Encoding

In value encoding, each chromosome is represented as the string of some value. Value can be integer, real number, character or some object. In case of Integer values, the crossover operator applied are same as that applied on binary encoding. Values can be anything connected to problem, form numbers, real numbers or chars to some complicated objects. The Figure 3.5 shows the Value encoding. Value Encoding can be used in neural networks. This encoding is

generally use in finding weights for neural network. Chromosome's value represents corresponding weights for inputs.

Figure 3.1: Tree Encoding

Tree Encoding

Tree encoding is mainly used for evolving programs or expressions for genetic programming. In tree encoding every chromosome is a tree of some objects, such as functions or commands in programming language. Tree encoding is good for evolving programs. LISP is useful in this encoding as it helps in constructing tree for parsing and hence the crossover and mutation can be performed easily. Chromosomes are functions represented in a tree as in The Figure 3.1.

3.2.3 Parent Selection Scheme

Roulette Wheel Selection

Parents are selected according to their fitness. It does not guarantee that the fittest member goes through to the next generation merely that it has a very good

chance of doing so. The better the chromosomes are, the more chances to be selected they have. Imagine that the population's total fitness score is represented by a pie chart, or roulette wheel. Now you assign a slice of the wheel to each member of the population. The size of the slice is proportional to that chromosomes fitness score. i.e. the fitter a member is the bigger the slice of pie it gets. Now, to choose a chromosome all you have to do is spin the ball and grab the chromosome at the point it stops.

Rank Selection

The Roulette Wheel selection will have problems when the fitnesses differs very much. For example, if the best chromosome fitness is 90% of all the roulette wheel then the other chromosomes will have very few chances to be selected.

Rank selection first ranks the population and then every chromosome receives fitness from this ranking. The worst will have fitness 1, second worst 2 etc. and the best will have fitness N (number of chromosomes in population).

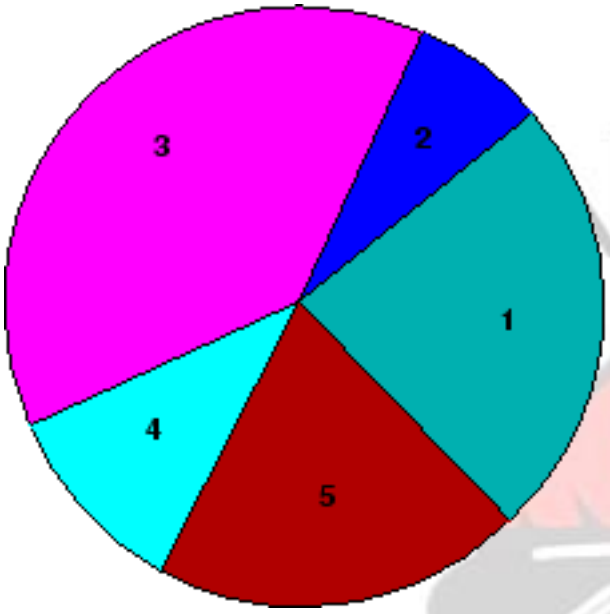
Steady-State Selection

This is not particular method of selecting parents. Main idea of this selection is that big part of chromosomes should survive to next generation.
GA then works in a following way. In every generation are selected a few (good - with high fitness) chromosomes for creating a new offspring. Then some (bad - with low fitness) chromosomes are removed and the new offspring is placed in their place. The rest of population survives to new generation.

Elitism

When creating new population by crossover and mutation, we have a big chance, that we will loose the best chromosome.

Elitism method copies the best chromosome (or a few best chromosomes) to new population. The rest is done in classical way. Elitism can very rapidly



Population	Fitness
1	25.0
2	5.0
3	40.0
4	10.0
5	20.0

Figure 3.2: Roulette Wheel Selection

increase performance of GA, because it prevents losing the best found solution.

3.2.4 Crossover or Recombination

Crossover rules combine two parents to form children for the next generation. Crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Cross over is a process of taking more than one parent solutions and producing a child solution from them.

The chance that two chromosomes will swap their bits is crossover rate. A good value for this is around 0.7. Crossover is performed by selecting a random gene along the length of the chromosomes and swapping all the genes after that point.

e.g. Given two chromosomes

10001001110010010

01010001001000011

Choose a random bit along the length, say at position 9, and swap all the bits after that point, so the above become:

10001001101000011

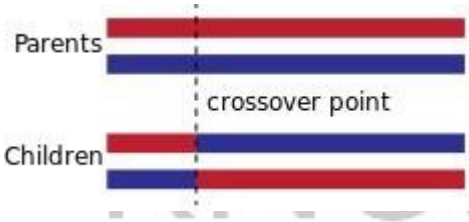


Figure 3.3: One Point Crossover

Figure 3.4: Two Point Crossover

01010001010010010

One-point crossover

A single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children:

Two-point crossover

Two-point crossover calls for two points to be selected on the parent organism strings. Everything between the two points is swapped between the parent organisms, rendering two child organisms:

"Cut and splice"

Another crossover variant, the "cut and splice" approach, results in a change in length of the children strings. The reason for this difference is that each parent string has a separate choice of crossover point.

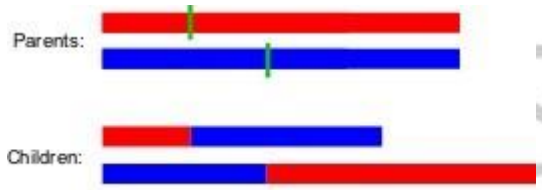


Figure 3.5: Cut and Splice Crossover

k-point Crossover

The concept of one-point crossover can be extended to k-point crossover, where k crossover points are used, rather than just one or two (Kumara Sastry and Kendall, 2005).

Uniform Crossover

In uniform crossover, every allele is exchanged between the a pair of randomly selected chromosomes with a certain probability, pe, known as the swapping probability. Usually the swapping probability value is taken to be 0.5 (Syswerda, 1989; Spears and De Jong, 1994; Kumara Sastry and Kendall, 2005).

Uniform Order-Based Crossover

The k-point and uniform crossover methods described above are not well suited for search problems with permutation codes such as the ones used in the traveling salesman problem. They often create offspring that represent invalid solutions for the search problem. Therefore,when solving search problems with permutation codes, a problem-specific repair mechanism is often required (and used) in conjunction with the above recombination methods to always create valid candidate solutions. Another alternative is to use recombination methods

developed specifically for permutation codes, which always generate valid candidate solutions. Several such crossover techniques are described in the following paragraphs starting with the uniform order-based crossover.

In uniform order-based crossover, two parents (say P1 and P2) are randomly selected and a random binary template is generated. Some of the genes for offspring C1 are filled by taking the genes from parent P1 where there is a one in the template. At this point we have C1 partially filled, but it has some ”gaps”. The genes of parent P1 in the positions corresponding to zeros in the template are taken and sorted in the same order as they appear in parent P2. The sorted list is used to fill the gaps in C1. Offspring C2 is created by using a similar process (Kumara Sastry and Kendall, 2005).

Order-Based Crossover

The order-based crossover operator (Davis, 1985) is a variation of the uniform order-based crossover in which two parents are randomly selected and two random crossover sites are generated. The genes between the cut points are copied to the children. Starting from the second crossover site copy the genes that are not already present in the offspring from the alternative parent (the parent other than the one whose genes are copied by the offspring in the initial phase) in the order they appear (Kumara Sastry and Kendall, 2005)

Partially Matched Crossover (PMX)

Apart from always generating valid offspring, the PMX operator (Goldberg and Lingle, 1985) also preserves orderings within the chromosome. In PMX, two parents are randomly selected and two random crossover sites are generated. Alleles within the two crossover sites of a parent are exchanged with the alleles corresponding to those mapped by the other parent (Kumara Sastry and Kendall, 2005)

3.2.5 Mutation

Mutation rules apply random changes to individual parents to form children. Mutation is a genetic operator used to maintain genetic diversity from one generation of population of genetic algorithm chromosomes to the next. It is analogous to bi-

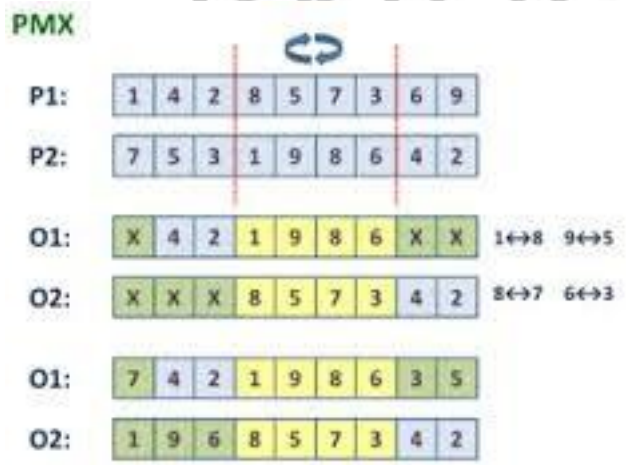


Figure 3.6: Partially Matched Crossover

ological mutation. Mutation alters one or more gene values in a chromosome from its initial state. The solution may change entirely from the previous solution. GA can come to better solution by using mutation. Mutation occurs during evolution according to a user-definable mutation probability. This probability should be set low. If it is set too high, the search will turn into a primitive random search. The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence will be changed from its original state. A common method of implementing the mutation operator involves generating a random variable for each bit in a sequence. This random variable tells whether or not a particular bit will be modified. This mutation procedure, based on the biological point mutation, is called single point mutation. Other types are inversion and floating point mutation. When the gene encoding is restrictive as in permutation problems, mutations are swaps, inversions and scrambles.

The purpose of mutation in GAs is preserving and introducing diversity. Mutation should allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. This reasoning also explains the fact that most GA systems avoid only taking the fittest of the population in generating the next but rather a random (or semi-random) selection with a weighting toward those that are fitter.

For different genome types, different mutation types are suitable.
The chance that a bit within a chromosome will be flipped is called mutation rate.

- 0 becomes 1
- 1 becomes 0

This is usually a very low value for binary encoded genes, say 0.001
Whenever chromosomes are chosen from the population the algorithm first checks to see if crossover should be applied and then the algorithm iterates down the length of each chromosome mutating the bits if applicable.

Bit string mutation

The mutation of bit strings ensue through bit flips at random positions.

The probability of a mutation of a bit is, where is the length of the binary vector. Thus, a mutation rate of per mutation and individual selected for mutation is reached.

Flip Bit

This mutation operator takes the chosen genome and inverts the bits. (i.e. if the genome bit is 1,it is changed to 0 and vice versa)

Boundary

This mutation operator replaces the genome with either lower or upper bound randomly. This can be used for integer and float genes.

Non-Uniform

The probability that amount of mutation will go to 0 with the next generation is increased by using non-uniform mutation operator. It keeps the population from stagnating in the early stages of the evolution. It tunes solution in later stages of evolution. This mutation operator can only be used for integer and float genes.

Uniform

This operator replaces the value of the chosen gene with a uniform random value selected between the user-specified upper and lower bounds for that gene. This mutation operator can only be used for integer and float genes.

Gaussian

This operator adds a unit Gaussian distributed random value to the chosen gene. If it falls outside of the user-specified lower or upper bounds for that gene, the new gene value is clipped. This mutation operator can only be used for integer and float genes.

Chapter 4

Analysis

The purpose of this work is to assess the potential of applying GA to solve a real world problem. The real world problem under consideration in this work is a four way traffic intersection. The purpose is to find a schedule for optimal drive order of traffic entering and leaving the intersection. The time duration of the traffic lights is supposed to be fixed.

4.1 Model Formulation

The following are the assumptions used in this work:

- The intersection under study is adjacent(That is, the intersection is a 'plus' junction)
- Vehicles may not cross into other lanes when traversing the intersection until they are a safe distance away from the intersection.
- No overtaking allowed within the intersection
- There are no islands in the road to aid pedestrian crossing
- Traffic will be grouped into streams and will traverse the intersection in four phases (four compatible streams).
- Traffic going from east to west and west to east is twice heavier than traffic going from north to south and from south to north.
- Only Traffic going from east to west and west to east are allowed to make 'U' turns
- A traffic cycle is made of four phases of traffic movements

- A phase may have no traffic stream traversing the intersection
- A phase may have upto six streams traversing the intersection

4.1.1 Description of the intersection

The intersection has six lanes; three lanes carrying traffic streams from east to west and another three lanes carrying traffic from west to east. There is also a lane carrying traffic from north to south, and another lane carrying traffic from south to north. The intersection has been simplified to have fourteen streams of traffic movement. Each stream has its own lane and an independent vehicle queue. We suppose overtaking is not allowed, which indicates that that vehicles on each lane need to pass the intersection on fist-in first-out way. The path used by a traffic stream to traverse the intersection is called the trajectory. A trajectory connects an approach on which vehicles enter the intersection to the leg on which these vehicles leave the intersection. Each stream has its own trajectory. The objective of the traffic control at the intersection is to transform input traffic flows into output ones while preventing traffic conflicts and satisfying a specific criterion. This work is based on making a traffic cycle of four phases. Each phase carries streams of traffic which are compatible. Compatible streams are streams whose trajectories do not cross.

