DEVELOPMENT OF A TRAFFIC LIGHT CONTROLLER MODEL USING ARTIFICIAL BEE COLONY BASED ADAPTIVE DYNAMIC SCHEDULING ALGORITHM.

BY

Risikat Folashade ONUNDI, B.Eng (UNIMAID) 2011

(M.Sc/ENG/25654/2012-2013)

A DISSERTATION SUBMITTED TO THE SCHOOL OF POST GRADUATE STUDIES, AHMADU BELLO UNIVERSITY, ZARIA

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF A MASTER OF SCIENCE (M.Sc) DEGREE IN TELECOMMUNICATION ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

FACULTY OF ENGINEERING

AHMADU BELLO UNIVERSITY, ZARIA

NIGERIA

February, 2017

DECLARATION

I, ONUNDI Risikat Folashade, hereby declare that the work in this dissertation entitled "Development of Traffic Light Controller Model Using Artificial Bee Colony Based Adaptive Dynamic Scheduling Algorithm" has been carried out by me in the Department of Electrical Engineering. The information derived from literature has been duly acknowledged in the text and a list of references provided. No part of this dissertation was previously presented for another degree or diploma at this or any other institution.

Onundi Risikat Folashade _____ _____

Signature Date

CERTIFICATION

This dissertation entitled "DEVELOPMENT OF TRAFFIC LIGHT CONTROLLER MODEL USING ARTIFICIAL BEE COLONY BASED ADAPTIVE DYNAMIC SCHEDULING ALGORITHM" by Risikat Folashade ONUNDI meets the regulations governing the award of degree of Master of Science (M.Sc) in Telecommunications Engineering of the Ahmadu Bello University, and is approved for its contribution to knowledge and literary presentation.

Dr. K.A. AbuBilal	_____	_____
(Chairman, Supervisory Committee)	(Signature)	Date
Prof. M. B. Mu'azu	_____	_____
(Member, Supervisory Committee)	(Signature)	Date
Dr. Y. Jibril	_____	_____
(Head of Department)	(Signature)	Date
Prof. S. Z. Abubakar	_____	_____
(Dean. School of Postgraduate Studies)	(Signature)	Date

DEDICATION

This research work is dedicated to my Parents; Prof. L. O. Onundi and Mrs Rashidat O. Onundi.

ACKNOWLEDGEMENT

Despite the numerous efforts put into this project, it would not have been possible without the help of God, the author of knowledge and wisdom. I am highly indebted to You.

I remain totally indebted to my parents Prof. Lateef Olorunfemi Onundi and Mrs Rashidat Omolola Onundi, your endless love and support, kind and understanding personality will always be appreciated. Thank you very much. To my siblings Suwaiba, Sadiq, Munirah, Aminah, I owe you all a big thank you.

I wish to express my deepest gratitude to my supervisors Dr. K. A. AbuBilal and Prof. M. B. Mu'azu, for their tireless efforts, valuable guidance, and constant supervision towards the success of this work. The completion of this work could not have been possible without their constant participation and assistance. Thank you very much, sir.

I acknowledge with thanks all the lecturers of Electrical and Computer Engineering, Ahmadu Bello University, namely: Dr. S. M. Sani, Prof. B. G. Bajoga, Prof. B. Jimoh, Dr. Y. Jibril, Dr. A. M. S Tekanyi, Dr. A. D. Usman, Dr. S. Garba, Dr. T. H. Sikiru, Engr. Zanna, Mallam Tukur and other administrative staff.

I wish to express my sincere gratitude to Etisalat Nigeria for their well taught out Corporate and Social Responsibility (CSR) for a master class programme in Telecommunications Engineering at Ahmadu Bello University, Zaria. The internship experience was a real eye – opener and rare opportunity to the field of telecommunications. It enabled me to have hands – on - experience to telecommunications engineering equipment. Similarly, I am also grateful to Etisalat Academy, Dubai for the weeklong training on Fibre Optics and subsequent certification in the area. The training also afforded me my first international experience to study in another environment. My hopes and prayers are that Etisalat Nigeria gain greater heights to become an undisputed leading telecommunications network not only in Nigeria but all across Africa.

I acknowledge the Vice President Regulatory and Corporate Affairs, Alhaji Ibrahim Dikko, Pioneer Project Manager, Emeritus Professor Munzali Jibril, Head of Network Quality and Architecture Assurance, Mr. Mohammed Elsisi, Mr. Habeeb Oyewunmi, Mr. Mohammed Suleh, Mrs. Oyetola Oduyemi and Mrs. Rose Makinwa.

My immeasurable gratitude goes to Mr. Zenon de Silva and Mr. Alex of Etisalat Academy, Dubai.

To my best friend, my colleague and course mate, "olowo ori mi" Adebiyi Hadir Busayo, I am highly grateful for your encouragement as always especially, during times I wished to give up. May Allah continue to answer your secret prayers (Amin).

To my prince, Muhammad El-Amin Temide Adeoye Adebiyi thank you for coming into my life.

I am also thankful to my friends and course mates, Abdulrashid M. Kabir, Stephen Joseph Soja (school father), Salawudeen Tijjani, Elvis Obi, Bashir Sadiq, Olaniyan Abdulrahman, Bukola Ishola, Atuman Joel, Ore Ofe Ajayi, Valentine Ikpo, Tijesunimi Alonge, Habeeb Bello, Maryam Lami, Arafat, Abdulmuminu, Abdulhakeem Lawal, Aisha Bello, Michael Lucky, Daniel Alatise, Janet Gadzama, Victor Faniwa, Gbenga Ogbonnaiye, Abubakar Ahmed (E/E Unimaid), John Askarju and Abubakar Umar. Their continuous support and contributions toward the success of this work would never be forgotten. May God reward them and strengthen our friendship.

I appreciate the efforts and contributions of Adebiyi's family, A&S family, Minaru's Osogbo family, Layi Oni's family, Prof. B.A. Adegboye, Prof. Osinubi, Albert Nelson (Elementary school tutor), Tajudeen Babatunde Awoye (High school tutor), Seun Onundi, Sunday Olorunfemi and all those who have contributed to my success. May God bless you all.

Risikat Folashade ONUNDI

February, 2017.

ABSTRACT

An Adaptive Dynamic Scheduling Algorithm (ADSA) based on Artificial Bee Colony (ABC) was developed for vehicular traffic control. The developed model optimally schedule green light timing in accordance with traffic condition in order to minimize the Average Waiting Time (AWT) at the cross intersection. A MATLAB based Graphic User Interface (GUI) traffic control simulator was developed. Three scenarios of vehicular traffic control were simulated and the results and the presented results shows that scenario one and two demonstrated the variation of the AWT and Performance of the developed algorithm with changes in the maximum allowable green light timing over the simulation interval. In the third scenario, an AWT of 38sec was recorded against a maximum allowable green light duration of 120sec, during which 1382 vehicles were evacuated from the intersection, leaving 22 vehicles behind. The algorithm also had a performance of 98.43% over a simulation duration of 1800sec. In order to demonstrate the effectiveness of the developed ADSA this research was validated with the literature. The result obtained for the AWT of the developed ADSA had a performance of 76.67%. While, for vehicular queues cleared at the intersection the developed ADSA had a performance of 53.33%. The results clearly expressed that the developed ADSA method has been successful in minimizing the Average Waiting Time and vehicular queues at intersection.

Table of Contents

CHAPTER TWO: LITERATURE REVIEW

CHAPTER THREE: MATERIALS AND METHODS

CHAPTER FOUR: RESULTS AND DISCUSSION

CHAPTER FIVE: CONCLUSION AND RECOMMENDATION

LIST OF APPENDICES

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ABC Artificial Bee Colony Algorithm

ADSA Adaptive Dynamic Scheduling Algorithm

ADR Average Departure Rate

APTS Advanced Public Transport System

ARA Average Arrival Rate

ATIS Advanced Traffic Information System

AWT Average Waiting Time

BS Base Station

CS Control Signal

ETPS Enabled Transport Pricing System

ETEPP Etisalat Telecommunications in Engineering Postgraduate Programme

FFD Full Function Device

GUI Graphic User Interface

ITS Intelligent Transportation System

IDA Intersection Decongestion Algorithm

OGTA Optimal Green light Timing Algorithm

MD Motion Decision

MATLAB Matrix Laboratory

RFD Reduced Function Device

RSA Road Selection Algorithm

VTCS Vehicular Traffic Control System

WSN Wireless Sensor Network

CHAPTER ONE

GENERAL INTRODUCTION

1.1 Background

Intelligent Transportation System (ITS) is a system in which information and communication technologies are applied in road transport, including infrastructure, vehicles and users, and in traffic management and mobility management, as well as for interfaces with other modes of transportation (Kotwal *et al.*, 2013). The Intelligent Transportation System (ITS) make use of technologies in electronics, communications, computers, control, sensing and detecting in all kinds of transportation system. The primary goals of ITS systems are to "increase transportation system efficiency and capacity such as: enhance mobility, improve safety, reduce energy and environmental costs" (Kotwal *et al.*, 2013).

1.2 Intelligent Transportation System (ITS)

Generally, Intelligent Transportation Systems (ITS) are classified into five systems according to their functions as follows (Samadi *et al.*, 2012):

1.2.1 Advanced traffic light management System (ATMS)

ATMS is usually employed to detect traffic situations, transmits them to a control center via communication network, and then develops traffic control strategies by combining all kinds of traffic information. The system makes use of base stations to carry out traffic control and transmits the information to drivers and concerned departments (Samadi *et al.*, 2012). ATMS includes ITS applications that focus on traffic control devices such as ramp metering, adaptive traffic signal control, traffic operation centers (TOC) and high occupancy vehicle control and so on (Ezell, 2010).

1.2.2 Advanced traveler information system, (ATIS)

This system allows road users to access real time information such as in the car, at home, in the office or outdoor by choosing transportation modes, travel trips and routes. The system mainly includes: changeable message signs, Highway Advisory Radio, Global Positioning System Road side weather information systems and so on (Ezell, 2010).

1.2.3 ITS –enabled transportation pricing systems, (ITS-ETPS)

Employing ITS-ETPS technologies to road network management helps drivers to automatically pay tolls electronically via a Dedicated Short Range Communications (DSRC-enabled onboard device) or tag placed on the windshield. Another ITS-enabled approach is Vehicles Miles Travelled (VMT) that charges motorists based on miles driven (Ezell, 2010). With this application ITS finds a central role in financing most countries transportation systems (Samadi *et al.,* 2012).

1.2.4 Advanced public transportation system, (APTS)

APTS applies the technology of ATMS, ATIS and ITS-ETPS in public transportation in order to improve the quality of service, increase efficiency and the number of people who take public transportation (Samadi *et al.,* 2012).The system mainly includes the automatic vehicle location (AVL) monitoring, computer scheduling and e–tickets (Ezell, 2010).

1.2.5 Vehicle to infrastructure integration (VII) and Vehicle to vehicle (V2V) integration.

Vehicle-to-vehicle applies the technology of Intelligent Speed Adaptation (ISA) which is aimed to assist drivers in keeping with speed limits. By correlating information about the vehicles position with a digital speed limit map thus enabling it to recognize if it has passed posted speed limit (Ezell, 2010). Also, Cooperative Intersection Collision Avoidance systems (CICAS) play a role in V2V integration systems (Samadi *et al.,* 2012).

Maintaining an efficient transportation system has been the main concern for public authorities in a city confronted with multiple traffic jams each day, which cause large parts of it to become irresponsive to traffic movement(s). Various studies have revealed that the major problem lies in the number of intersections and their coordination in terms of controlling (releasing or halting) the traffic which leads to traffic congestion (Cosariu *et al.*, 2013). Many intelligent control systems have been developed and most of the researchers test these systems on simple networks with theoretical (and often static) traffic volumes (Sinhmar, 2012) and (Mohan & Rani,2013). This research work is set out to develop a traffic light controller model using an artificial bee colony based adaptive dynamic scheduling algorithm to minimize the average waiting time and conflict.

1.3 Motivation

The problems associated with fixed- time controllers which made use of pre-determined traffic control that does not respond to dynamic changes to traffic condition on each lane was the reason a new optimization algorithm known as the Artificial Bee Colony was introduced.

1.4 Significance of the research

To develop an adaptive traffic controller model using Artificial Bee Colony that is aimed at minimizing average waiting time and conflicts at vehicular intersections with emphasis on simultaneous moves.

1.5 Statement of the problem

The order and durations of the green lights for intersection traffic management are pre-determined and do not adapt dynamically to the traffic conditions. Detectors are sometimes used to count vehicles on each lane of an intersection but the data they report is generally used only to select between a few static sequences and timings setups. When unmanaged, high vehicle volume within urban traffic networks results in congested and slow moving traffic. This has led to many negative

economic and environmental consequences (McKenney, 2011). It is therefore important to efficiently control road networks vehicles. Many researchers have made an effort to improve the efficiency of traffic signals using many intelligent systems evaluated using unrealistic traffic models which often include single intersection and static traffic volumes. The developed model for this work introduced a more realistic control strategy through the development of an adaptive dynamic scheduling algorithm based on an artificial bee colony algorithm that is capable of controlling the vehicular traffic at road intersections.

1.6 Aim and Objectives

The aim of the research is development of a traffic light controller model using an artificial bee colony (ABC) based adaptive dynamic scheduling algorithm. The objectives are:

1 To develop a model for vehicular traffic control system and implement an artificial bee colony based adaptive dynamic scheduling algorithm for road intersection using MATLAB R2012a Software.

2 To develop a Graphic User Interface (GUI) based vehicular traffic management simulator using MATLAB R2012a software.

3 To simulate various scenarios of traffic management using the developed simulator in (2) and compare the results of average waiting time with those obtained in the work

(Erwan *et.al*, 2015).

1.7 Methodology

The following sequence of steps is adopted in order to achieve the aim and objectives of this research.

1. Development of mathematical model for a cross-junction, fitted with traffic signals and sensors to monitor traffic movement.
2. Development of a motion control strategy which comprised of twenty two possible phases and four simultaneous moves of the traffic at the intersection.
3. Implementation of a motion control strategy on the developed intersection in (1).
4. Development of an adaptive dynamic scheduling algorithm using artificial bee colony that optimally allocate time to the various traffic signals controlling traffic movement.
5. Development of a MATLAB GUI simulator to aid vehicular traffic control simulation based on the developed algorithm in (4).
6. Simulating various scenarios of traffic control to obtain results for comparison with those of the work of (Erwan *et al.,* 2015) for validation purpose.

1.8 Outline

In Chapter one, general background on Intelligent Transportation System (ITS) was presented. In Chapter two, a concise review of the fundamental concepts and literature regarding Adaptive Traffic Control and various optimization algorithms are presented. In Chapter three, the developed model for Vehicular Traffic Control System was developed using Artificial Bee Colony Algorithm are presented respectively; MATLAB GUI are also developed and presented in the chapter. In Chapter four, Vehicular Traffic Control System was simulated using the developed simulator (presented in Chapter three); the results are presented are briefly discussed; and the developed simulator were further compared with the results obtained by (Erwan *et al.,* 2015) as a means of

validation. Chapter five presents the conclusion, recommendations, and limitations of the entire research work. The list of cited references and MATLAB codes in the appendices are provided at the end of this dissertation.

CHAPTER TWO

LITERATURE REVIEW

2.1 Introduction

This section reviews pertinent fundamental concepts and similar research works. It is divided into two parts: (i) the review of fundamental concepts relevant to this research work and, (ii) review of similar research works.

2.2 Review of Fundamental Concepts

This sub-section presents the review of the fundamental concepts pertinent to this research work**.** These concepts which include: Terminologies used in intelligent transportation systems, Sensor networks, overview of a road intersection, computational approaches to traffic optimization, approaches in traffic congestion, traffic stream significance, generalized model base formulation for single and multiple intersections and approaches used in programming traffic.

2.2.1 Terminologies used in intelligent transportation systems

Some of the terminologies used in intelligent transportation systems as described by Yousef *et al.*, (2010) are as follows:

(i) **Traffic phase:** defined as the group of directions that allow waiting vehicles to pass the intersection at the same time without any conflict.

(ii) **Conflict:** This occurs whenever any traffic element in a certain arm must not cross the path of another stream in order to clear an intersection.

(iii) **Traffic cycle:** is one complete series of a traffic phase plan executed in a round robin fashion.

(iv) **Traffic cycle duration (T):** is the time of one traffic cycle needed for the green and red time duration for each traffic signal.

(v) **Average waiting time (AWT**): defined as delay in queues but excludes service time measured in units of time.

(vi) **Traffic queue:** the average queue size of vehicles which equals the arrival rate (vehicles per unit time) multiplied by the average waiting time (both delay in queues plus service time) (in units of time).

(vii) **Peak hour** – Time of a day the traffic congestion on the road is at its highest and is normally found twice a day.

(viii) **Non-peak hour** – The day apart from the Peak hours and the weekends where the traffic congestion is not very high.

(ix) **Arrival rate** – It can be defined as the average number of customers arriving to a particular system.

(x) **Service time** – The average time taken by a server to service the customers waiting at a system.

(xi) **Green light time:** once the phase is selected, the minimum time required to let all vehicles.

2.2.2 Wireless Sensor Network

A sensor is defined as a detection unit with generally limited capacities in terms of energy, memory, computation power and communication (Karpis, 2013). This type of network is able to make decisions without external assistance. Sensors when networked to a base station, could be used to control one or more intersection(s) of a city. Sensing units used to detect vehicles are magnetometers, as they are accurate, small and relatively cheap (Karpis, 2013). Figure 2.1 presents a typical wireless sensor network setup (Arrobo, 2012).

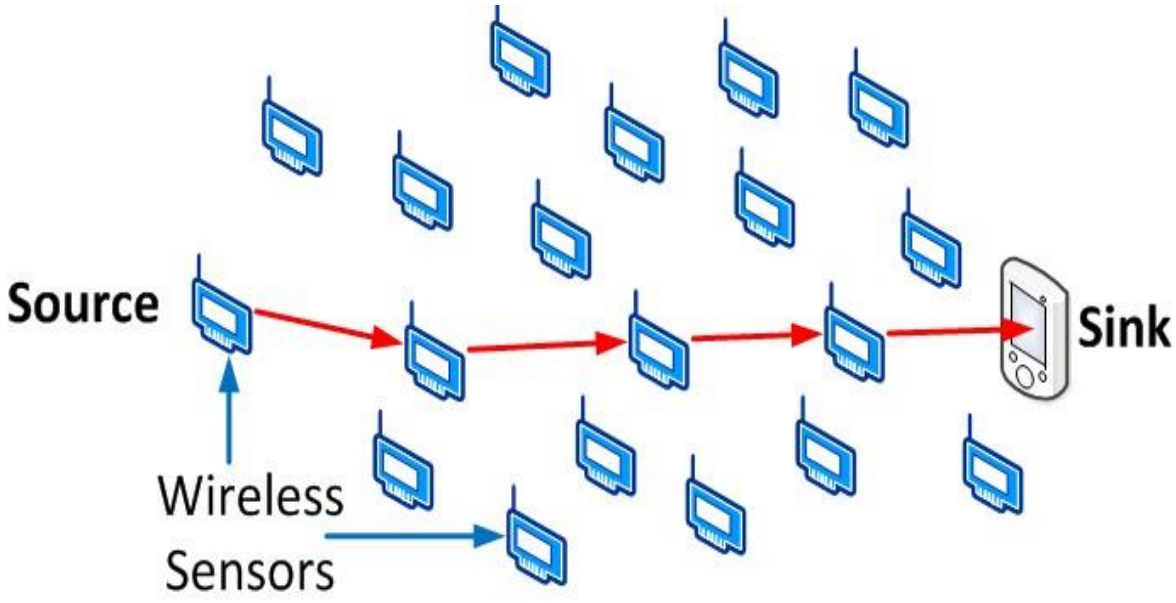


Figure 2.1: Typical Wireless Sensor Network (Arrobo, 2012).

2.2.3 Sensing Technologies

Advancement in telecommunications and information technology have enhanced the technical capabilities that would facilitate motorist safety to benefit from intelligent transportation systems globally using sensors. There are various types of sensors namely (Kotwal *et al.*, 2013):

(a) Inductive Loop Detection

An inductive loop is a continuous run of wire that enters and exits from the same point. The two ends of the loop wire are connected to the loop extension cable, which in turn connects to the vehicle detector. The detector powers the loop causing a magnetic field in the loop area.

The loop resonates at a constant frequency that the detector monitors (www.marsh.com). When a large metal object, such as a vehicle, moves over the loop, the resonate frequency increases. This increase in frequency is sensed and, depending on the design of the detector, forces a normally open relay to close. The relay will remain closed until the vehicle leaves the loop and the frequency returns to the base level. The relay can trigger any number of devices such as an audio intercom

system, a gate, a traffic light, etc. Inductive loops could be placed in a roadbed to detect vehicles as they pass through the loop's magnetic field. Figure 2.2 shows an inductive loop system which consists of three components basically: a loop (preformed or saw-cut), loop extension cable and a detector (www.marsh.com). The simplest detectors count the number of vehicles during a unit of time as they pass over the loop (Kotwal *et al.*, 2013).