4.1.2 Steps involved in the algorithm

This algorithm consists of five steps:

- chromosome encoding,
- initialize population,
- evaluate population,
- chromosome selection and

Vector	Traffic Stream
1	Traffic moving from North to West
2	Traffic moving from North to South
3	Traffic moving from North to East
4	Traffic moving from West to North
5	Traffic moving from West to East
6	Traffic moving from West to South
7	Traffic moving from South to West
8	Traffic moving from South to North
9	Traffic moving from South to East
10	Traffic moving from East to South
11	Traffic moving from East to West
12	Traffic moving from East to North
13	Traffic making U-turn from West
14	Traffic making U-turn from East

Table 4.1: Vectors are assigned to each traffic stream based on its direction

- crossover.

4.1.3 Chromosome Encoding

Vectors (1 to 14) are assigned to traffic movement streams. This is to give identity to each stream and to make encoding of traffic cycles less difficult. The vectors are assigned as shown in Table 4.1

4.1.4 Initialize Population

We used Permutation Encoding in this work as the problem under study is an ordering problem. Each chromosome contains twenty four genes. The genes are the vectors assigned to the traffic streams based on how they traverse the intersection. There are fourteen streams and that gives fourteen vectors. The rest of ten genes in the chromosome are zeros which act as place holders. Each vector represents a traffic stream. The order of the vectors in the chromosome

represents the sequence of movement. Permutation encoding is only useful for problems that have specific order.

Samples of chromosomes used in this study are presented in Table 4.2

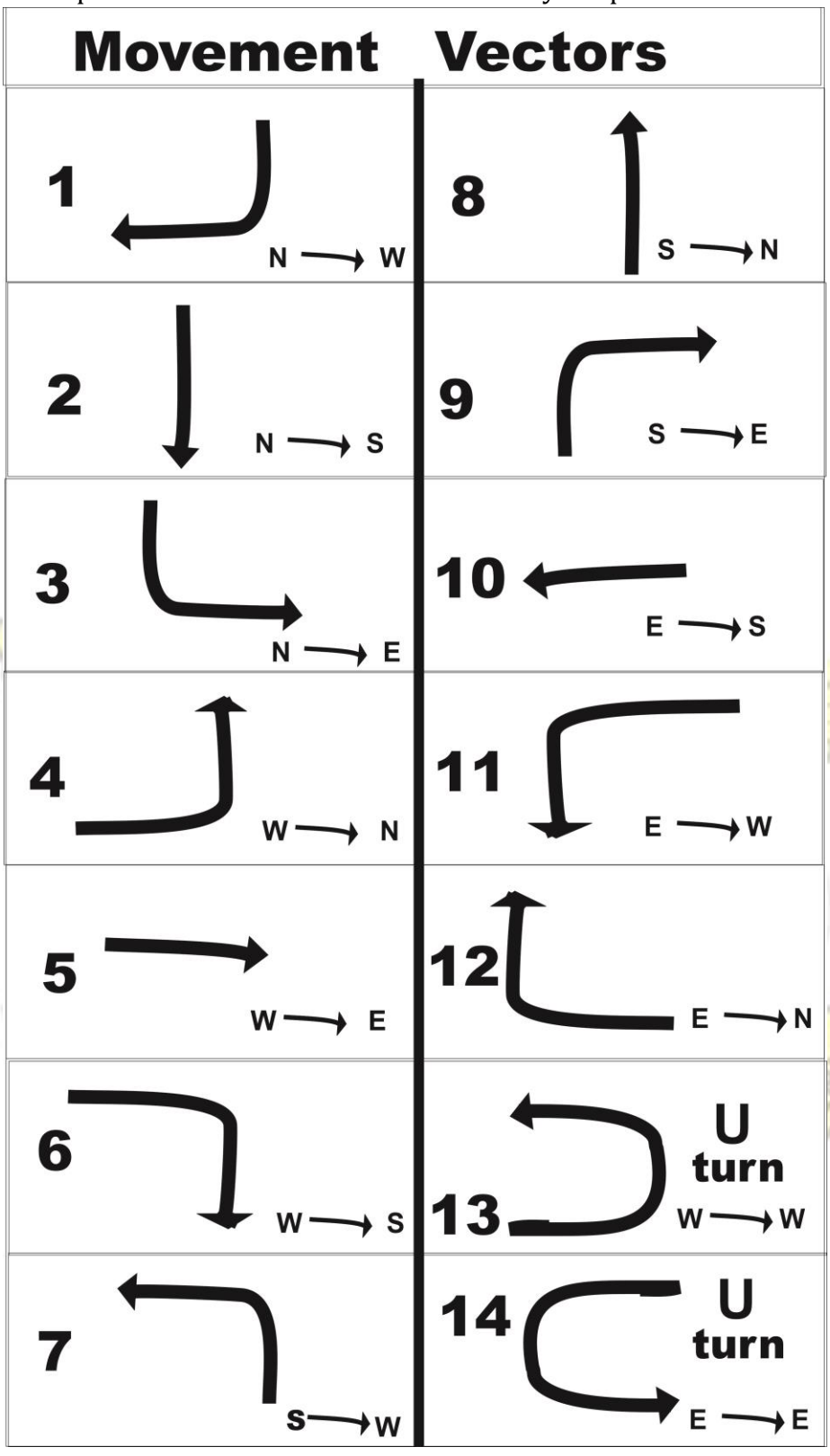


Figure 4.1: Vectors assigned to traffic streams

A	5	0	0	14	10	1	13	2	8	11	0	3	0	0	0	0	12	0	9	7	4	0	0	6
B	11	0	10	0	6	14	1	0	2	3	7	0	0	0	13	4	0	12	9	0	5	8	0	0
C	14	0	0	4	11	6	12	0	7	2	5	0	0	0	3	13	0	0	0	9	10	1	8	0