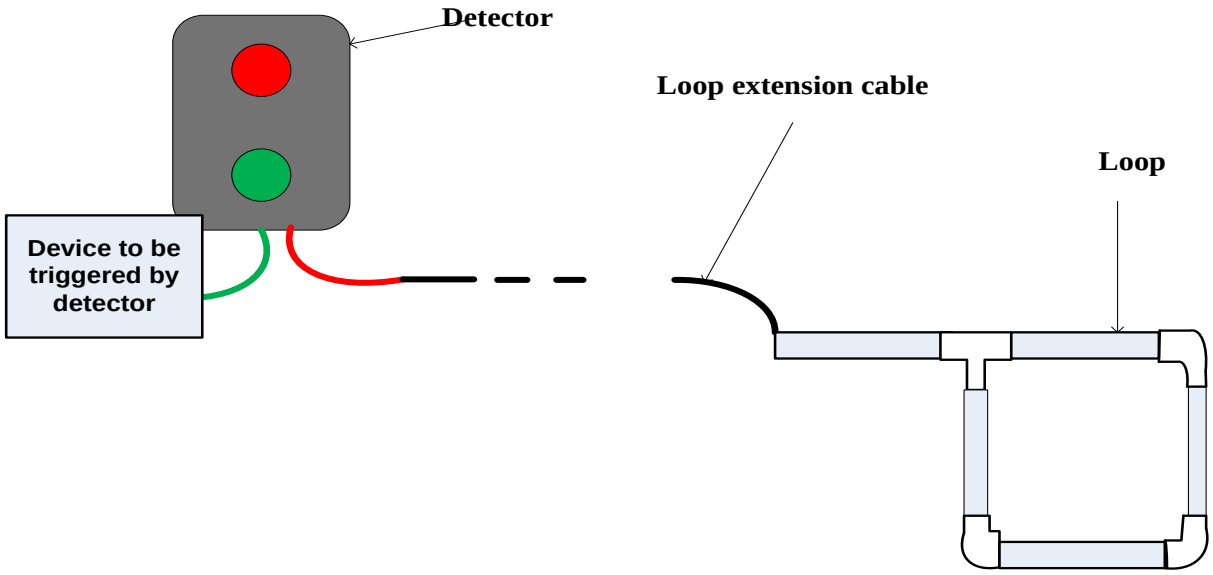


Figure 2.2: An Inductive Loop System (www.marsh.com)

(b) Video Vehicle Detection

Traffic flow measurement and automatic incident detection using a video camera is another form of vehicle detection it is known as non – intrusive detection. Video from cameras are fed into processors that analyse the changing characteristics of the video image as vehicles pass. There are different sizes of the vehicles that use the road, some of them automobile, trucks which could be observed and counted for traffic violations, such as lane crossing, wrong parking and even stranded vehicles that are blocking the roads (Odeh, 2013). Figure 2.3 illustrates an urban road network where camera is incorporated for vehicle detection system. Cameras are typically mounted on

poles or structures above to the roadway. The data obtained by these sensors are sent and converted to information in a central control room (base station) (Odeh, 2013).

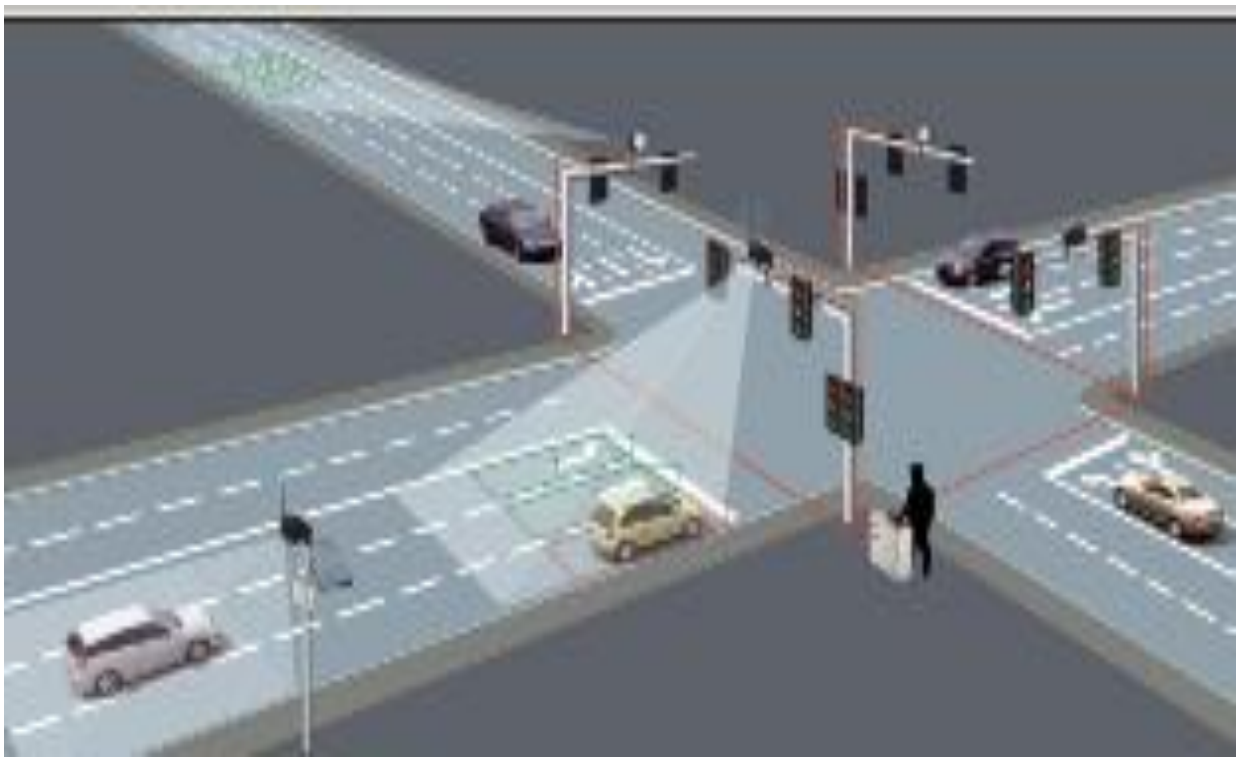


Figure 2.3: Video Camera Based Traffic Control Monitoring System (Odeh, 2013).

(c) Wireless Sensor Detection (WSD)

The WSD is implemented based on IEEE 802.15.4 standard protocol. Wireless sensors are responsible for collection of traffic data, such as, the number of cars queued in the lanes related to each traffic light phase. WSD is made up of two types of network devices namely; Reduced Function (RFD) and Full Function (FFD) devices. The system architecture of a WSD is described by Figure 2.4. The Reduced Function Devices (RFDs) are typically located along the roadside, within a road cavity, following a linear topology. Most RFDs detect the presence of queued cars by evaluating the distortion of the Earth's magnetic field produced by the presence of ferrous objects (Collotta *et al.*, 2015). The data related to each lane is then collected by the Full Function

Devices (FFDs), which poll the associated RFDs. The collected aggregate data is then forwarded to the First Pan Coordinator (FPC). The communication among the network devices is realized through the IEEE 802.15.4 protocol working in a beacon-enabled mode, using the Guaranteed Time Slots (GTSs) in order to avoid collisions (Collotta *et al.*, 2015). The FPC is embedded in the intersection manager, which is located in the center of the traffic lights intersection.

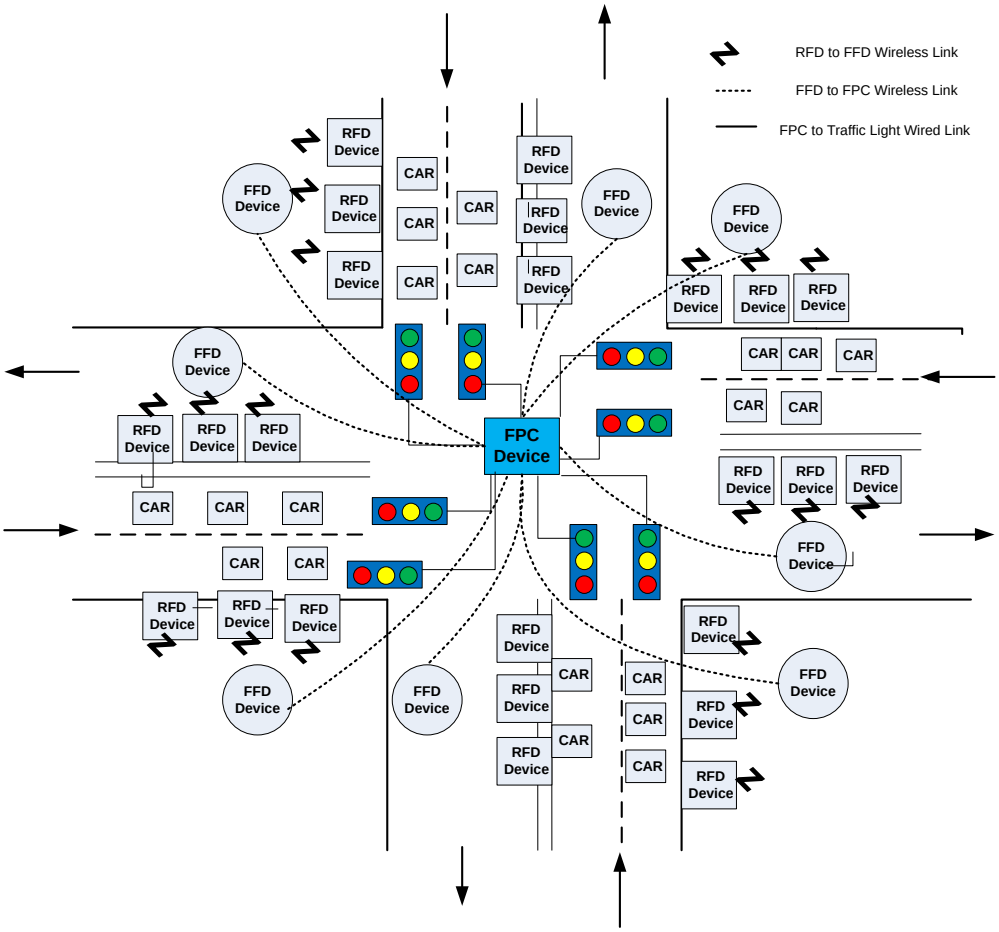


Figure 2.4: WSD Based Traffic System Architecture (Collotta *et al.*, 2015).

The motivation for adoption of wireless sensor in traffic monitoring is that there is minimal disruption during installation and maintenance. Wireless sensors allow for easier coverage extensions of Intelligent Transportation Systems (ITS) applications. The Wireless sensors allows for scalable and flexible deployments. These sensor nodes can be placed anywhere on the road way as far as they are within their communication range. Covering large areas due to Multi-hop communication, this characteristics is a big advantage over other traffic monitoring technologies (Collotta *et al.*, 2015).