Table 4.2: Samples of chromosomes encoding

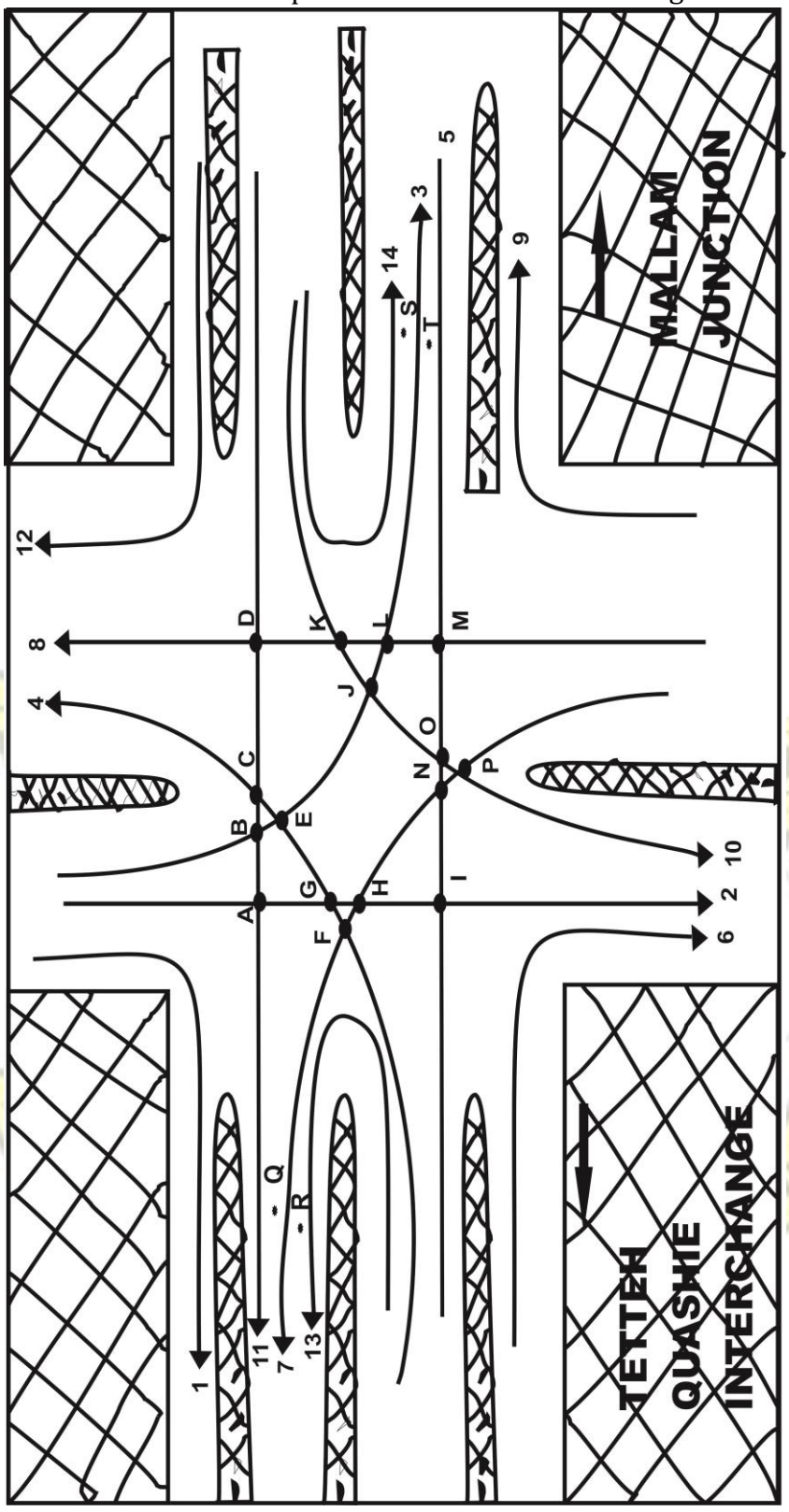


Figure 4.2: A sketch of La Paz Intersection

Chromosome No.	Chromosome
Chromosome 1	11 7 0 0 1 0 8 6 3 2 0 9 5 0 0 13 4 14 12 10 0 0 0 0
Chromosome 2	0 7 2 0 13 0 3 6 4 1 0 8 12 11 0 14 0 5 9 10 0 0 0 0
Chromosome 3	11 0 0 14 12 7 0 9 0 5 6 13 10 0 3 0 0 4 0 0 2 0 8 1
Chromosome 4	3 2 0 10 7 1 4 0 0 12 5 0 0 0 0 0 13 8 11 14 0 9 6 0
Chromosome 5	14 5 0 4 1 0 10 0 0 2 3 7 12 0 8 0 9 0 11 0 0 13 6 0
Chromosome 6	0 1 3 0 7 4 9 6 11 0 12 0 0 0 14 0 8 5 0 2 0 0 10 13
Chromosome 7	0 0 0 9 11 14 7 0 0 5 13 6 0 0 10 12 0 3 0 8 0 2 4 1
Chromosome 8	9 8 5 0 6 13 1 3 0 10 0 0 0 11 0 7 12 2 0 4 14 0 0 0
Chromosome 9	0 2 13 0 12 10 11 3 14 9 5 0 0 0 4 7 0 0 0 1 8 6 0 0
Chromosome 10	0 4 13 0 7 14 3 5 0 8 10 2 9 0 6 11 0 0 0 0 0 0 1 12
Chromosome 11	6 0 1 12 11 0 0 3 0 5 0 10 4 7 14 2 8 9 0 0 0 0 0 13
Chromosome 12	0 7 0 0 12 0 4 8 5 0 9 10 14 0 1 0 6 2 11 0 13 0 3 0
Chromosome 13	0 8 0 2 9 0 10 1 5 0 0 0 13 0 0 7 0 4 6 14 12 3 0 11
Chromosome 14	0 8 0 12 7 0 0 0 5 0 4 13 0 0 1 9 0 6 14 10 11 0 3 2
Chromosome 15	0 9 2 0 0 11 1 5 12 7 0 0 3 0 0 13 4 6 0 0 10 14 8 0
Chromosome 16	0 0 0 0 11 0 10 7 9 0 0 0 14 0 13 2 4 8 0 3 12 5 6 1
Chromosome 17	0 7 12 0 0 1 8 3 11 6 0 0 2 0 10 0 4 9 0 0 5 14 13 0
Chromosome 18	0 4 0 11 10 0 0 14 12 0 0 2 1 7 9 6 0 8 13 0 3 0 0 5
Chromosome 19	0 2 0 0 10 12 0 0 0 4 3 14 0 13 8 7 11 0 9 1 6 0 0 5
Chromosome 20	0 7 0 6 10 4 0 2 12 14 1 0 3 8 0 13 0 0 11 9 0 0 0 5
Chromosome 21	0 0 11 6 13 5 0 0 0 0 8 7 12 2 0 10 0 3 4 1 9 0 14 0
Chromosome 22	10 11 1 0 5 0 8 3 0 0 0 7 12 0 13 0 9 4 2 6 0 14 0 0
Chromosome 23	0 0 11 9 1 4 5 14 0 0 0 7 12 0 3 0 2 0 6 10 13 8 0 0
Chromosome 24	0 13 6 1 0 11 14 2 0 0 10 0 9 12 3 8 0 4 0 7 0 0 0 5
Chromosome 25	7 14 0 0 12 0 0 3 2 1 13 0 8 0 6 10 0 0 11 5 0 0 4 9
Chromosome 26	14 3 0 10 0 9 5 12 0 1 11 2 8 6 4 0 0 0 7 0 13 0 0 0
Chromosome 27	0 0 9 0 1 0 14 0 0 8 2 4 0 13 12 6 10 7 0 11 5 0 0 3
Chromosome 28	7 0 5 0 0 1 0 9 0 4 3 11 10 0 13 12 0 2 14 0 6 0 0 8
Chromosome 29	0 6 0 2 0 4 12 5 13 0 0 10 7 3 0 0 14 0 1 0 8 0 9 11
Chromosome 30	0 4 7 0 9 10 0 11 0 6 12 3 13 0 2 0 8 0 0 14 0 0 1 5
Chromosome 31	0 6 0 9 0 13 3 7 14 0 5 0 11 12 0 0 10 0 2 8 1 0 0 4
Chromosome 32	6 5 0 0 0 0 10 2 14 12 0 11 0 3 8 9 0 0 13 0 4 0 1 7
Chromosome 33	0 6 0 11 0 1 3 0 13 0 10 0 9 5 14 4 7 12 2 0 0 0 8 0
Chromosome 34	11 0 0 0 0 0 14 3 6 2 1 5 7 4 12 0 0 13 8 0 10 0 9 0
Chromosome 35	10 6 2 11 13 1 14 0 12 8 5 0 9 0 0 0 4 0 0 0 0 7 0 3
Chromosome 36	0 8 0 2 9 0 10 1 5 0 0 0 13 0 0 7 0 4 6 14 12 3 0 11
Chromosome 37	6 5 0 0 0 0 10 2 14 12 0 11 0 3 8 9 0 0 13 0 4 0 1 7
Chromosome 38	13 11 0 14 9 0 10 0 7 5 0 6 2 0 4 3 0 0 0 12 8 0 0 1
Chromosome 39	0 1 12 3 10 0 6 8 7 0 11 0 13 0 9 0 2 14 0 4 0 0 0 5
Chromosome 40	0 2 0 0 10 12 0 0 0 4 3 14 0 13 8 7 11 0 9 1 6 0 0 5
Chromosome 41	14 0 6 4 0 0 0 13 0 0 7 0 12 0 10 3 11 0 1 5 9 0 2 8
Chromosome 42	12 9 6 5 0 0 1 4 0 0 14 0 10 8 13 2 3 11 7 0 0 0 0 0
Chromosome 43	0 0 10 12 0 1 0 0 2 14 4 13 7 6 0 0 5 8 0 9 0 0 3 11

Chromosome 44	0 7 0 0 12 0 4 8 5 0 9 10 14 0 1 0 6 2 11 0 13 0 3 0
Chromosome 45	0 10 6 12 5 0 0 14 2 8 4 11 0 1 0 0 7 0 0 9 0 0 3 13
Chromosome 46	14 0 6 4 0 0 0 13 0 0 7 0 12 0 10 3 11 0 1 5 9 0 2 8
Chromosome 47	0 3 11 2 0 0 13 14 0 4 9 8 1 7 0 0 10 0 5 12 6 0 0 0
Chromosome 48	0 5 8 7 0 0 12 10 0 13 4 14 0 0 2 0 0 0 3 11 1 0 9 6
Chromosome 49	0 0 1 14 0 0 12 6 4 7 0 5 11 9 0 2 3 0 13 10 0 0 0 8
Chromosome 50	5 12 1 0 14 0 0 0 0 0 3 9 8 10 13 2 4 6 7 0 0 11 0 0

Table 4.3: Initial Population

Part 1	5	0	0	14	10	1
Part 2	13	2	8	11	0	3
Part 3	0	0	0	0	12	0
Part 4	9	7	4	0	0	6

Table 4.4: Chromosomes divided into four parts showing four phases traffic movement

Fifty chromosomes are randomly generated to set the initial population to fifty (50). Kindly refer to Table 5.6

4.1.5 Evaluate Population

Evaluation provides a way to rate how each chromosome (candidate solution) solves the problem at hand. The steps involved in this evaluation are:

- Breaking each chromosome into four equal parts.
  - The first six genes makes part one (1)
  - the seventh to twelve genes makes part two (2),
  - the thirteenth gene to the eighteenth makes part three (3) and – the nineteenth to the twenty forth gene makes part four (4).

Division of Chromosomes into parts can be found in Table 4.4

  - Each part of the chromosome forms a phase movement.

* Part 1 is phase 1,

* part 2 is phase 2,

* part 3 and

* part 4 makes phase 4.

These four parts makes one traffic cycle.

Part 1	5	0	0	14	10	1

Table 4.5: Interpretation of chromosome division - part 1

Part 2	13	2	8	11	0	3

Table 4.6: Interpretation of chromosome division - part 2

4.1.6 Interpretation of Chromosomes

- Streams 5, stream 14, stream 10 and stream 1 will be given right of way to traverse the intersection concurrently in phase 1. Table 4.5
- Streams 13, stream 2, stream 8, stream 11 and stream 3 will be given right of way to traverse the intersection concurrently in phase 2. Table 4.6
- Streams 12 will be given right of way to traverse the intersection in phase 3. Table 4.7
- Streams 9, stream 7, stream 4 and stream 6 will be given right of way to traverse the intersection concurrently in phase 4. Table 4.8
- Formation of unique pairs.
- The vectors (genes) are first sorted in ascending order. This is also a Mutation being applied to each chromosome. This Mutation is called Re-order list mutation. Table 4.9

• Unique pairs are formed from each part. Table 4.10

Part 3	0	0	0	0	12	0

Table 4.7: Interpretation of chromosome division - part 3

Part 4	9	7	4	0	0	6

Table 4.8: Interpretation of chromosome division - part 4

Part 1	0	0	1	5	10	14
Part 2	0	2	3	8	11	13
Part 3	0	0	0	0	0	12
Part 4	0	0	4	6	7	9

Table 4.9: Re-ordering of genes in sample

4.1.7 Fitness Function

Fitness of each individual is calculated over the four chromosomes. Each phase movement may have a number of conflicts. That is where one or more streams in a phase interfere with each other. Where there is interference, a penalty of one (1) is assigned to the phase. These are summed over the four phases. The higher the penalty accrued, the less fit the individual. The fitness function for this work will be:

$$F_n = \underset{1}{X}C_1 + \underset{1}{X}C_2 + \underset{1}{X}C_3 + \underset{1}{X}C_4 \qquad (4.1)$$

$$C_k = \underset{a\ b}{XX}(X_{ak}, X_{bk}); \quad a = 1, \ldots, n-1 \quad b = a+1, \ldots, n \quad k = 1 \text{ to } 4$$

$n = 6$ (*maximum number of traffic streamsassigned to a phase*) (4.2)

$$X_{ak}, X_{bk} = \begin{cases} 0 & \text{where} \end{cases}$$

where $X_{a,k}, X_{b,k} = Knot$ (4.3)

$\lfloor_1$ where $X_{a,k}, X_{b,k}\ 6 = Knot$

Part 1	(0,0)	(0,1)	(0,5)	(0,10)	(0,14)	(0,1)	(0,5)	(0,10)	(0,14)	(1,5)	(1,10)	(1,14)	(5,10)	(5,14)	(10,14)
Part 2	(0,2)	(0,3)	(0,8)	(0,11)	(0,13)	(2,3)	(2,8)	(2,11)	(2,13)	(3,8)	(3,11)	(3,13)	(8,11)	(8,13)	(11,13)
Part 3	(0,0)	(0,0)	(0,0)	(0,0)	(0,12)	(0,0)	(0,0)	(0,0)	(0,12)	(0,0)	(0,0)	(0,12)	(0,0)	(0,12)	(0,12)
Part 4	(0,0)	(0,4)	(0,6)	(0,7)	(0,9)	(0,4)	(0,6)	(0,7)	(0,9)	(4,6)	(4,7)	(4,9)	(6,7)	(6,9)	(7,9)

Table 4.10: Unique pairs in sample

4.1.8 Investigating if trajectories cross path

From Equation 4.1

$$C_k = \overset{n-1}{\underset{a}{\mathrm{XX}}}\overset{n}{\underset{b}{}}(X_{ak}, X_{bk}); \quad a = 1, \quad b = a + 1, \quad k = 1 \quad to \quad 4, \quad n = 6 \quad (4.4)$$

When $k = 1, \quad n = 6$

$$C_1 = \overset{5}{\underset{1}{\mathrm{XX}}}\overset{6}{\underset{2}{}}(X_{1,a}, X_{1,b})$$

$$C_1 = X_{1,1}, X_{1,2} + X_{1,1}, X_{1,3} + X_{1,1}, X_{1,4} + X_{1,1}, X_{1,5} + X_{1,1}, X_{1,6} + X_{1,2}, X_{1,3} + X_{1,2}, X_{1,4} +$$

$$X_{1,2}, X_{1,5} + X_{1,2}, X_{1,6} + X_{1,3}, X_{1,4} + X_{1,3}, X_{1,5} + X_{1,3}, X_{1,6} + X_{1,4}, X_{1,5} + X_{1,4}, X_{1,6} + X_{1,5}, X_{1,6} \quad (4.5)$$

When $k = 2, \quad n = 6$

$$C_2 = \overset{5}{\underset{1}{\mathrm{XX}}}\overset{6}{\underset{2}{}}(X_{2,a}, X_{2,b})$$

$$C_2 = X_{2,1}, X_{2,2} + X_{2,1}, X_{2,3} + X_{2,1}, X_{2,4} + X_{2,1}, X_{2,5} + X_{2,1}, X_{2,6} + X_{2,2}, X_{2,3} + X_{2,2}, X_{2,4} +$$

$$X_{2,2}, X_{2,5} + X_{2,2}, X_{2,6} + X_{2,3}, X_{2,4} + X_{2,3}, X_{2,5} + X_{2,3}, X_{2,6} + X_{2,4}, X_{2,5} + X_{2,4}, X_{2,6} + X_{2,5}, X_{2,6}$$

(4.6)

When $k = 3$, $n = 6$

$$C_3 = \underset{1}{\overset{5}{\mathrm{XX}}}\underset{2}{\overset{6}{}}(X_{3,a}, X_{3,b})$$

$$C_3 = X_{3,1},X_{3,2}+X_{3,1},X_{3,3}+X_{3,1},X_{3,4}+X_{3,1},X_{3,5}+X_{3,1},X_{3,6}+X_{3,2},X_{3,3}+X_{3,2},X_{3,4}+$$

$$X_{3,2},X_{3,5}+X_{3,2},X_{3,6}+X_{3,3},X_{3,4}+X_{3,3},X_{3,5}+X_{3,3},X_{3,6}+X_{3,4},X_{3,5}+X_{3,4},X_{3,6}+X_{3,5},X_{3,6}$$

(4.7)

When $k = 4$, $n = 6$

$$C_4 = \underset{1}{\overset{5}{\mathrm{XX}}}\underset{2}{\overset{6}{}}(X_{4,a}, X_{4,b})$$

$$C_3 = X_{4,1},X_{4,2}+X_{4,1},X_{4,3}+X_{4,1},X_{4,4}+X_{4,1},X_{4,5}+X_{4,1},X_{4,6}+X_{4,2},X_{4,3}+X_{4,2},X_{4,4}+$$

$$X_{4,2},X_{4,5}+X_{4,2},X_{4,6}+X_{4,3},X_{4,4}+X_{4,3},X_{4,5}+X_{4,3},X_{4,6}+X_{4,4},X_{4,5}+X_{4,4},X_{4,6}+X_{4,5},X_{4,6}$$

(4.8)

The knots (conflicts) identified at the intersection under study are in Table 4.11

A	(2,11)	traffic stream 2 and stream 11 meet at A
B	(3,11)	traffic stream 3 and stream 11 meet at B
C	(2,7)	traffic stream 2 and stream 7 meet at C
D	(3,4)	traffic stream 3 and stream 4 meet at D
E	(4,11)	traffic stream 4 and stream 11 meet at E
F	(8,11)	traffic stream 8 and stream 11 meet at F
G	(8,10)	traffic stream 8 and stream 10 meet at G
H	(3,10)	traffic stream 3 and stream 10 meet at H
I	(4,7)	traffic stream 4 and stream 7 meet at I
J	(2,4)	traffic stream 2 and stream 4 meet at J
K	(2,5)	traffic stream 2 and stream 5 meet at K

L	(7,10)	traffic stream 7 and stream 10 meet at L
M	(5,10)	traffic stream 5 and stream 10 meet at M
N	(5,7)	traffic stream 5 and stream 7 meet at N
O	(3,5)	traffic stream 3 and stream 5 meet at O
P	(5,8)	traffic stream 5 and stream 8 meet at P
Q	(3,8)	traffic stream 3 and stream 8 meet at Q
R	(5,14)	traffic stream 5 and stream 14 meet at R
S	(11,13)	traffic stream 11 and stream 13 meet at S
T	(7,11)	traffic stream 7 and stream 11 meet at T

Table 4.11: Knots: Points of traffic conflict

Suppose

$$Knot \quad A == X_{4,1}, X_{4,2}$$

$$Knot \quad B == X_{4,5}, X_{4,6}$$

$$Knot \quad C == X_{3,4}, X_{3,6},$$

A penalty of two (2) *will be placed on* C_4.

Pairs $X_{a,k}, X_{b,k}$ *are compared to the knots A to T. Equivalence of stream pairs to any knot text attracts a penalty of one* (1). *The penalties are summed to determine the fitness of the chromosome.* (4.9)

Total conflict (fitness) in the chromosome 1 is the sum of conflicts or

	Pairs	0,1	0,5	0,10	0,14	0,14	0,1	0,14	0,5	1,5	1,10	0,0	1,14	5,10	5,14	10,14	TOTAL
Knots																	
2,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

2,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,10		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
5,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,14		0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
11,13		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FITNESS																	2

Table 4.12: Testing for Fitness in Chromosome 1

interferences in all its four parts.

$$F_n = XC_1^6 + XC_2^6 + XC_3^6 + XC_4^6$$

$$F_1 = 2 + 7 + 0 + 1 = 10$$

(4.10)

The process is repeated for fifty (50) randomly generated chromosomes

(traffic movement sequences).

4.1.9 Testing for interference in Chromosome 1

Finding the fitness for Chromosome 1 is outlined in Table 4.12

4.1.10 Testing for interference in Chromosome 2

Finding the fitness for Chromosome 2 is outlined in Table 4.13

	Pairs	0,2	0,3	0,8	0,11	0,13	2,3	2,8	2,11	2,13	3,8	3,11	3,13	8,11	8,13	11,13	TOTAL
Knots																	
2,11		0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
3,11		0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
2,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8,11		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
3,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,4		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
7,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,10		0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1
5,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,8		0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
5,14		0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
11,13		0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

FITNESS	7

Table 4.13: Testing for Fitness in Chromosome 2

	Pairs	0,0	0,0	0,0	0,0	0,12	0,0	0,0	0,0	0,12	0,0	0,0	0,12	0,0	0,12	0,12	TOTAL
Knots																	
2,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,14		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11,13		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FITNESS																	0

Table 4.14: Testing for Fitness in Chromosome 3

	Pairs	0,0	0,4	0,6	0,7	0,9	0,4	0,6	0,7	0,9	4,6	4,7	4,9	6,7	6,9	7,9	TOTAL
Knots																	
2,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4,7		0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1
2,4		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7,11		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,10		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,7		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,5		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3,8		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5,14		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11,13		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
FITNESS																	1

Table 4.15: Testing for Fitness in Chromosome 4

4.1.11 Testing for interference in Chromosome 3

Finding the fitness for Chromosome 3 is outlined in Table 4.14

4.1.12 Testing for interference in Chromosome 4

Finding the fitness for Chromosome 4 is outlined in Table 4.15

4.1.13 Test Results

Points of conflicts identified in the intersection are named knots. About twenty (20) knots were identified. The knots are grouped into three depending on the intensity of damage to lives and property should there be collision. This can be found in Appendix B.

- A penalty of one is given for level 1 conflicts. This level of conflict can be tolerated as there will be little or no damage during collision.
- A penalty of two is given for level 2 conflicts. This level of conflict is not tolerated but do not cause severe damage
- A penalty of there is given for level 3 conflicts. This level of conflict must be avoided completely.

The fitness of each chromosome over three generations is recorded in the Table 4.16

4.1.14 Selection

Thirty Chromosomes with the best fitness are selected for recombination (cross over).

4.1.15 Cross Over

Uniform Like Crossover (ULX) was used in this work. This crossover operator was proposed by **?**. It works as follows. First, all items assigned to the same position in both parents are copied to this position in the child. Second, the unassigned positions of a permutation are scanned from left to right: for the unassigned

position, an item is chosen randomly, uniformly from those in the parents if they are not yet included in the child. Third, remaining items are assigned at random.

The thirty (30) selected chromosomes are recombined (cross over) using ULX to produce fifteen (15) new offsprings. The fifteen (15) offsprings are added to the parents to make forty five chromosomes. An addition of five (5) chromosomes are generated at random and added to make the new generation population

fifty (50).

Table 4.17 to Table 4.26 shows recombinations of twenty chromosomes to form ten offsprings.

The Algorithm is run over three generations. The Chromosomes that have the least conflict are considered as as a potential order to sequence traffic phases at the intersection. In this trial run, the minimum conflict is 2.

Conflict	First Generation	Second Generation	Third Generation
KnotM1	2	0	4
KnotM2	0	7	2
KnotM3	0	2	2
KnotM4	4	0	3
KnotM5	4	6	8
KnotM6	11	10	8
KnotM7	11	6	7
KnotM8	11	6	9
KnotM9	9	11	9
KnotM10	10	7	6
KnotM11	9	8	9
KnotM12	6	4	12
KnotM13	8	8	8
KnotM14	10	8	9
KnotM15	10	10	7
KnotM16	6	7	7
KnotM17	9	9	8
KnotM18	4	14	9
KnotM19	8	8	9
KnotM20	11	9	6
KnotM21	11	1	9
KnotM22	6	7	7
KnotM23	7	9	8
KnotM24	9	6	10
KnotM25	6	7	9
KnotM26	8	9	10
KnotM27	11	11	6
KnotM28	10	8	10

KnotM29	8	4	9
KnotM30	11	9	11
KnotM31	6	11	9
KnotM32	7	7	7
KnotM33	8	9	8
KnotM34	6	6	10
KnotM35	8	7	9
KnotM36	11	9	10
KnotM37	10	11	6
KnotM38	2	8	10
KnotM39	8	4	9
KnotM40	11	9	11
KnotM41	8	7	4
KnotM42	15	8	9
KnotM43	10	6	7
KnotM44	5	5	4
KnotM45	6	9	12
KnotM46	15	9	8
KnotM47	8	6	7
KnotM48	10	9	7
KnotM49	9	9	10
KnotM50	5	17	7
Min (Best Fitness)	0	0	2
Max (Worst Fitness)	15	17	12

Table 4.16: Testing fitness of Chromosomes across three generations

Parent1	7 14 0 0 12 0 0 3 2 1 13 0 8 0 6 10 0 0 11 5 0 0 4 9
Parent2	0 6 0 11 0 1 3 0 13 0 10 0 9 5 14 4 7 12 2 0 0 0 8 0
Offspring1	7 0 0 0 5 3 0 8 0 6 13 0 10 12 0 11 14 4 2 9 0 0 0 1

Table 4.17: Parent 1 and Parent 2 crossed over to produce offspring 1 using Uniformlike Crossover (ULX)

Parent3	0 0 11 6 13 5 0 0 0 0 8 7 12 2 0 10 0 3 4 1 9 0 14 0
Parent4	0 0 8 0 0 5 13 0 2 4 9 3 0 0 0 14 1 7 10 0 12 11 0 6
Offspring2	0 0 0 0 10 5 1 0 4 0 13 0 0 2 0 12 7 0 6 11 3 14 8 9

Table 4.18: Parent 3 and Parent 4 crossed over to produce offspring 2 using Uniformlike Crossover (ULX)

Parent5	0 8 0 2 9 0 10 1 5 0 0 0 13 0 0 7 0 4 6 14 12 3 0 11
Parent6	6 5 0 0 0 0 10 2 14 12 0 11 0 3 8 9 0 0 13 0 4 0 1 7
Offspring3	13 11 0 14 9 0 10 0 7 5 0 6 2 0 4 3 0 0 0 12 8 0 0 1

Table 4.19: Parent 5 and Parent 6 crossed over to produce offspring 3 using Uniformlike Crossover (ULX)

Parent7	0 1 12 3 10 0 6 8 7 0 11 0 13 0 9 0 2 14 0 4 0 0 0 5
Parent8	0 2 0 0 10 12 0 0 0 4 3 14 0 13 8 7 11 0 9 1 6 0 0 5
Offspring4	0 4 0 11 10 0 0 14 12 0 0 2 1 7 9 6 0 8 13 0 3 0 0 5

Table 4.20: Parent 7 and Parent 8 crossed over to produce offspring 4 using Uniformlike Crossover (ULX)

Parent9	3 0 10 8 1 9 5 4 11 0 0 12 13 0 0 7 0 0 0 0 6 2 14 0
Parent10	0 13 0 8 5 11 14 3 0 7 0 0 0 0 9 4 2 0 0 0 6 1 12 10
Offspring5	1 5 14 8 0 12 0 2 7 0 0 13 0 0 9 0 4 0 0 0 6 3 10 11

Table 4.21: Parent 9 and Parent 10 crossed over to produce offspring 5 using Uniformlike Crossover (ULX)

Parent11	0 13 6 1 0 11 14 2 0 0 10 0 9 12 3 8 0 4 0 7 0 0 0 5
Parent12	14 0 6 4 0 0 0 13 0 0 7 0 12 0 10 3 11 0 1 5 9 0 2 8
Offspring6	12 9 6 5 0 0 1 4 0 0 14 0 10 8 13 2 3 11 7 0 0 0 0 0

Table 4.22: Parent 11 and Parent 12 crossed over to produce offspring 6 using Uniformlike Crossover (ULX)

Parent13	0 0 10 12 0 1 0 0 2 14 4 13 7 6 0 0 5 8 0 9 0 0 3 11
Parent14	0 7 0 0 12 0 4 8 5 0 9 10 14 0 1 0 6 2 11 0 13 0 3 0
Offspring7	0 10 6 12 5 0 0 14 2 8 4 11 0 1 0 0 7 0 0 9 0 0 3 13

Table 4.23: Parent 13 and Parent 14 crossed over to produce offspring 7 using Uniformlike Crossover (ULX)

Parent15	14 3 0 10 0 9 5 12 0 1 11 2 8 6 4 0 0 0 7 0 13 0 0 0
Parent16	0 4 7 0 9 10 0 11 0 6 12 3 13 0 2 0 8 0 0 14 0 0 1 5
Offspring8	0 3 11 2 0 0 13 14 0 4 9 8 1 7 0 0 10 0 5 12 6 0 0 0

Table 4.24: Parent 15 and Parent 16 crossed over to produce offspring 8 using Uniformlike Crossover (ULX)

Parent17	0 12 14 5 0 4 13 3 10 9 0 0 6 8 0 0 2 0 7 0 0 11 1 0
Parent18	0 9 0 5 0 0 0 0 13 0 1 10 11 2 12 4 8 14 7 0 6 3 0 0
Offspring9	0 8 1 5 0 0 14 0 3 6 12 0 0 11 4 0 13 2 7 0 0 9 10 0

Table 4.25: Parent 17 and Parent 18 crossed over to produce offspring 9 using Uniformlike Crossover (ULX)

Parent19	5 7 1 0 11 0 2 0 3 0 4 0 0 14 0 6 0 8 9 0 13 10 0 12
Parent20	10 11 1 0 5 0 8 3 0 0 0 7 12 0 13 0 9 4 2 6 0 14 0 0
Offspring10	5 12 1 0 14 0 0 0 0 0 3 9 8 10 13 2 4 6 7 0 0 11 0 0

Table 4.26: Parent 19 and Parent 20 crossed over to produce offspring 10 using Uniformlike Crossover (ULX)

Chapter 5

Result, Conclusion and Recommendations

5.1 Results

The output of the program is evaluated for both computational efficiency and accuracy. The Chromosomes that meet the minimum conflict will be used by the traffic engineer for implementation. Where more than one sequence meets the required criteria, the discretion of the traffic engineer will come to play on which movement sequence will be implemented.

The program resulted in chromosomes 2 meeting the criteria or minimum or best minimum conflict which is two (2).

5.1.1 Interpretation of Chromosome 2 result

In Phase1 (Table 5.2), there will be three traffic streams traversing the intersection when the right of way is given.