2.2.4 Overview of a road intersection

Basically there are four paths marked North (N), South (S), East (E), West (W) leading to a road intersection and each path has three lane in the incoming direction, which turn left (L), go-forward (F) and turn right (R). Each direction is further decomposed into one left lane for vehicles turning left and one or more right lanes for vehicles going straight or right. As illustrated in Figure 2.5 (Faye *et al.*, 2012), a typical wireless sensor detection is generally deployed around a traffic controller, or base station. This is used to provide access and enhance connectivity to a control center in which operators are able to modify lights behavior and timing. Traditionally, it is considered that each incoming lane is equipped with two sensors: one located at the lights to count the departures and the other at a fixed distance before the light, to count arrivals. Generally, this detection distance is recommended to be set in terms of number of vehicles in a queue and is usually specified to be between 5 and 8 vehicles (Yousef *et al.*, 2010) or depending on the maximum authorized green time. An illustration of a single road intersection network using wireless sensor network is shown in Figure 2.5. There are eight (8) possible phases within the path (P) of {E, S, W and N} and a direction (D) of {L, F, R} (Faye *et al.*, 2012). Thus, a lane where a vehicle is running can be determined by a pair of {P, D}. As a result, there is at most twelve lanes

operating relative to the pair (P, D). From the Figure 2.6 (Yousef *et al.*, 2010), the traffic sensor nodes are located on each lane, which comprise of at least two traffic sensor nodes. One is placed before the traffic signal and the other is after the traffic signal to detect the arrival and departure. Therefore, each intersection would need at least twenty four traffic sensor node to get full information of the traffic flow.

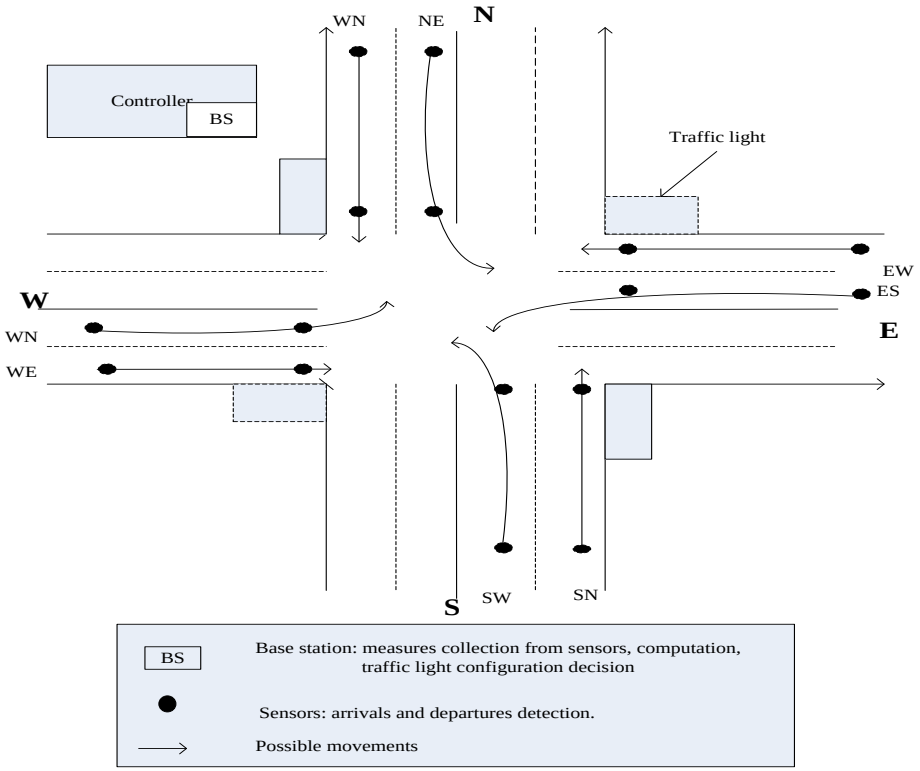


Figure 2.5: Single Road Intersection Network using Wireless Sensor Detection (Faye *et al.*, 2012).

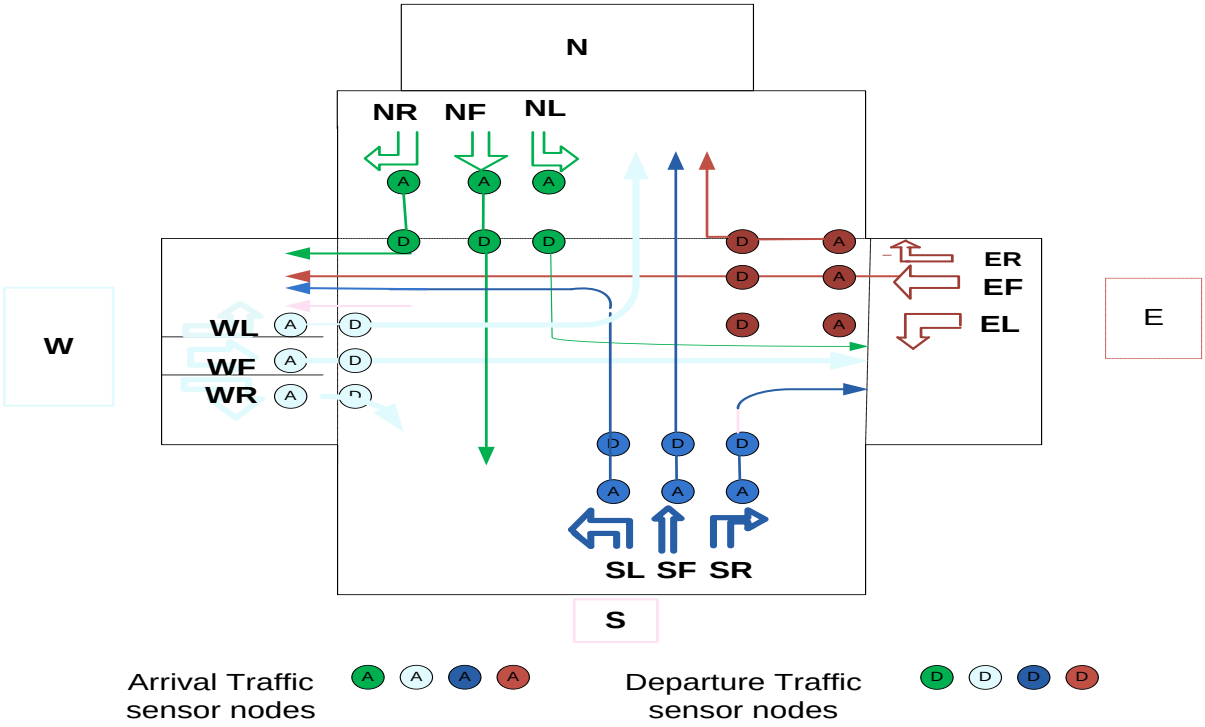


Figure 2.6: Intersection and Traffic Sensor Node (Yousef *et al.*, 2010).

2.2.5 Computational Approaches to Traffic Light Optimization

This section includes background information regarding a number of approaches which have been used in the traffic light optimization domain. These approaches include, stochastic approach namely Monte Carlo algorithm. Similarly, heuristic approach comprising of fuzzy logic, genetic algorithm and artificial bee colony (ABC).

(a) Monte Carlo simulation

Monte Carlo simulation is a stochastic computerized mathematical technique. The Monte Carlo method is also known as the statistical experimental method, which is based on the probability theory and mathematical statistics for simulation guidance (Wang *et al.*, 2014). Here, random numbers are generated to obtain random observation from probability distributions. It is usually a sequence of numbers whose probability of occurrence is the same as that of any other number in

the sequence. The advantage for this method is that it does not rely on any specific form of the simulation limit state equations (Shanmugasundaram & Banumathi 2015). However, the disadvantages are that, high precision requires a very large number of simulations and can lead to a large amount of calculation; as a result the calculation time becomes longer.

(b) Fuzzy Logic

The idea of fuzzy set logic theory was first proposed by Lofti A. Zadeh in 1965. Fuzzy logic representations are founded on fuzzy set theory that try to capture the way humans represent and reason with real- world knowledge. A fuzzy set can be defined mathematically by assigning to each possible individual in the universe of discourse, a value representing its grade of membership in the fuzzy set. This grade corresponds to the degree to which that individual is similar with the concept represented by the fuzzy set. In other words, fuzzy set represent flexible sense of membership of elements to a set (Kumar&Kumar, 2012).When using fuzzy logic, truth values of variables can take on a continuous value in the range of [0*;* 1]. Decisions in fuzzy logic are usually made using a rule base which can be developed by expert knowledge or trial-and-error (Kumar&Kumar, 2012). Figure 2.7 shows the Flowchart describing the five steps in the design of a fuzzy system (Stavroulaskis, 2004).

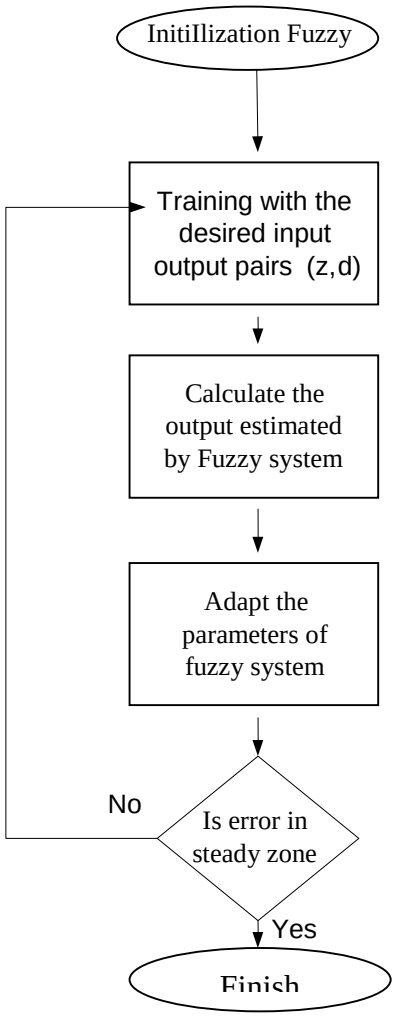


Figure 2.7: Flowchart for a Fuzzy System (Stavroulakis, 2004).

(c) Genetic Algorithm

In the field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. It was proposed by Holland in 1973 (Holland, 1973). Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover (Stavroulaskis, 2004). The Flowchart for genetic algorithm is shown in Figure 2.8. Each candidate solution has a set of properties (its chromosomes or genotype) which can be mutated and altered. Traditionally, solutions are represented in binary as strings of

0s and 1s. The evolution usually starts from a population of randomly generated individuals and it is an iterative process, with the population in each iteration called a generation. In each generation, the fitness of every individual in the population is evaluated; the fitness is usually the value of the function in the optimization problem being solved. The more fit individuals are stochastically selected from the current population, and each individual's genome is to form a new generation. The new generation of candidate solutions is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when either a maximum number of generations has been produced, or a satisfactory fitness level has been reached for the population (Stavroulaskis, 2004).