They are:

- stream 11 traversing from east to west.
- stream 13 traversing from west to west (traffic making U turn from west)
- stream 14 traversing from east to east (traffic making U turn from east)

Chromosome 2	0 0 11 0 13 14 0 6 1 12 5 3 0 0 0 2 10 4 0 9 8 0 0 7
Phase 1	0 0 11 0 13 14
Phase2	0 6 1 12 5 3

Phase3	0 0 0 2 10 4
Phase4	0 9 8 0 0 7

Table 5.1: Least conflict result

Phase 1	0 0 11 0 13 14

Table 5.2: Phase 1 of test result

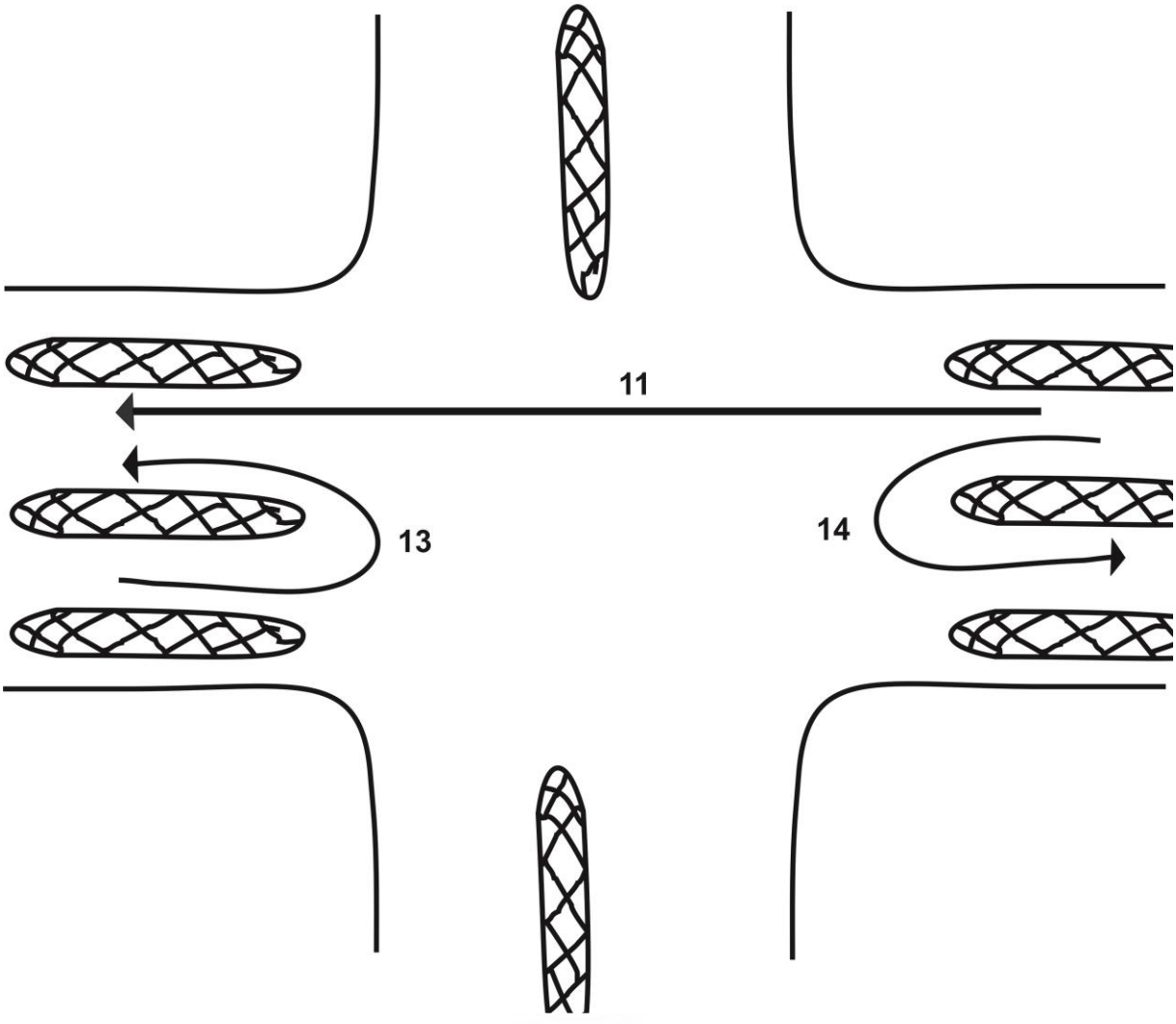


Figure 5.1: Test Result of Phase 1

Phase2	0 6 1 12 5 3

Table 5.3: Phase 2 of test result

Phase3	0 0 0 2 10 4

Table 5.4: Phase 3 of test result

Phase4	0 9 8 0 0 7

Table 5.5: Phase 4 of test result

In Phase2, there will be five traffic streams traversing the intersection when the right of way is given.

They are:

- stream 6 traversing from west to south
- stream 1 traversing from north to west
- stream 12 traversing from east to north
- stream 5 traversing from west to east
- stream 3 traversing from north to south

In Phase3, there will be three traffic streams traversing the intersection when the right of way is given.

They are:

- stream 2 traversing from north to south
- stream 10 traversing from east to south
- stream 4 traversing from west to north

There is a conflict between stream 2 and stream 4.

In Phase4, there will be three traffic streams traversing the intersection when the right of way is given.

They are:

- stream 9 traversing from south to east
- stream 8 traversing from south to north

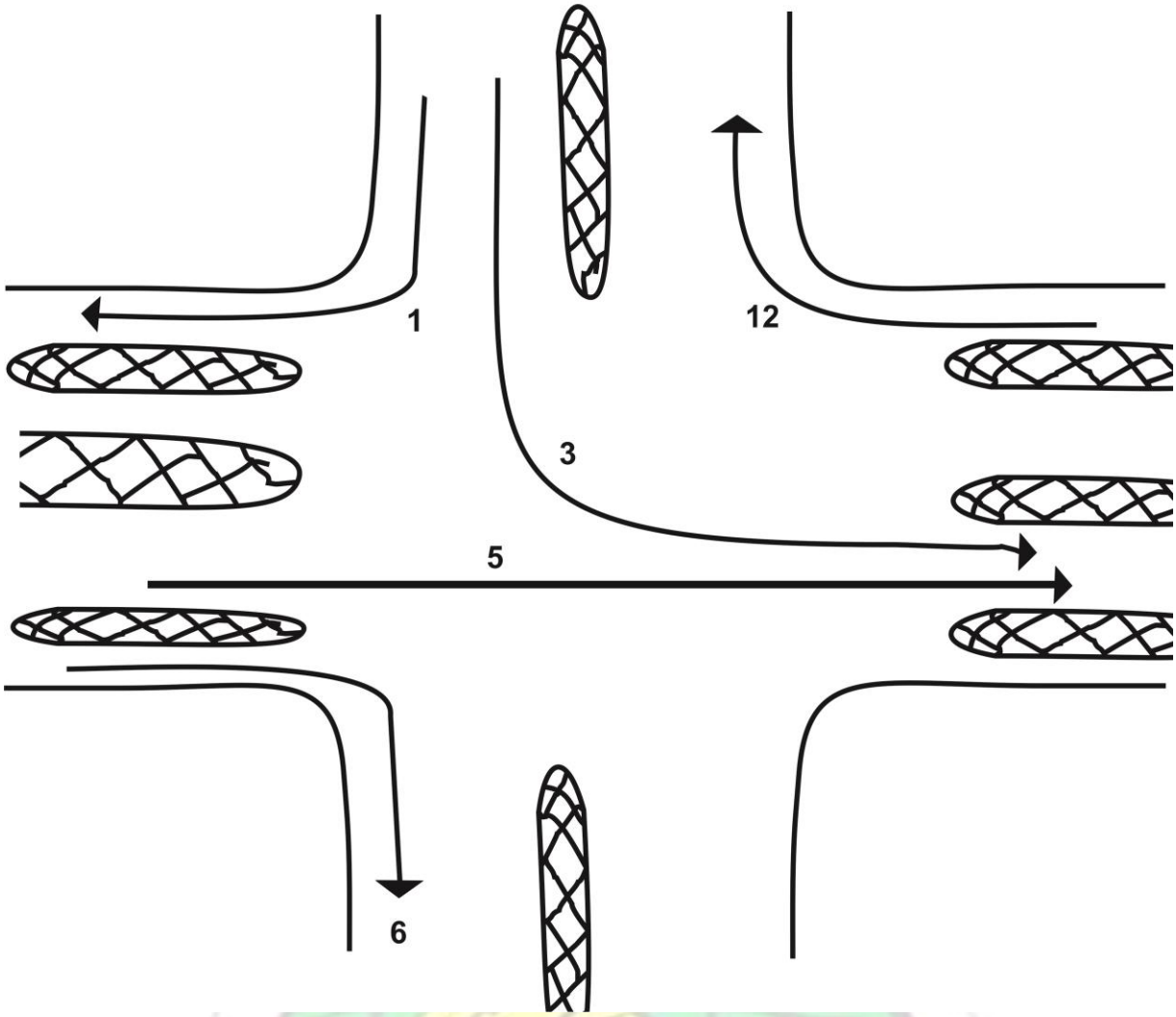


Figure 5.2: Test Result of Phase 2

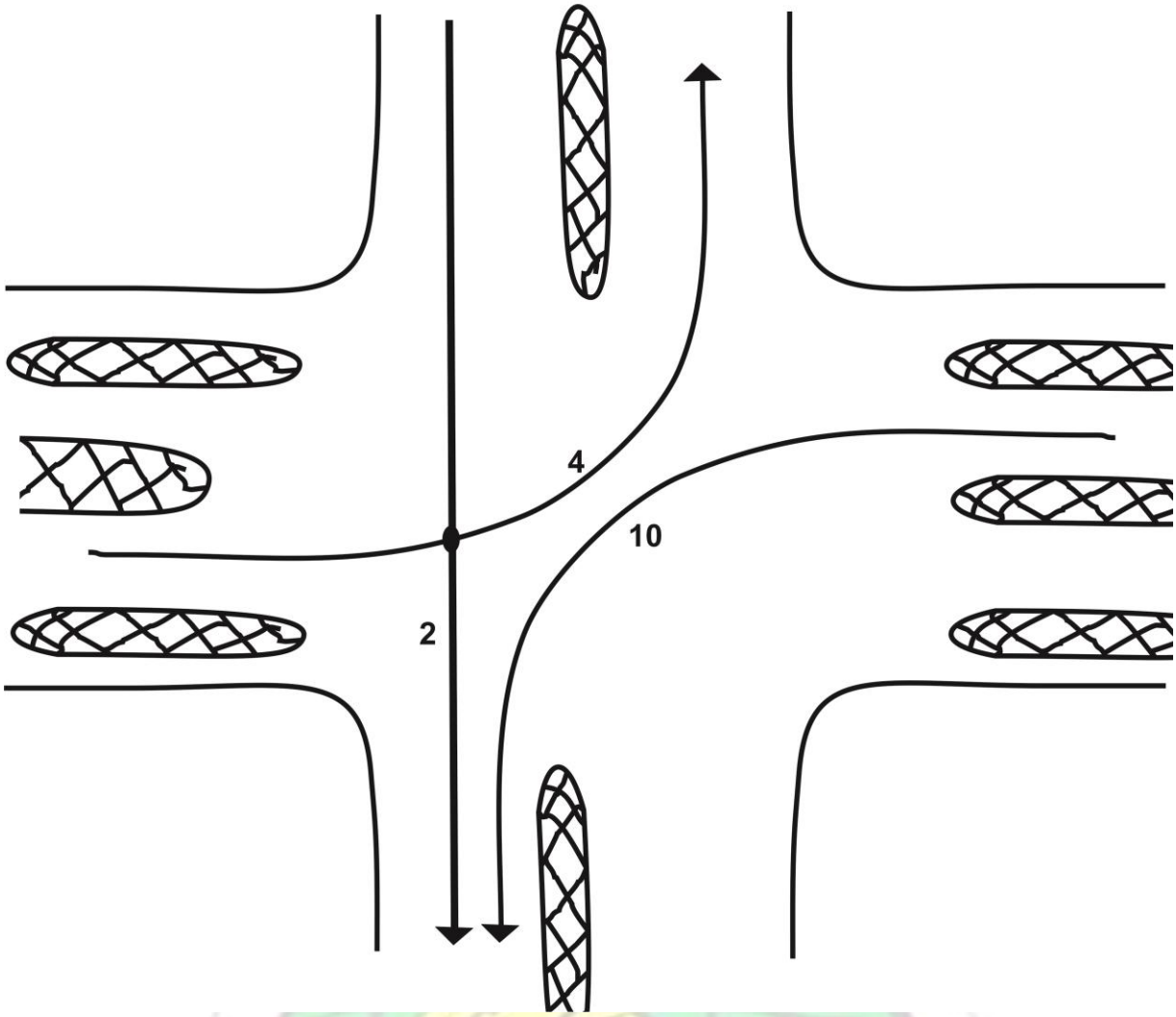


Figure 5.3: Test Result of Phase 3

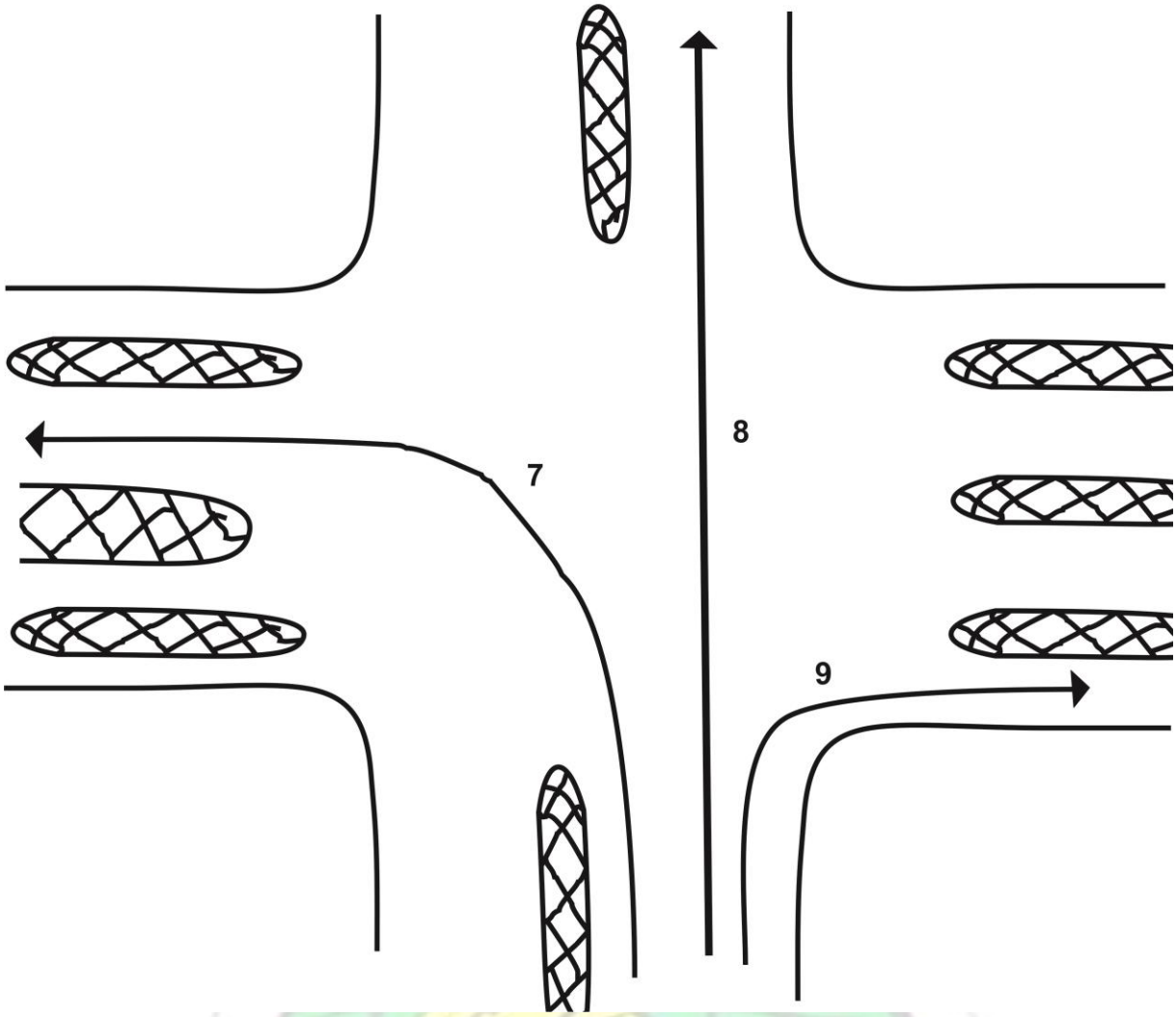


Figure 5.4: Test Result of Phase 4

- stream 7 traversing from south to west

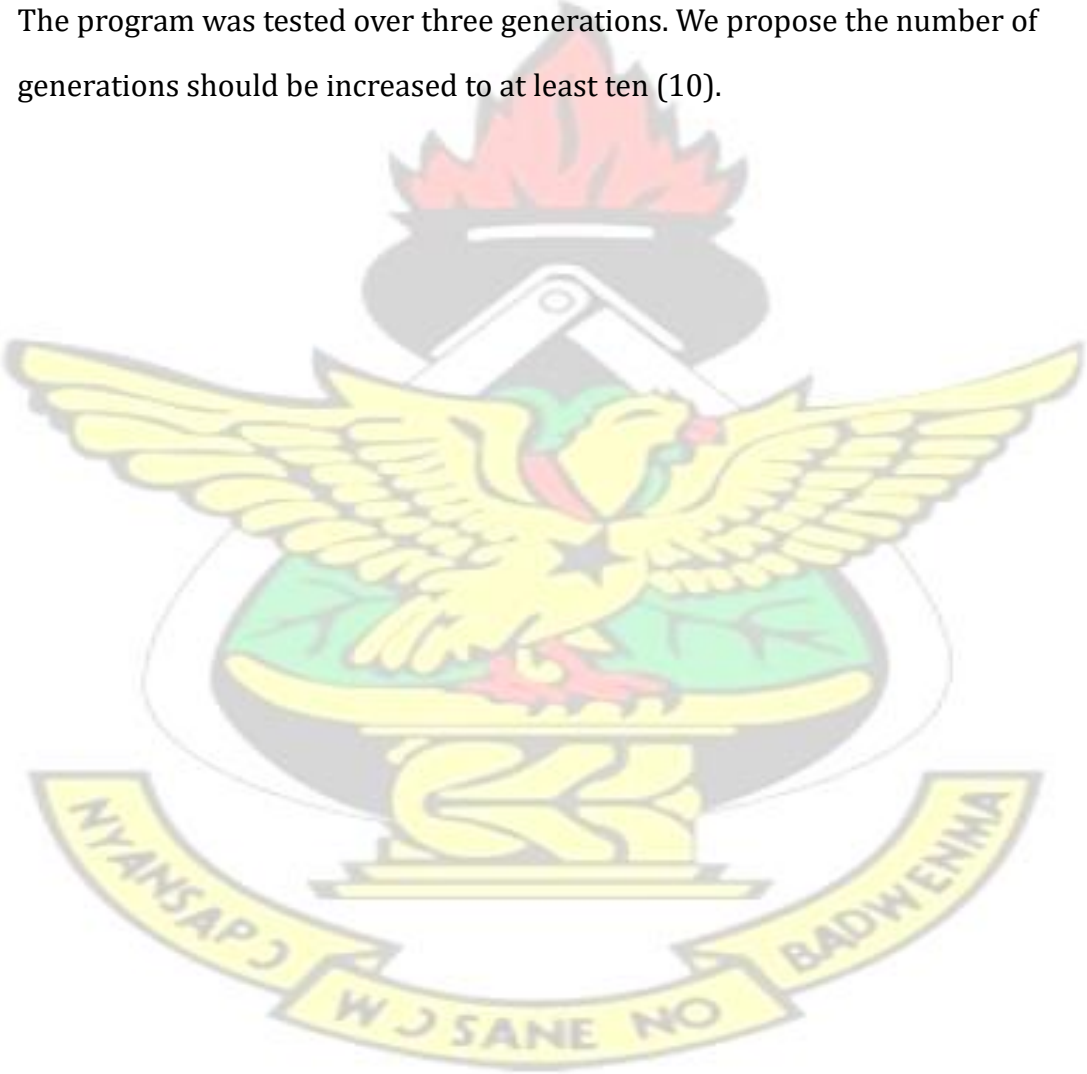
5.2 Conclusion

The current traffic systems at the La Paz intersection performs well outside rush hours, when traffic volumes are not much. However, at peak times, the systems can no longer regulate the traffic and this results in chaos, long queues, congestion, accidents, etc. At this stage the optimised traffic flow sequence obtained in this study is the best answer.

5.3 Recommendations

There is room for more improvement on this work.

- A bigger population than should be considered in future works
- This work is purely mathematical and may not be helpful to those less inclined mathematically. In future work, simulations could be added for simplicity and further clarity.
- The program was tested over three generations. We propose the number of generations should be increased to at least ten (10).



REFERENCES

Ai-Junkie (2015). Genetic algorithm in plain english. *http://www.ai-junkie.com/ga/intro/gat1.html*, page 1.

Amegashie, M. (2012). Canage on george walker bush n1 high-way. *http://mawulolo.blogspot.com/2012/02/geroge-walker-bush-n1-highwaycarnage.html*.

Andrea Vogel, C. G. and von Seelen, W. (2000). Evolutionary algorithms for optimizing traffic signal operation.

Axelrod, R. (1984). The evolution of cooperation. *Basic Books*.

Axelrod, R. and Dion, D. (1988). The further evolution of cooperation. *Science*, 242(4884):1385-1390.

Axelrod, R. and Kaufmann, M. (1987). The evolution of strategies in the iterated prisoner's dilemma.

Baricelli, N. A. (1957). Symbiogenetic evolution processes realized by artificial methods. *Methodos*, 9:35–36, 143–182.

Baricelli, N. A. (1962). Numerical testing of evolution theories. *ACTA Biotheoretica*, 16:69–126.

Bhattacharya, P., V. M. R. (2005). Optimization for pedestrian and vehicular delay in a signal network. in transportation research record 1939, trb, national research council, washington, d.c. pages 115–122.

Bledsoe, W. W. (1961). The use of biological concepts in the analytical study of systems. paper presented at orsa-tims national meeting, san francisco.

Box, G. E. P. (1957). Evolutionary operation: A method for increasing industrial productivity. *Journal of the Royal Statistical Society*, C6 No 2:81–101.

Bremermann, H. J. (1962). *Optimization through evolution and recombination.*

Davis, L. (1985). Applying algorithms to epistatic domains. In *Proc. Int. Joint Conf. on Artifical Intelligence*, page 162-164.

Duerr, P. (2000). Dynamic right-of-way for transit vehicles: Integrated modeling approach for optimizing signal control on mixed traffic arterials. *In Transportation Research Record: Journal of the Transportation Research Board, No. 1731. TRB, National Research Council, Washington, D.C.*, pages 31–39.

Fogel, L. J., O. A. J. and Walsh, M. J. (1966). Artificial intelligence through simulated evolution.

Foy, M., R. F. B. and Goldberg., D. E. (1992). Signal timing determination using genetic algorithms. in transportation research record 1365, trb, national research council, washington, d.c. pages 108–115.

Fraser, A. S. (1957a). Simulation of genetic systems by automatic digital computers: I. introduction. *Australian Journal of Biological Science*, 10:484–491.

Fraser, A. S. (1957b). Simulation of genetic systems by automatic digital computers: Ii. effects of linkage on rates of advance under selection. *Australian Journal of Biological Science*, 10:492–499.

Friedman, G. J. (1959). Digital simulation of an evolutionary process. *General Systems Yearbook*, 4:171–184.

Girianna, M. and Benekohal, R. F. (2002). Dynamic signal coordination for networks with oversaturated intersections. *In Transportation Research Record 1811, TRB, National Research Council, Washington, D.C.,*, pages 122–130.

Girianna, M. and Benekohal, R. F. (2004). Using genetic algorithms to design signal coordination for oversaturated networks. *In Journal of ITS: Technology, Planning, and Operations,*, 8(2):117–129.

Goldberg, D. E. and Lingle, R. (1985). Alleles, loci, and the tsp. In Goldberg, D. E., Lingle, R., . A. l., and the TSP, in: Proc. 1st Int. Conf. on Genetic Algorithms, p. ., editors, *1st Int. Conf. on Genetic Algorithms,*, page 154-159.

Holland, J. H. (1975). Adaptation in natural and artificial systems. *University of Michigan Press*, Second edition: MIT Press, 1992.

HA˜ 14sken, M., C. G. and Vogel, A. (2000). Fast adaptation of the solution of differential equations to changing constraints. *In Proceedings of the Second International ICSC Symposium on Neural Computation (NC'2000). ICSC Academic Press*.

Ishaque, M. M., N. R. (2007). Trade-offs between vehicular and pedestrian traffic using micro-simulation methods. *Transportation Policy 14*, pages 124–138.

Ishaque, M. M., N. R. B. (2005). Multimodal microsimulation of vehicle and pedestrian signal timings. in transportation research record 1939, trb, national research council, washington, d.c. pages 107–114.

Keesing, R. and Stokes, D. G. (1991). Evolution and learning in neural networks: The number and distribution of learning trails affect the rate of evolution. *In Neural Information Processing Systems*, 3.

Kok Khiang Tan, M. K. and Yusof, R. (1996). *Intelligent Traffic Lights Control by Fuzzy Logic*, volume 9 of *Malaysian Journal of Computer Science*.