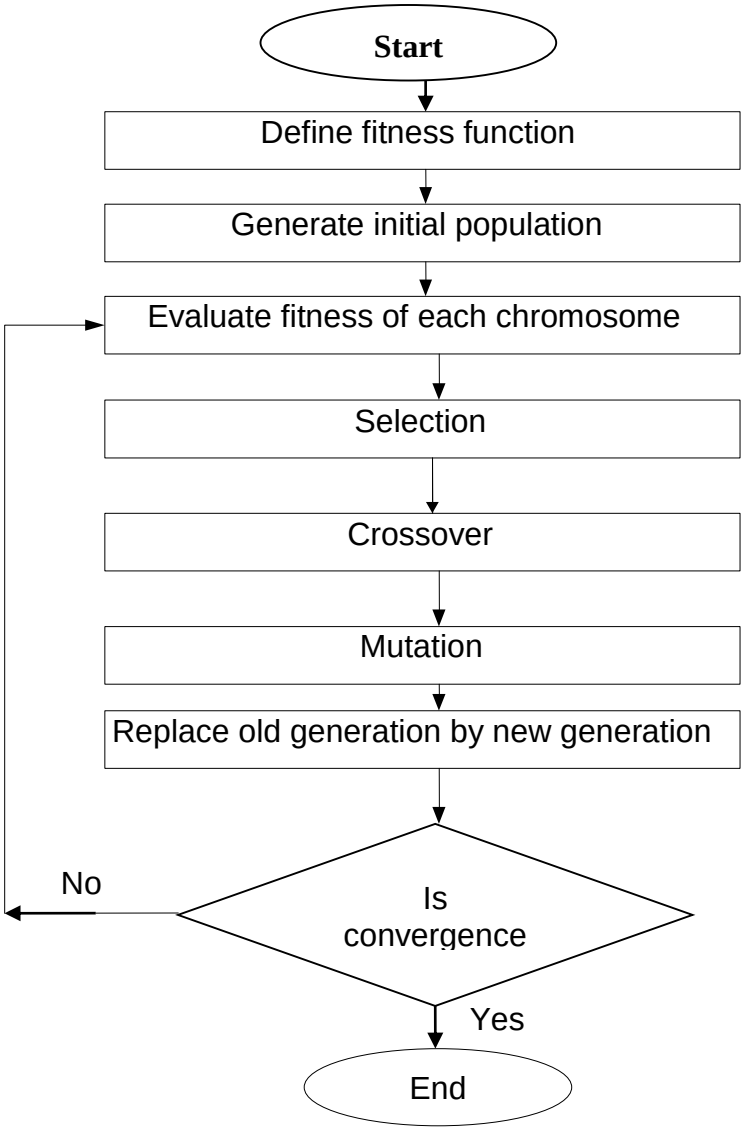


Figure 2.8: Flowchart of Genetic Algorithm (Kumar and Jyohstiree, 2012)

(a) Some of the advantages of genetic algorithm (www.intelligenttechniquesinmechatroics.com)

(i) Can solve every optimization problem which can be described with chromosome encoding.

(ii) Solves problem with multiple solutions.

(iii) Easy to understand, does not use knowledge of mathematics.

(b) Limitations of genetic algorithm (www.intelligenttechniquesinmechatroics.com)

(i) Certain optimization problems cannot be solved using genetic algorithm.

(ii) There is no constant optimization response time.

(iii) Genetic algorithms are limited in real-time control due to random solutions and convergence.

(d) Artificial Bee Colony

The Artificial Bee Colony (ABC) algorithm is a population-based meta-heuristic optimization which was inspired by the foraging behaviour of honeybee swarm. The ABC colony is divided into employed bees and onlooker bee. In ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution (Karaboga *et al.*, 2014) Employed bees are responsible for exploiting the food sources (nectar) and they pass the information to onlooker bees in the hive about quality (fitness) of the food source site which they are exploiting. The employed bees whose food source has been exhausted by the bees becomes a scout (Karaboga *et al.*, 2014). There are three main stages of ABC namely (Karaboga *et al.*, 2014):

(i) Employed Bee Stage

An employed bee searches for a food source and finds a new food source within the neighbourhood of the food source in a random manner using (Karaboga & Akay, 2009).

$$v_{ij} = x_{ij} + \Phi\left(x_{kj} - x_{ij}\right) \tag{2.1}$$

The fitness of the new solution is calculated using the expression (Karaboga & Basturk, 2007).

$$fitness(x_i) = \begin{cases} \dfrac{1}{1+f(x_i)} if & f(x_i) \geq 0 \\ 1+|f(x_i)| & otherwise \end{cases} \quad (2.2)$$

where:

x_i = old food source
v_i = new food source
Φ_{ij} = uniform random value $(-1,1)$
f_i = the function to be minimized

If v_i has better fitness than the old food position x_i, then x_i is replaced by v_i. Otherwise the previous position x_i is retained

(ii) Onlooker Bee Stage

An onlooker receives information from the employed about the quality of food source via waggle dance. The quality and melody of this waggle dance influence the choice of food source to be exploited (Karaboga & Akay, 2009). The probability P_i that the employed bee with food source x_i would be picked by an onlooker bee is computed using (Karaboga & Basturk, 2007).

$$P_i = \frac{fitness(x_i)}{\sum_{n=0}^{SN} fitness(x_n)} \quad (2.3)$$

Like the employed bees, each onlooker bee also employs equation (2.1) to produce trial food source v_i in the vicinity of its current food source position x_i. If v_i has better fitness then x_i is replaced by v_i Otherwise, v_i is discarded.

(iii) Scout Bee Stage

A scout bee is created only when a particular food source x_i cannot be improved through a pre-determined number of trials "limit" (Karaboga & Akay, 2009). The employed bee now becomes a scout bee and its food source is positioned randomly across the search space using equation (2.4), where j = 1, 2… D and $\lfloor \min_j, \max_j \rfloor$ is the search space along the j^{th} dimension (Karaboga & Basturk, 2007).

$$x_{i_j} = \min_j + rand(0,1)(\max_j - \min_j) \tag{2.4}$$

A generalized Flowchart on artificial bee colony is shown in Figure 2.9

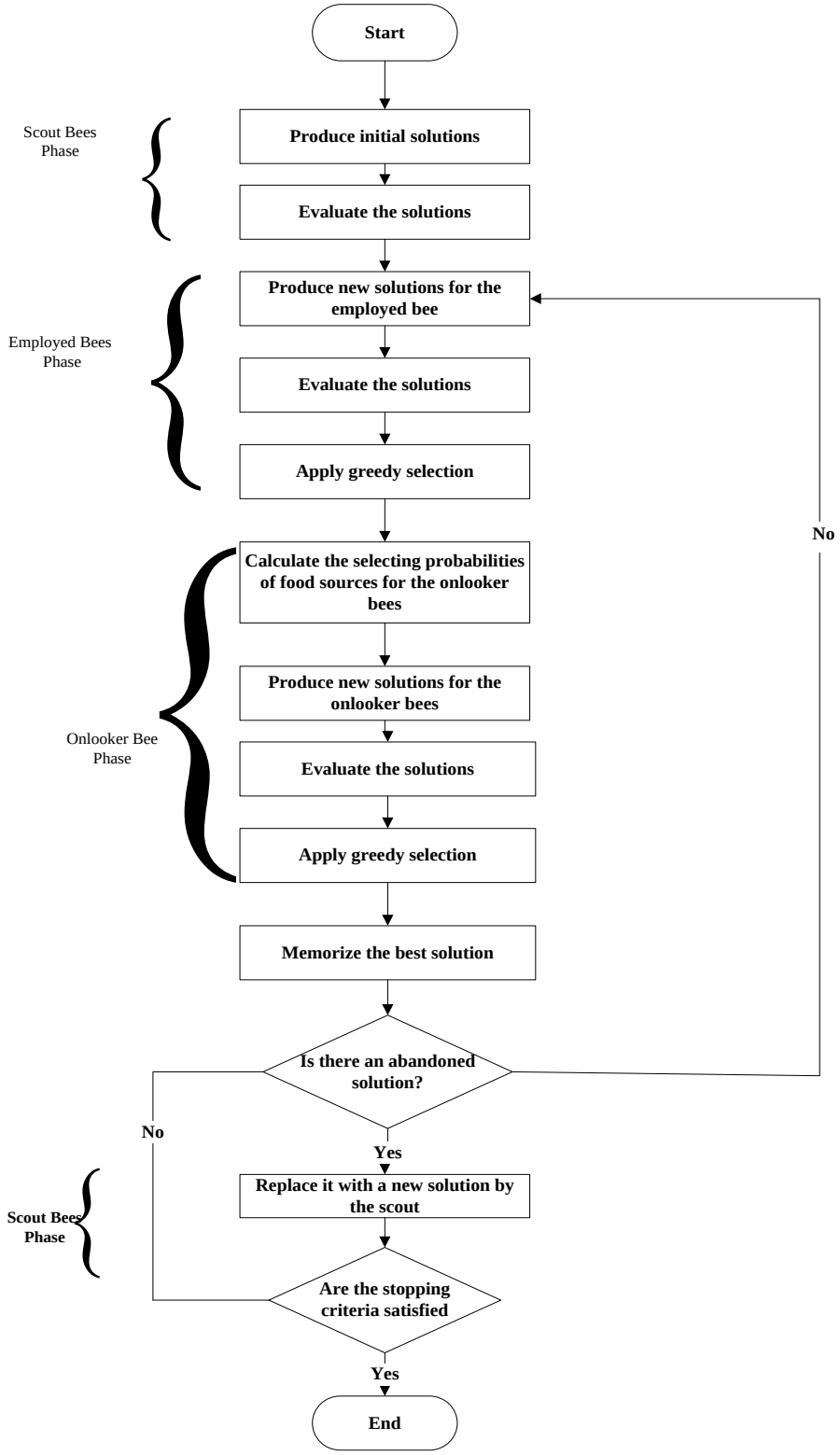


Figure 2.9: Flowchart of Artificial Bee Colony (Karaboga & Akay, 2009).

(iv) The three (3) main parameters of ABC are (Karaboga & Akay, 2009)

a) Population size (SN): this is the number of food sources (or solutions) in the population.

b) Maximum Cycle Number (MCN): this refers to the maximum number of generations.

c) Limit: is used to diversify the search, to determine the number of allowable generations for each non- improved food source to be abandoned.

(v) The steps of the ABC algorithm are given as follows (Karaboga & Akay, 2009)

Initialize the population of solutions xi , i=1,2,…,SN, and evaluate them

Step 1: Generate new solutions *vi* for the employed bees by equation (2.1) and determine the quality of the solutions

Step 2: Apply the greedy selection process for the generated new solutions in Step 1

Step 3: Calculate the probability values P_i for the solutions xi by equation (2.3)

Step 4: Generate the new solutions for the onlookers from the solutions xi due to the probabilities pi using roulette wheel selection

Step 5: Apply the greedy selection process for the onlookers

Step 6: Determine the solution for the scout bee and replace it with produced solution xi by equation (2.4)

Step 7: If the number of cycle is reached MCN, the algorithm is terminated. Else go to step 1.

(vi) Some of the advantages of using Artificial Bee Colony are as follows (Karaboga & Akay, 2009):

1. It is a simple and flexible algorithm that uses fewer number of control parameters in comparison to other prominent optimization algorithm.
2. The use of few parameters shows significant improvement in computational time to dissipate traffic as compared to other algorithms.
3. Artificial Bee Colony uses an iterative process which suits the stochastic nature in which traffic queues occur.