Kumara Sastry, D. G. and Kendall, G. (2005). *Search Methodologies*.

Li, M., A. W. K. M. N. H. A. (2009). Traffic signal optimization strategy considering both vehicular and pedestrian flows. presented at the 89th annual meeting of the transportation research board, washington, d.c.

Lindgren, K. (1992). *Evolutionary phenomena in simple dynamics. In C. G.*

Langton, C. Taylor, J. D. Farmer, and S. Rasmussen, eds., Artificial Life II. Addison-Wesley.

Martin, F. G. and Cockerham, C. C. (1960). High speed selection studies. *In O. Kempthorne, ed., Biometrical Genetics. Pergamon.*

Melanie, M. (1999). *An Introduction to Genetic Algorithm*. Massachusetts Institute of Technology.

Noland, R. (1996). Pedestrian travel times and motor vehicle traffic signals. *Transportation Research Record: Journal of the Transportation Research Board. (1553). Transportation Research Board of the National Academics. Washington.*, pages 28–33.

Pappis, C. P. and Mamdani, E. H. (Vol. SMC-7, No. 10, October 1977). A fuzzy logic controller for a traffic junction. *IEEE Transactions Systems, Man, and Cybernetics*, 7(10):707–717.

Park, B., J. M. and Urbanik, T. (1999). Traffic signal optimization program for oversaturated conditions. in transportation research record 1683, trb, national research council, washington, d.c. pages 133–142.

Rechenberg (1965). *Evolutions Strategies*.

Rechenberg (1973). *Evolution Strategies*.

Rechenberg, I. (1994). Evolutionsstrategie ’94. stuttgart: Friedrich frommann holzboog verlag.

Reed, J., T. R. and Barricelli, N. A. (1967). Simulation of biological evolution and machine learning. *Journal of Theoretical Biology*, 17:319–342.

Ringel, R. . (1995). Schaffung eines tools zur modellierung und simulation von prozessen des strabenverkehrs. Master’s thesis, Technische Universitat Dresden.

Schnabel, W. and Lohse, D. (1997). Grundlagen der strabenverkehrstechnik und der verkehrsplanung. *Verlag fA˜41r das Bauwesen*, 1.

Schwefel, H. (1975). *Evolutionsstrategie und numerische Optimierung*. PhD thesis, Technische Universitat Berlin.

Schwefel, H. (1977). Numerische optimierung von computer - modellen mittels der evolutionstrategie.

Spears, W. M. and De Jong, K. A. (1994). On the virtues of parameterized uniform crossover,. In *4th Int. Conf. on Genetic Algorithms.*

Syswerda, G. (1989). *Uniform crossover in genetic algorithms, in: Proc. 3rd Int. Conf. on Genetic Algorithms*.

TRB (2007). Artificial intelligence in transportation (information for application). artificial intelligence and advanced computing applications committee, washington, d.c.

Virkler, M. R. (1998a). Pedestrian compliance effects on signal delay. *In Transportation Research Record 1636, TRB, National Research Council, Washington, D.C.,*, pages 88–91.

Virkler, M. R. (1998b). Prediction and measurement of travel time along pedestrian routes. *In Transportation Research Record 1636, TRB, National Research Council, Washington, D.C.,*, pages 37–42.

Virkler, M. R. (1998c). Scramble and crosswalk signal timing. *In Transportation Research Record 1636, TRB, National Research Council, Washington, D.C.,*, pages 83–87.

Virkler, M. R. (1998d). Signal coordination benefits for pedestrians. *In Transportation Research Record 1636, TRB, National Research Council, Washington, D.C.,*, pages 77–82.

Appendix A

Chromosome No.	Chromosome
Chromosome 1	11 7 0 0 1 0 8 6 3 2 0 9 5 0 0 13 4 14 12 10 0 0 0 0
Chromosome 2	0 7 2 0 13 0 3 6 4 1 0 8 12 11 0 14 0 5 9 10 0 0 0 0
Chromosome 3	11 0 0 14 12 7 0 9 0 5 6 13 10 0 3 0 0 4 0 0 2 0 8 1
Chromosome 4	3 2 0 10 7 1 4 0 0 12 5 0 0 0 0 0 13 8 11 14 0 9 6 0
Chromosome 5	14 5 0 4 1 0 10 0 0 2 3 7 12 0 8 0 9 0 11 0 0 13 6 0
Chromosome 6	0 1 3 0 7 4 9 6 11 0 12 0 0 0 14 0 8 5 0 2 0 0 10 13
Chromosome 7	0 0 0 9 11 14 7 0 0 5 13 6 0 0 10 12 0 3 0 8 0 2 4 1
Chromosome 8	9 8 5 0 6 13 1 3 0 10 0 0 0 11 0 7 12 2 0 4 14 0 0 0
Chromosome 9	0 2 13 0 12 10 11 3 14 9 5 0 0 0 4 7 0 0 0 1 8 6 0 0
Chromosome 10	0 4 13 0 7 14 3 5 0 8 10 2 9 0 6 11 0 0 0 0 0 0 1 12
Chromosome 11	6 0 1 12 11 0 0 3 0 5 0 10 4 7 14 2 8 9 0 0 0 0 0 13
Chromosome 12	0 7 0 0 12 0 4 8 5 0 9 10 14 0 1 0 6 2 11 0 13 0 3 0
Chromosome 13	0 8 0 2 9 0 10 1 5 0 0 0 13 0 0 7 0 4 6 14 12 3 0 11
Chromosome 14	0 8 0 12 7 0 0 0 5 0 4 13 0 0 1 9 0 6 14 10 11 0 3 2
Chromosome 15	0 9 2 0 0 11 1 5 12 7 0 0 3 0 0 13 4 6 0 0 10 14 8 0
Chromosome 16	0 0 0 0 11 0 10 7 9 0 0 0 14 0 13 2 4 8 0 3 12 5 6 1
Chromosome 17	0 7 12 0 0 1 8 3 11 6 0 0 2 0 10 0 4 9 0 0 5 14 13 0
Chromosome 18	0 4 0 11 10 0 0 14 12 0 0 2 1 7 9 6 0 8 13 0 3 0 0 5
Chromosome 19	0 2 0 0 10 12 0 0 0 4 3 14 0 13 8 7 11 0 9 1 6 0 0 5
Chromosome 20	0 7 0 6 10 4 0 2 12 14 1 0 3 8 0 13 0 0 11 9 0 0 0 5
Chromosome 21	0 0 11 6 13 5 0 0 0 0 8 7 12 2 0 10 0 3 4 1 9 0 14 0
Chromosome 22	10 11 1 0 5 0 8 3 0 0 0 7 12 0 13 0 9 4 2 6 0 14 0 0
Chromosome 23	0 0 11 9 1 4 5 14 0 0 0 7 12 0 3 0 2 0 6 10 13 8 0 0
Chromosome 24	0 13 6 1 0 11 14 2 0 0 10 0 9 12 3 8 0 4 0 7 0 0 0 5
Chromosome 25	7 14 0 0 12 0 0 3 2 1 13 0 8 0 6 10 0 0 11 5 0 0 4 9
Chromosome 26	14 3 0 10 0 9 5 12 0 1 11 2 8 6 4 0 0 0 7 0 13 0 0 0
Chromosome 27	0 0 9 0 1 0 14 0 0 8 2 4 0 13 12 6 10 7 0 11 5 0 0 3
Chromosome 28	7 0 5 0 0 1 0 9 0 4 3 11 10 0 13 12 0 2 14 0 6 0 0 8
Chromosome 29	0 6 0 2 0 4 12 5 13 0 0 10 7 3 0 0 14 0 1 0 8 0 9 11
Chromosome 30	0 4 7 0 9 10 0 11 0 6 12 3 13 0 2 0 8 0 0 14 0 0 1 5
Chromosome 31	0 6 0 9 0 13 3 7 14 0 5 0 11 12 0 0 10 0 2 8 1 0 0 4
Chromosome 32	6 5 0 0 0 0 10 2 14 12 0 11 0 3 8 9 0 0 13 0 4 0 1 7
Chromosome 33	0 6 0 11 0 1 3 0 13 0 10 0 9 5 14 4 7 12 2 0 0 0 8 0
Chromosome 34	11 0 0 0 0 0 14 3 6 2 1 5 7 4 12 0 0 13 8 0 10 0 9 0
Chromosome 35	10 6 2 11 13 1 14 0 12 8 5 0 9 0 0 0 4 0 0 0 0 7 0 3
Chromosome 36	0 8 0 2 9 0 10 1 5 0 0 0 13 0 0 7 0 4 6 14 12 3 0 11
Chromosome 37	6 5 0 0 0 0 10 2 14 12 0 11 0 3 8 9 0 0 13 0 4 0 1 7
Chromosome 38	13 11 0 14 9 0 10 0 7 5 0 6 2 0 4 3 0 0 0 12 8 0 0 1
Chromosome 39	0 1 12 3 10 0 6 8 7 0 11 0 13 0 9 0 2 14 0 4 0 0 0 5
Chromosome 40	0 2 0 0 10 12 0 0 0 4 3 14 0 13 8 7 11 0 9 1 6 0 0 5
Chromosome 41	14 0 6 4 0 0 0 13 0 0 7 0 12 0 10 3 11 0 1 5 9 0 2 8
Chromosome 42	12 9 6 5 0 0 1 4 0 0 14 0 10 8 13 2 3 11 7 0 0 0 0 0
Chromosome 43	0 0 10 12 0 1 0 0 2 14 4 13 7 6 0 0 5 8 0 9 0 0 3 11
Chromosome 44	0 7 0 0 12 0 4 8 5 0 9 10 14 0 1 0 6 2 11 0 13 0 3 0
Chromosome 45	0 10 6 12 5 0 0 14 2 8 4 11 0 1 0 0 7 0 0 9 0 0 3 13

Chromosome 46	14 0 6 4 0 0 0 13 0 0 7 0 12 0 10 3 11 0 1 5 9 0 2 8
Chromosome 47	0 3 11 2 0 0 13 14 0 4 9 8 1 7 0 0 10 0 5 12 6 0 0 0
Chromosome 48	0 5 8 7 0 0 12 10 0 13 4 14 0 0 2 0 0 0 3 11 1 0 9 6
Chromosome 49	0 0 1 14 0 0 12 6 4 7 0 5 11 9 0 2 3 0 13 10 0 0 0 8
Chromosome 50	5 12 1 0 14 0 0 0 0 0 3 9 8 10 13 2 4 6 7 0 0 11 0 0

Table 5.6: Initial Population

Appendix B

Knots: Provision was made for the introduction of other knots in the future. The addition of (NaN, NaN) makes it easy for such replacements, say (4, 9).

Knots used in this work includes: 2,11, 3 11, 2 7, 4 11, 8 11, 8 10, 3 10, 4 7, 2 4, 2 5, 7 10, 5 10, 5 7, 3 5, 5 8, 3 8, 5 14, 11 13, 7 11, 3 4, 8 11, 2 5, 7 10, 4 7, 3 4, 3 10, 7 11, 5 14, 5 7, 5 10, 3 11, 2 11, 2 5, 8 11, 5 8, 4 11, 3 5, 8 10, 2 4, 5 14, 3 8, NaN NaN, NaN NaN, NaN NaN, NaN NaN, NaN NaN, NaN NaN, NaN NaN, NaN NaN, NaN NaN,

Appendix C

Chromosome No.	Chromosome
Chromosome 1	7 14 0 0 12 0 0 3 2 1 13 0 8 0 6 10 0 0 11 5 0 0 4 9
Chromosome 2	0 6 0 11 0 1 3 0 13 0 10 0 9 5 14 4 7 12 2 0 0 0 8 0
Chromosome 3	7 0 0 0 5 3 0 8 0 6 13 0 10 12 0 11 14 4 2 9 0 0 0 1
Chromosome 4	0 0 11 6 13 5 0 0 0 0 8 7 12 2 0 10 0 3 4 1 9 0 14 0
Chromosome 5	0 0 8 0 0 5 13 0 2 4 9 3 0 0 0 14 1 7 10 0 12 11 0 6
Chromosome 6	0 0 0 0 10 5 1 0 4 0 13 0 0 2 0 12 7 0 6 11 3 14 8 9
Chromosome 7	0 8 0 2 9 0 10 1 5 0 0 0 13 0 0 7 0 4 6 14 12 3 0 11
Chromosome 8	6 5 0 0 0 0 10 2 14 12 0 11 0 3 8 9 0 0 13 0 4 0 1 7
Chromosome 9	13 11 0 14 9 0 10 0 7 5 0 6 2 0 4 3 0 0 0 12 8 0 0 1
Chromosome 10	0 1 12 3 10 0 6 8 7 0 11 0 13 0 9 0 2 14 0 4 0 0 0 5
Chromosome 11	0 2 0 0 10 12 0 0 0 4 3 14 0 13 8 7 11 0 9 1 6 0 0 5
Chromosome 12	0 4 0 11 10 0 0 14 12 0 0 2 1 7 9 6 0 8 13 0 3 0 0 5
Chromosome 13	3 0 10 8 1 9 5 4 11 0 0 12 13 0 0 7 0 0 0 0 6 2 14 0
Chromosome 14	0 13 0 8 5 11 14 3 0 7 0 0 0 0 9 4 2 0 0 0 6 1 12 10
Chromosome 15	1 5 14 8 0 12 0 2 7 0 0 13 0 0 9 0 4 0 0 0 6 3 10 11
Chromosome 16	0 13 6 1 0 11 14 2 0 0 10 0 9 12 3 8 0 4 0 7 0 0 0 5
Chromosome 17	14 0 6 4 0 0 0 13 0 0 7 0 12 0 10 3 11 0 1 5 9 0 2 8
Chromosome 18	12 9 6 5 0 0 1 4 0 0 14 0 10 8 13 2 3 11 7 0 0 0 0 0
Chromosome 19	0 0 10 12 0 1 0 0 2 14 4 13 7 6 0 0 5 8 0 9 0 0 3 11
Chromosome 20	0 7 0 0 12 0 4 8 5 0 9 10 14 0 1 0 6 2 11 0 13 0 3 0
Chromosome 21	0 10 6 12 5 0 0 14 2 8 4 11 0 1 0 0 7 0 0 9 0 0 3 13
Chromosome 22	14 3 0 10 0 9 5 12 0 1 11 2 8 6 4 0 0 0 7 0 13 0 0 0
Chromosome 23	0 4 7 0 9 10 0 11 0 6 12 3 13 0 2 0 8 0 0 14 0 0 1 5
Chromosome 24	0 3 11 2 0 0 13 14 0 4 9 8 1 7 0 0 10 0 5 12 6 0 0 0
Chromosome 25	0 12 14 5 0 4 13 3 10 9 0 0 6 8 0 0 2 0 7 0 0 11 1 0
Chromosome 26	0 9 0 5 0 0 0 0 13 0 1 10 11 2 12 4 8 14 7 0 6 3 0 0
Chromosome 27	0 8 1 5 0 0 14 0 3 6 12 0 0 11 4 0 13 2 7 0 0 9 10 0
Chromosome 28	5 7 1 0 11 0 2 0 3 0 4 0 0 14 0 6 0 8 9 0 13 10 0 12
Chromosome 29	10 11 1 0 5 0 8 3 0 0 0 7 12 0 13 0 9 4 2 6 0 14 0 0
Chromosome 30	5 12 1 0 14 0 0 0 0 0 3 9 8 10 13 2 4 6 7 0 0 11 0 0
Chromosome 31	0 12 1 0 0 11 0 0 5 0 6 13 2 4 0 3 7 0 9 10 0 14 8 0
Chromosome 32	4 6 0 9 0 14 13 2 0 0 12 10 8 0 3 0 0 11 0 0 7 0 1 5
Chromosome 33	0 6 0 2 0 4 12 5 13 0 0 10 7 3 0 0 14 0 1 0 8 0 9 11
Chromosome 34	10 6 2 11 13 1 14 0 12 8 5 0 9 0 0 0 4 0 0 0 0 7 0 3
Chromosome 35	0 0 0 0 11 0 10 7 9 0 0 0 14 0 13 2 4 8 0 3 12 5 6 1
Chromosome 36	0 9 2 0 0 11 1 5 12 7 0 0 3 0 0 13 4 6 0 0 10 14 8 0
Chromosome 37	0 0 0 0 3 10 12 0 6 1 2 0 0 13 7 4 0 0 14 11 0 5 8 9
Chromosome 38	0 0 11 0 13 14 0 6 1 12 5 3 0 0 0 2 10 4 0 9 8 0 0 7
Chromosome 39	0 0 9 0 1 0 14 0 0 8 2 4 0 13 12 6 10 7 0 11 5 0 0 3
Chromosome 40	0 0 1 14 0 0 12 6 4 7 0 5 11 9 0 2 3 0 13 10 0 0 0 8
Chromosome 41	11 7 0 0 1 0 8 6 3 2 0 9 5 0 0 13 4 14 12 10 0 0 0 0
Chromosome 42	0 7 2 0 13 0 3 6 4 1 0 8 12 11 0 14 0 5 9 10 0 0 0 0
Chromosome 43	11 0 0 14 12 7 0 9 0 5 6 13 10 0 3 0 0 4 0 0 2 0 8 1
Chromosome 44	3 2 0 10 7 1 4 0 0 12 5 0 0 0 0 0 13 8 11 14 0 9 6 0
Chromosome 45	14 5 0 4 1 0 10 0 0 2 3 7 12 0 8 0 9 0 11 0 0 13 6 0

Chromosome 46	0 1 3 0 7 4 9 6 11 0 12 0 0 0 14 0 8 5 0 2 0 0 10 13
Chromosome 47	0 0 0 9 11 14 7 0 0 5 13 6 0 0 10 12 0 3 0 8 0 2 4 1
Chromosome 48	9 8 5 0 6 13 1 3 0 10 0 0 0 11 0 7 12 2 0 4 14 0 0 0
Chromosome 49	0 2 13 0 12 10 11 3 14 9 5 0 0 0 4 7 0 0 0 1 8 6 0 0
Chromosome 50	0 4 13 0 7 14 3 5 0 8 10 2 9 0 6 11 0 0 0 0 0 0 1 12

Table 5.7: Second Generation of Chromosomes

Appendix D

Chromosome No.	Chromosome
Chromosome 1	3 0 10 8 1 9 5 4 11 0 0 12 13 0 0 7 0 0 0 0 6 2 14 0
Chromosome 2	0 0 11 0 13 14 0 6 1 12 5 3 0 0 0 2 10 4 0 9 8 0 0 7
Chromosome 3	9 0 10 12 3 13 14 7 8 1 0 11 0 0 0 0 6 5 0 0 4 2 0 0
Chromosome 4	14 3 0 10 0 9 5 12 0 1 11 2 8 6 4 0 0 0 7 0 13 0 0 0
Chromosome 5	0 0 9 0 1 0 14 0 0 8 2 4 0 13 12 6 10 7 0 11 5 0 0 3
Chromosome 6	7 0 5 0 0 1 0 9 0 4 3 11 10 0 13 12 0 2 14 0 6 0 0 8
Chromosome 7	0 6 0 2 0 4 12 5 13 0 0 10 7 3 0 0 14 0 1 0 8 0 9 11
Chromosome 8	0 4 7 0 9 10 0 11 0 6 12 3 13 0 2 0 8 0 0 14 0 0 1 5
Chromosome 9	0 6 0 9 0 13 3 7 14 0 5 0 11 12 0 0 10 0 2 8 1 0 0 4
Chromosome 10	6 5 0 0 0 0 10 2 14 12 0 11 0 3 8 9 0 0 13 0 4 0 1 7
Chromosome 11	0 6 0 11 0 1 3 0 13 0 10 0 9 5 14 4 7 12 2 0 0 0 8 0
Chromosome 12	11 0 0 0 0 0 14 3 6 2 1 5 7 4 12 0 0 13 8 0 10 0 9 0
Chromosome 13	10 6 2 11 13 1 14 0 12 8 5 0 9 0 0 0 4 0 0 0 0 7 0 3
Chromosome 14	0 9 0 5 0 0 0 0 13 0 1 10 11 2 12 4 8 14 7 0 6 3 0 0
Chromosome 15	7 1 9 4 6 13 0 0 0 2 10 8 0 12 14 0 5 11 0 0 0 0 0 3
Chromosome 16	0 12 14 5 0 4 13 3 10 9 0 0 6 8 0 0 2 0 7 0 0 11 1 0
Chromosome 17	0 0 10 12 0 1 0 0 2 14 4 13 7 6 0 0 5 8 0 9 0 0 3 11
Chromosome 18	0 0 3 0 0 12 10 4 2 6 8 13 0 11 0 0 1 9 7 0 0 14 0 5
Chromosome 19	4 6 0 9 0 14 13 2 0 0 12 10 8 0 3 0 0 11 0 0 7 0 1 5
Chromosome 20	7 0 0 0 5 3 0 8 0 6 13 0 10 12 0 11 14 4 2 9 0 0 0 1
Chromosome 21	8 0 0 13 0 7 0 0 0 0 9 14 11 6 10 12 0 3 2 4 1 0 5 0
Chromosome 22	0 1 12 3 10 0 6 8 7 0 11 0 13 0 9 0 2 14 0 4 0 0 0 5
Chromosome 23	0 13 0 8 5 11 14 3 0 7 0 0 0 0 9 4 2 0 0 0 6 1 12 10
Chromosome 24	0 0 13 5 10 7 6 1 0 0 14 0 3 0 9 0 2 11 0 0 0 8 4 12
Chromosome 25	14 0 6 4 0 0 0 13 0 0 7 0 12 0 10 3 11 0 1 5 9 0 2 8
Chromosome 26	0 3 11 2 0 0 13 14 0 4 9 8 1 7 0 0 10 0 5 12 6 0 0 0
Chromosome 27	0 5 8 7 0 0 12 10 0 13 4 14 0 0 2 0 0 0 3 11 1 0 9 6
Chromosome 28	0 0 1 14 0 0 12 6 4 7 0 5 11 9 0 2 3 0 13 10 0 0 0 8
Chromosome 29	5 12 1 0 14 0 0 0 0 0 3 9 8 10 13 2 4 6 7 0 0 11 0 0
Chromosome 30	6 0 1 12 11 0 0 3 0 5 0 10 4 7 14 2 8 9 0 0 0 0 0 13
Chromosome 31	0 7 0 0 12 0 4 8 5 0 9 10 14 0 1 0 6 2 11 0 13 0 3 0
Chromosome 32	0 8 0 2 9 0 10 1 5 0 0 0 13 0 0 7 0 4 6 14 12 3 0 11
Chromosome 33	0 8 0 12 7 0 0 0 5 0 4 13 0 0 1 9 0 6 14 10 11 0 3 2
Chromosome 34	0 9 2 0 0 11 1 5 12 7 0 0 3 0 0 13 4 6 0 0 10 14 8 0
Chromosome 35	0 0 0 0 11 0 10 7 9 0 0 0 14 0 13 2 4 8 0 3 12 5 6 1
Chromosome 36	0 7 12 0 0 1 8 3 11 6 0 0 2 0 10 0 4 9 0 0 5 14 13 0
Chromosome 37	0 4 0 11 10 0 0 14 12 0 0 2 1 7 9 6 0 8 13 0 3 0 0 5
Chromosome 38	0 2 0 0 10 12 0 0 0 4 3 14 0 13 8 7 11 0 9 1 6 0 0 5
Chromosome 39	0 7 0 6 10 4 0 2 12 14 1 0 3 8 0 13 0 0 11 9 0 0 0 5

Chromosome 40	0 0 11 6 13 5 0 0 0 0 8 7 12 2 0 10 0 3 4 1 9 0 14 0
Chromosome 41	10 11 1 0 5 0 8 3 0 0 0 7 12 0 13 0 9 4 2 6 0 14 0 0
Chromosome 42	0 0 11 9 1 4 5 14 0 0 0 7 12 0 3 0 2 0 6 10 13 8 0 0
Chromosome 43	0 13 6 1 0 11 14 2 0 0 10 0 9 12 3 8 0 4 0 7 0 0 0 5
Chromosome 44	77 14 0 0 12 0 0 3 2 1 13 0 8 0 6 10 0 0 11 5 0 0 4 9
Chromosome 45	0 5 7 0 10 9 14 13 8 0 11 0 4 2 6 0 0 12 0 3 0 0 0 1
Chromosome 46	4 3 13 0 0 6 9 11 10 0 14 0 12 1 0 2 7 5 8 0 0 0 0 0
Chromosome 47	0 0 7 0 14 3 4 12 1 10 2 13 5 8 0 11 0 0 0 0 0 0 9 6
Chromosome 48	0 1 0 8 0 11 13 0 7 0 3 6 4 14 12 0 2 0 10 0 0 5 9 0
Chromosome 49	0 12 13 14 5 10 1 11 0 4 6 8 2 0 3 0 0 0 0 0 0 0 7 9
Chromosome 50	0 0 4 7 0 0 0 2 0 6 10 3 11 14 0 1 0 13 8 0 12 0 5 9

Table 5.8: Third Generation of Chromosomes

Pictures of the Study Area

Figure 5.5: The intersection is about 60m wide. This makes its crossing risky

Figure 5.6: Pedestrians waiting on an island in the middle of the road waiting for a chance to traverse the intersection

Figure 5.7: The road divides a once one-community into two halves

Figure 5.8: Police personnel had to always be around to manage the confusion