Using the stochastic approach which Monte Carlo simulation method is categorized, the error within simulation can be determined. However, Monte Carlo simulation still requires high precision using very large number of simulations that leads to high computational time.

Based on the heuristic approaches discussed so far, Genetic Algorithm offer an advantage of vast application but often generates solutions that are highly random in nature and does not guarantee a real and fast solution. Fuzzy Logic on the other hand is quite promising in its solution search but involves a number of rules that may not be suitable to modify and quite complex. Finally, Artificial Bee Colony (ABC) has a disadvantage of having fewer parameters which can result in slow solution search but can easily be modified by introducing model parameters that could aid better performance in terms of traffic management control and analysis.

2.2.6 Adaptive Dynamic Scheduling Algorithm

The Adaptive dynamic Scheduling algorithm run real-time traffic light regulation of phase sequence and green time duration based on data gathered by the WSN on the road (Aljaafreh & Oudat 2014). The phase sequence is determined by using queue length for each input variable or

flow or by sorting them in descending order of priority (Collata *et al.*, 2014). The phase sequence assigns priority to each phase equal to the maximum queue length of that phase. The calculation of green time is then considered. The algorithm for each flow, processes the number of vehicles that passed during the previous traffic light cycle and uses it to determine the current traffic volume (Collata *et al.*, 2014). This value will then be used to re-calculate the green time duration of the next cycle based on detected traffic volume. The control of the green times also ensured enough pedestrian crossings occurred (Collata *et al.*, 2014).

2.2.7 Traffic Congestion

Traffic congestion is simply a condition on roadway which could occur as slower speeds, longer travel times and increased vehicular queuing. When traffic demand is high, the interaction between vehicles slows the speed of the traffic stream resulting to congestion (Ding, 2011).

2.2.8 Traffic Stream

Traffic stream are complex and non-linear phenomena which are defined by multi-dimensional traffic lanes with flow of vehicles over time (Ding, 2011). There are three main characteristics to visualize a traffic stream: speed, density and flow. These features are further explained as follows:

1. Speed (V): is defined as travel distance per unit time in traffic flow. The precise speed of each car is difficult to measure. In practice, the average speed is calculated using the sample vehicles. Speed is defined as,

   $$V = \frac{dx}{dt} \tag{2.5}$$

2. Traffic density (k): expressed as the numbers of vehicles per unit road. Traffic density is the inverse of spacing, which corresponds to the distance between two vehicles as in (Ding, 2011):

$$k = \frac{1}{s} \tag{2.6}$$

Where: *k* represents density; and

s represents spacing.

3. Traffic flow (q): defined as the numbers of vehicles per unit of time. In practice, it is usually counts as hour (Ding, 2011). Flow can be calculated as (Ding, 2011):

$$q = \frac{1}{h_{i+1} - h_i} \tag{2.7}$$

where:

$q =$ represents flow

$h =$ represents the i - th vehicle pass the settle point (resting point)

2.2.9 Single Intersection Base Model Formulation

In single intersections, an M/M/1 queue model is used to model each lane with random arrivals and exponential service times. These arrivals follow Poisson distribution with constant average rate λ. The length of the M/M/1 queue can be computed as follows (see Figure. 2.10). Assuming that each traffic signal is to be associated with a certain lane. The proportion of the time the traffic signal (server) is idle is assumed to be given by P_0 and the proportion of time the system is busy is given by ρ. Figure. 2.10 demonstrates that in the green time, the traffic signal queue has both arrivals and departures, while in the red time there are arrivals, but there are no departures. Hence, the queue length is expressed as (Yousef *et al.*, 2010):

$$Q_L = \frac{\rho^2}{1-\rho} \tag{2.8}$$

$$Q_L = \lambda W \tag{2.9}$$

Where: W: is the average time spends in the system, hence the *AWT* in the queue is expressed as:

$$W = \frac{Q_L}{\lambda} . \tag{2.10}$$

Therefore, queue length varies according to the following fundamental equation (Yousef *et al.*,2010):

$$Q_{Lj} = Q_{Lj-1} + \lambda_G G - \mu_G G + \lambda_R R \tag{2.11}$$

where:

j = represents the traffic cycle number,
Q_j = represents the expected queue length of all lane for the next cyclej,
Q_{Lj-1} = represents the expected queue length of one lane for the next cyclej,
λ_G = represent λ the arrival rate in the green phase,
λ_R = represents the arrival rate in the red phases and is considered equal to λ_G within the same cycle,
G = represents the green period of one phase in seconds, and
R = represents the red light period in seconds and is equal to difference between theT and the green time period.

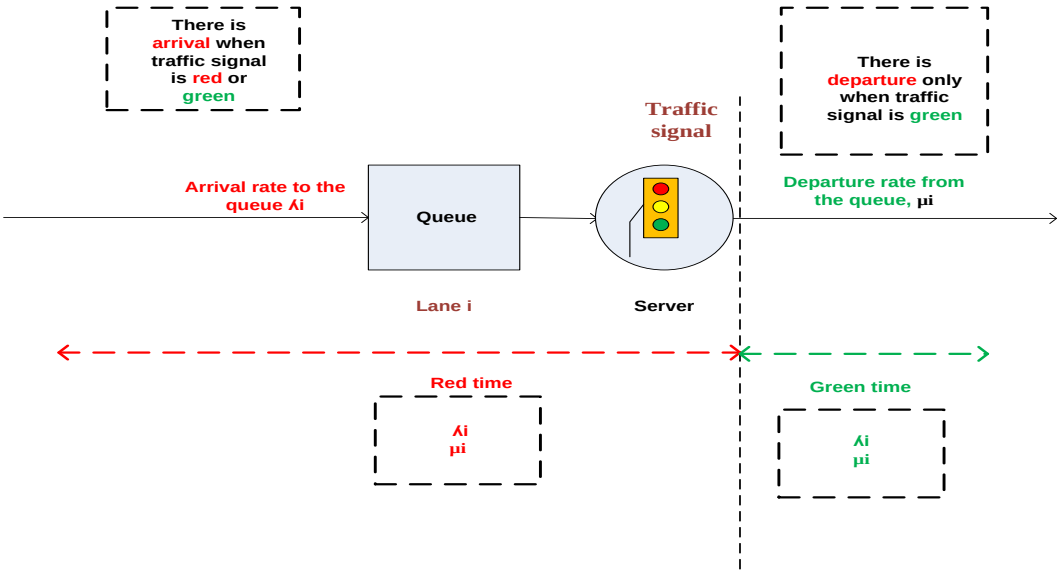


Figure 2.10: Traffic Signal Flow Queue (Yousef *et al.*, 2010).

Another important aspect that will be considered is the change in the queue length on the roadways. This is critical in computing the adaptive time control corresponding to the queue length change. To generalize the change of queue length for all the operating lanes twelve directions, equation (2.11) becomes

$$Q^{D}{}_{Lj} = Q^{D}{}_{Lj-1} + \lambda^{D} G^{D} - \mu^{D}{}_{G} G^{D} + \lambda^{D} R^{D} \tag{2.12}$$

2.2.10 Traffic Control on Multiple Intersection (TCAMI)

TCAMI has the ability to find an efficient time allocation to light signals at each single intersection. Despite traffic streams leaving one intersection exhibit an in-determinate behavior because of dependency between intersections. The design of TCAMI depends on the coordination and setting of traffic parameters (Faye *et al.,* 2013). Conditions on multiple intersections in general and on the successive intersections in specific aim to minimize delays. Multiple intersections are viewed as a set of nodes interacting with each other so that each intersection has the characteristics of the base model that correlates to an M/M/1 queue model. The TCAMI executed on each intersection will generate traffic information, which in turn represents an input to the subsequent intersection, and so on. Through this approach traffic flow will be controlled in a flexible manner (Yousef *et al.,* 2010).

2.2.11 Approaches used in Programming Traffic Signals

Sensing systems for intelligent transportation systems (ITS) are either fixed time controllers or vehicle actuated networked systems.

(a) Normal sequence

Sequencing method is the most common programming method used in the conventional traffic light system. It consists of normal sequence or fixed-cycle during its operation. The block diagram

for a normal sequence is shown in Figure 2.11. The traffic light system would operate according to the sequence from lane 1 to lane 2, followed by lane 3 and then repeat the cycle to lane 1. This approach does not consider real-time traffic behavior.

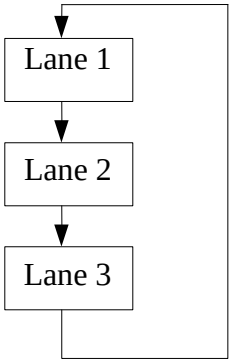


Figure 2.11: Block Diagram of Normal Sequence Programming (Subramaniam et al., 2012).

(b) Sensor - based Controller (vehicle actuated method)

In this approach, the magnetic loop detection sensor is mostly used. The sensor would be connected to the traffic light controller for the changes in the sensing and the traffic light controller would respond accordingly as seen in Figure 2.12 (Subramaniam *et al.*, 2012).

The method is more dynamic in scheduling extensions for green signals according to arrival rates but fails to consider the queuing length at all the red signal phases thus making it less efficient.

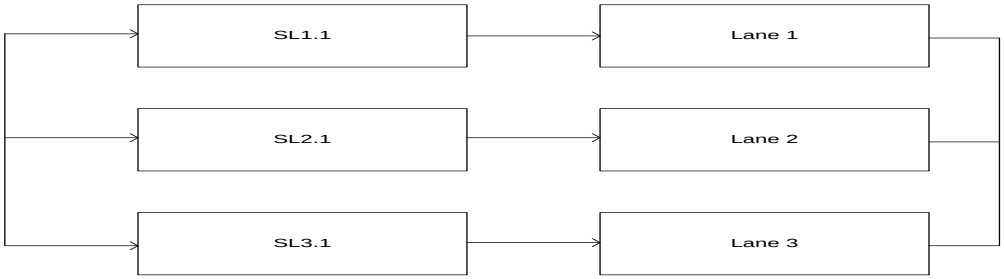


Figure 2.12: Block Diagram of Sensor in Each Road End Before Junction (Subramaniam et al., 2012).

2.3 Review of Similar Works

Quite a number of relevant literatures regarding vehicular traffic control system commonly installed at various intersections were reviewed. The following paragraphs summarises some of these similar/relevant literatures.

Tubaishat *et al.,* (2008) examined a wireless sensor based traffic light control. The researchers defined cycles by ordering three set of pre-defined phases (4-phase, 6- phase, 8- phase) to mimic conventional traffic controllers. The researchers analysed the performance of other controllers in comparison to fixed cycle. These selection were based on queue sizes. Several experiments were conducted to determine the suitable distance between two sensors in calculating queue length. Results showed that the distance between two sensors on the same road had no great effect on traffic volume. However, only queue length was considered. Other parameters such as average waiting time and other combination of possible movements on each lane were not taken into consideration.

Zhou *et al.,* (2010) developed an adaptive traffic light control algorithm that adjusts both the sequence and length of traffic light in accordance with the real-time traffic detected. The algorithm selected the sequence of phases among a set of conflict -free situations according to multiple criteria: traffic volume, priority vehicles, waiting time, starvation degree and queue lengths. Results obtained showed the algorithm maximized traffic throughput and minimized the average waiting time at an intersection in comparison with a fixed-time control algorithm and the actuated algorithm. However, the algorithm was based on the assumption that all vehicles were of the same type and run at the same speed.

Osigwe *et al.,* (2011) presented a hybrid methodology obtained by the crossing of the Structured System Analysis Design Methodology (SSADM) and the Fuzzy Logic based Design Methodology. An analysis of current traffic systems in Nigeria was carried out which necessitated the design of an intelligent traffic control system. Java software was used for simulation of the system using popular '+' junction in Eastern Nigeria notorious for traffic congestions. The system assumed that traffic movement is only from North to South as that from one side while East to West as traffic from another side. The fuzzy logic provided better performance in terms of total waiting time and total moving time. However, the fuzzy logic approach failed to adapt into complex intersection based on the 68% success rate for average vehicular traffic dissipated.

Faye *et al.,* (2012) developed a distributed algorithm for adaptive traffic light control. The algorithm decides in a dynamic fashion, the green light sequences by selecting movements comprising each phase and its duration. The system model made use of sensors organized into four hierarchical layers with several communication paths. Simulation results obtained showed proper tuning of the algorithm could reduce the average waiting time at an intersection. However, in the adaptive approach developed, choosing the appropriate value of the green light limit, T_{max} influenced performance of the average waiting time when conflicts were allowed and forbidden.

Subramaniam *et al.,* (2012) developed a novel sensing method which is capable of counting the total number of vehicles entering and exiting a certain junction on real-time basis. It was aimed at reducing the average waiting time. The programmable logic controller will trigger the traffic light indicators according to real-time demand. The self-algorithm traffic light system was designed to make use of three sensors in addition to one conventional sensor for detection of vehicles. The proposed system allowed the controller to detect the vehicle travelling out of each possible junction for detection during peak hours. The system was compared with the conventional traffic light and

results showed that the self-algorithm traffic light approach was efficient in choosing lane priority and less probability to traffic congestion. However, the work laid more emphasis on the long queue length which in some case might result in omitting lane with short queue(s).

Mohan & Rani, (2013) proposed an intelligent algorithm for traffic signal scheduling for a two lane crossed junction with dividers. The intelligent transport signal control (ITSC) comprised of sensors placed at all four phases of a crossed junction with an intelligent fuzzy controller. The data provided by the sensors determined the congestion characteristics in terms of area covered by vehicles at an intersection. The working of the ITSC were based on two timings: Non-peak and Peak hours. The process of phase scheduling was facilitated using two inputs to the fuzzy Logic controller (FLC): Area covered and the inter-arrival time with four and two membership functions and a single output. The simulation results showed positive results for improved phase scheduling and green signal allocation. The analysis showed that the intelligent transport signal control (ITSC) performed better than the conventional controllers in minimizing delay incurred at signalized intersections. However, the approach used only certain type of vehicles and sensors were installed at a distance of 100m away from the junction. This may not allow dissipation of the highest amount of vehicular traffic per unit time.

Odeh, (2013) proposed a system that aided in the management of an intelligent traffic light using genetic algorithm. The system detected the level of congestion and the abnormal situations in two main highways and four intersections. Collection of data was achieved using a video imaging system, which captured and interpreted images detected and count of the vehicles. The system made real-time decision that determined the interval of green light time for each traffic light at each intersection. The average accuracy of the counted vehicles was 96%. This was due to noise caused from detecting objects size too large or too small for consideration as a vehicle. However,

even though the algorithm was suited for obtaining global optimum solution, it required a longer simulation time to effectively implement the system.

Singh & Rewari, (2013) developed a novel approach for automatic control of road traffic congestion using imaging mosaicking technique. The paper discussed a new method named Structured System Analysis to analyse the existing traffic congestion problem and designing a new model using image mosaicking technique. By assembling multiple overlapping images of the same scene into a larger image. The algorithm proposed adopted the use of inter- arrival and inter-departure times to simulate the leaving of vehicles. The model provided good and satisfactory results. However, the model was not also suitable for real-time traffic situations during left and right turns for two – way road ways. Issues of False Rejection Rate in the image tracking system may occur thereby not allowing or authorizing vehicular flow at some instance.

Alam &Pandey, (2014) presented an advanced traffic light system based on congestion estimation using fuzzy logic. The number of cars sensed at the input of the fuzzy controller were converted to fuzzy values. The advance traffic light control system outperformed the fixed traffic light control or even the actuated traffic controllers. This was observed from results obtained in terms of total waiting time as well as total moving time at the intersections. However, the use of fuzzy logic may not be suitable in adapting to dynamic changes in traffic demands.

Collata *et al.,* (2014) developed a self- powered wireless sensor network for dynamic management of queues at traffic lights. The proposed architecture provides a technique to power the sensors based on piezoelectric materials generated through vibration of moving vehicles. A decision making algorithm was used to dynamically evaluate the cycles based on queue length. Results obtained were promising in terms of queue management and energy harvesting. However, the

approach used considered only queue length on lanes with higher priority. Simultaneous movements which does not results in conflict or overlap were not considered.

Gündoğan *et al.,* (2014) presented an evaluation of an adaptive traffic light control system in Istanbul, Turkey. The algorithm developed made use of fuzzy and genetic algorithm as optimization tool. The system determined optimum signal timings according to measured real-time traffic and occupancy. The adaptive system automatically calculated the signal timings and sends these new signal timing plans to control devices at the end of each cycle. The performance had a 10% improvement in terms of cycle time and 15% for travel time. The result showed that the adaptive system outperformed the dynamic intervention to vehicular traffic. However, decisions were taken at the end of each cycle leading to longer period in time for a lane to achieve green light signal timing.

Punetha *et al.,* (2014) proposed an intelligent traffic light system architecture for safe passage in adaptive traffic light. The system that helped save fuel consumption and time using active adaptation to speed on urban roads. The algorithm proposed featured prediction of speed, by incorporating Raspberry Pi model B interfaced with global positioning system (GPS) and global system mobile (GSM). A method of intelligent traffic light control was presented, with automatic interaction of traffic light with the vehicles using GPS. Based on simulation results were obtained, for interaction with and without traffic light adaptation algorithm. However, even though an adaptive traffic light was presented by the researchers. It was not evident that the system gave priority to lanes in order to reduce the average waiting time.

Saleh *et al.,* (2014) developed a hierarchical scheduling algorithm for congestion traffic control using multi-agent systems. The proposed control system was modelled based on Packet switched networking model, where different classes of real-time traffics (audio and video) requesting the best quality of service were guaranteed. The algorithm made use of Weighted Fair Queue (WFQ) First Come First Serve (FCFS) approach. According to multi-agent the system design was based on three main layers namely: decomposition, modeling and communication protocol. The aim was to minimize the vehicle delay time at a single intersection. Simulation results showed that the agent based algorithm outperformed baseline algorithm as the variance of arrival time increases. However, the system mostly considered queue length which may result in omitting lanes with shorter queues.

Salehi *et al.,* (2014) presented an application of fuzzy logic for multi- agent based autonomous traffic lights control system using wireless sensor networks. The real-time parameters such as traffic density and queue length were obtained by using image-processing techniques for the two junctions. Sensors were installed 500m away from the junctions. The system was designed to consider emergency vehicles (ambulance, police unit, and fire brigade) coming from three different directions at the same time with different speed ratio. Based on their speed detected by the sensors, two actions were performed. One, traffic flow is minimized on their routes so that they can pass by with maximum possible speed. The second approach was for collisions to be avoided. The proposed fuzzy logic system and fixed time controller produced little difference in results; in terms of constant traffic flow. While, in the case of time varying traffics, the proposed FLSC was superior to the fixed time controller. One of the limitation of the work was that the performance of the fuzzy logic approach was affected by the configuration of the membership functions. The input and output variables as well as rule base were difficult to configure.

Wang *et al.*, (2014) proposed an algorithm for vehicle queuing system using the Monte Carlo simulation technique. The developed algorithm was intended to reduce vehicle waiting time and vehicle queues at intersections in urban region. A queuing theory combined with the knowledge of static simulation (M/M/1 queuing model) was employed with a view to reducing the effect of traffic congestion. The Monte Carlo algorithm generated random numbers using the queuing model to realize improvement at the intersections of vehicles queuing system. However, the use of Monte – Carlo algorithm requires large amount of simulation leading to high computational time.

Erwan *et al.*, (2015) presented the design of an adaptive traffic light controller using fuzzy logic Sugeno method. The Fuzzy Logic was used to determine the green time at an intersection. The system made use of three inputs namely the number of queues, waiting time and traffic flow of vehicles. The design was applied in a simulation to observe the number of queues, waiting time, and the number of vehicles passing an intersection. The simulation results showed that the traffic light using fuzzy logic control performed than using fixed time control. The number of queues and waiting time were lower, and the number of departures was higher than using the fixed time controller for traffic light. However, using the Fuzzy Logic system can be improved upon by using simultaneous movements.

Most of the works reviewed carried out simulation using fixed volume of vehicles of the same kind and laid much more emphasis on queue length without considering the effect of a particular set of movements on the reduction of traffic congestion. Consequently, this research work is set out to develop a traffic management scheme using adaptive dynamic scheduling algorithm based on an Artificial Bee Colony (ABC) technique. The algorithm is to incorporate a set of probabilistic

event parameters governing the possible movement decision of each and every vehicle. Emphasis is laid on possible movements that will evacuate the highest number of vehicles per time.

CHAPTER THREE

MATERIALS AND METHODS

3.1 Introduction

In this chapter, the methods, materials and procedures employed for the successful completion of this research (as in section 1.7) are discussed. The mathematical model for the developed Vehicular Traffic Control System (VTCS) were presented and discussed. The steps involved in the development of the developed Adaptive Dynamic Scheduling Algorithm (ADSA) and the Graphic User Interface (GUI) simulator were also presented. Other materials that were useful in achieving the set objectives were also discussed.

3.2 Mathematical Model of Vehicular Traffic Control System (VTCS)

In this research work, MATLAB was used as an environment for simulation and therefore all the developed models are based on MATLAB. An n-road intersection may be considered as an n-dimensional problem in which each of the roads forming the intersection is considered as a separate entity. In order, to formulate a mathematical model for typical traffic road intersections, some parameters were chosen.

In order to ease the development of the VTCS model, consider a typical n-road intersection, having left and right lanes on each road. Therefore, the following assumption can be made:

1. Each of the road forming the intersection has two lanes that are wide enough to support three separate vehicle queue
2. Vehicles are allowed to queue-up on each lane according to their intended motion direction decision, i.e.
    a) Left decision → Left part of the lane
    b) Right decision → right part of the lane

c) Straight decision → center of the lane

3. Delay is not caused by service vehicles crossing the intersection in non-conflicting manner i.e. vehicles do not maneuver their way while at the intersection.
4. The sensors can detect vehicles on each of the lane regardless of their speed and position i.e. vehicles are accurately counted.

Let λ_i denote the average arrival rate of vehicles on the i^{th} lane measured over a considerable large time interval; Let μ_i indicate the average departure rate of vehicles on the i^{th} lane; let t denote a time interval. Therefore, the values of i will range from 1 to n, where n is the number of roads forming the intersection. The total number of vehicles at the intersection at any time can be expressed using (Yousef *et al.*, 2010):

$$N = N_O + \sum_{i=1}^{n} \lambda_i (T - t_o) - \sum_{i=1}^{n} \mu_i (T - t_o) \tag{3.1}$$

Thus:

$$N_o = \sum_{i=1}^{n} N_{Oi} \tag{3.2}$$

where: T is the current/present time instant.

N is the total number of vehicles at the intersection at a given time instant

N_O is the total number of vehicles on roads, at the time when monitoring begins

t_o is the start time / initial time

N_{Oi} is the initial queue length on the i^{th} road

$\sum_{i=1}^{n} \lambda_i (T - t_o)$ is the total number vehicles that have arrived within the interval $(T - t_o)$

$\sum_{i=1}^{n} \mu_i (T - t_o)$ is the total number vehicles that have departed within the interval $(T - t_o)$

Therefore, replacing $T - t_o$ with t, and re-organizing equation (3.1) can give rise to equation (3.2) as follows:

$$N = \sum_{i=1}^{n} N_{oi} + \lambda_i t - \mu_i t \tag{3.3}$$

Equation 3.3 represents the objective function of the VTCS which has to be minimized at any instant in time. Let λ represent the average arrival rate at the intersection; let μ represent the average departure rate of the intersection, equation (3.4) and (3.5) can be used to express λ and μ in terms of their equivalents for each of the roads (Yousef *et al.*, 2010).

$$\lambda = \sum_{i=1}^{n} \lambda_i \tag{3.4}$$

$$\mu = \sum_{i=1}^{n} \mu_i \tag{3.5}$$

The average waiting time on the i^{th} road can therefore be expressed as:

$$AWT_i = \frac{N_{oi}}{\lambda_i} \tag{3.6}$$

While, the average waiting time for the entire system is also expressed as in (Yousef *et al.*, 2010):

$$AWT = \frac{N}{\lambda} \tag{3.7}$$

3.2.1 Developed Intersection Model

In this work, the VTCS was developed based on a cross-junction fitted with traffic signals and sensors. Though, the model was also extended to cover other kinds of intersection such as the three-road and five-road intersections. This can be achieved using the Graphic User Interface (GUI) where provision for number of roads has been made. The developed simulator could be used in executing physical vehicular motion control. Figure 3.1 shows a cross -road intersection as generated by the developed simulator.

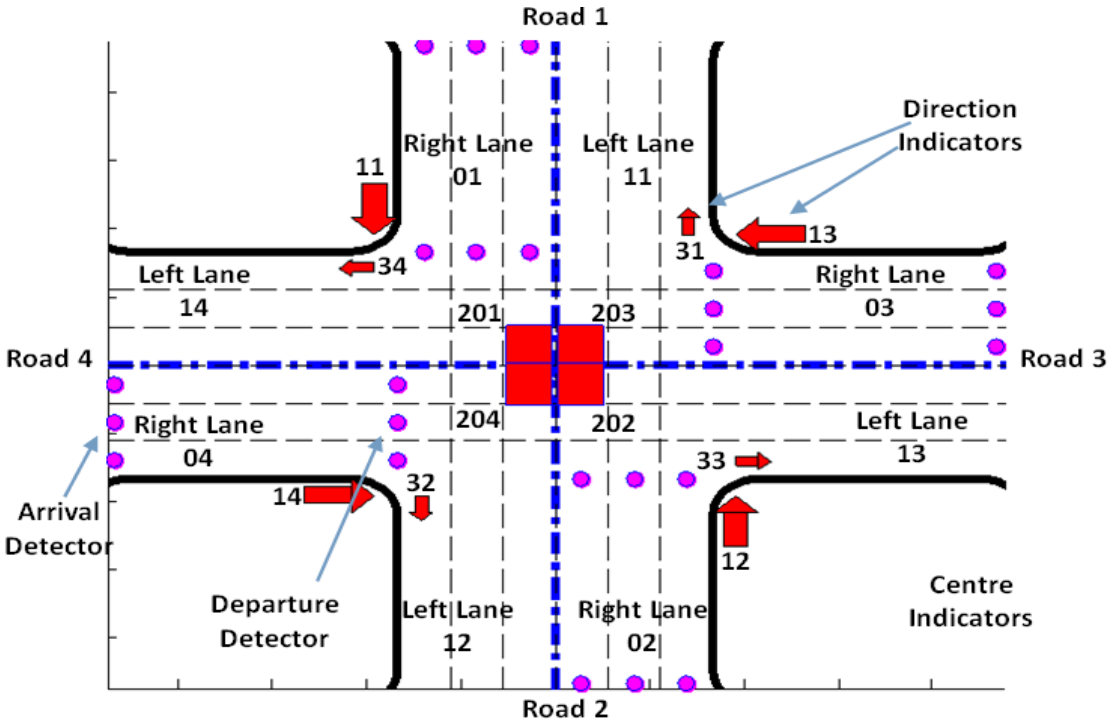


Figure 3.1: Cross – Road Intersection as Generated by the Developed Simulator

The indicators shown in the Figure 3.1 are deployed for demonstration purpose and therefore may not represent the actual traffic signals configuration in the real life. A single indicator in Figure 3.1 represents a bank of traffic signals placed in the appropriate manner and at the right location so to cause vehicular traffic flow in the direction indicated by the indicator. In the developed

model, each lane is represented by a double digit array / number as further described in the following:

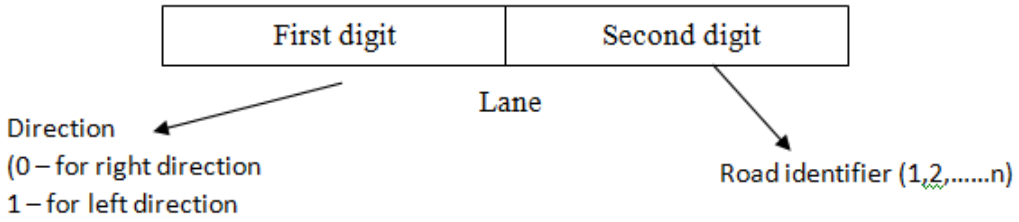


Figure 3.2: Coding a Lane at the Intersection

For example, 01 is used to represent the right lane of road 1, while 14 denotes the left lane of road 4. There is no particular manner in which the roads must be numbered, but what matters is that once the roads are numbered, the order remains the same throughout the simulation. The indicators / traffic signals are also coded in the same pattern as to allow vehicles to be serviced accordingly. A serviced vehicle must pass through a right lane of one road to the left lanes of the other. Therefore, three set of control signal are involved as further described in the following:

1. An indicator that command the vehicle to cut into the intersection (service start indicator) usually located at the right edge of a lane
2. An indicator that command vehicle to move across the lane usually located at the straight /center intersection.
3. An indicator that command vehicle to cut out of the intersection (service end indicator) also located the right edge of the cut out lane.

These set of traffic indicators are also tagged with 2 digit coded number, with the first digit indicating the position (Start or end) and the second digit denoting the road number or zero (center). The center indicator also has additional digit. E.g. 11, 12, 13 and 14 represent the start indicator of road 1, 2, 3 and 4 respectively; 31, 32, 33 and 34 represent the end signal indicator of

road 1, 2, 3 and 4 respectively while 201, 202, 203 and 204 represent the center indicator corresponding to the right lane of roads 1, 2, 3 and 4 respectively.

In the developed model, vehicles are modelled as entities that have both direction decision variable δ and service duration requirement variable τ, such that τ, depends on the position of a particular vehicle on the queue. The closer the vehicle is to the intersection the lesser the value of τ. The δ is a four digit number that denotes the arrival and departure lane. A vehicle with δ = 0113, is intended to move from the right lane of road 1 to the left lane of road 3. This motion can only occur if the appropriate indicators are triggered as green. For a cross - road intersection, the green light command of 1120120233 could trigger δ = 0113 as in section 3.2.1, Example (a). The duration of the green light give rise to the total service time of the vehicles whose decision variable is δ =0113. These descriptions are based on the cross - road intersection shown in Figure 3.1

The developed VTCS is characterized by optimal selection of green light duration in order to minimize equation (3.3). Therefore, the VTCS control parameters explained earlier can be summarized in Figure 3.3.

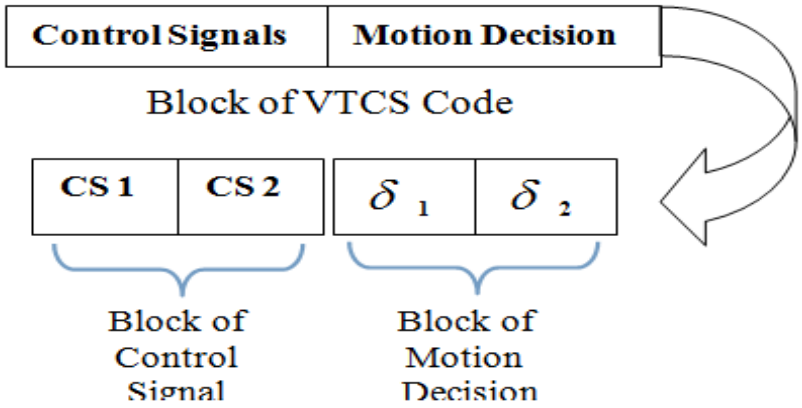


Figure 3.3: VTCS Control Parameters

As illustrated in Figure 3.3, a block of VTCS code comprises of two parts namely a control signal code and a motion decision variable, which are both further divided into two control signals (CS1 and CS2) and two motion decision variables δ_1 and δ_2. This means that a single block of VTCS code could comprise of several CSs and δ s depending on the desired control decision. Examples of VTCS codes are:

(a) 11 201 202 33 0113

(b) 11 34 12 33 13 31 14 32 0114 0213 0311

The VTCS codes are generated using the proper green light timing from the cross-road intersection in section 3.2.1

The pseudo code for decoding a VTCS code is given in algorithm 1.

Algorithm 1: VTCS Model

1. Input: VTCS Code;

2. Set m = number of 4 – digit integer within the VTCS code

3. for i= 1:m

4. split the i^{th} integer into two digit integer (representing lane code)

5. end

6. Split the VTCS Code into 2, from the point where the 4-digit integers begins

7. Starting from the right, form a subset of control signals (starting and ending with a two digit integer)

8. Read the integers in each CSs separated by a space.

9. Stop (If the number of CS equal to m)

Thus, the VTCS code shown in example (a) can be described / decoded as in Figure 3.4

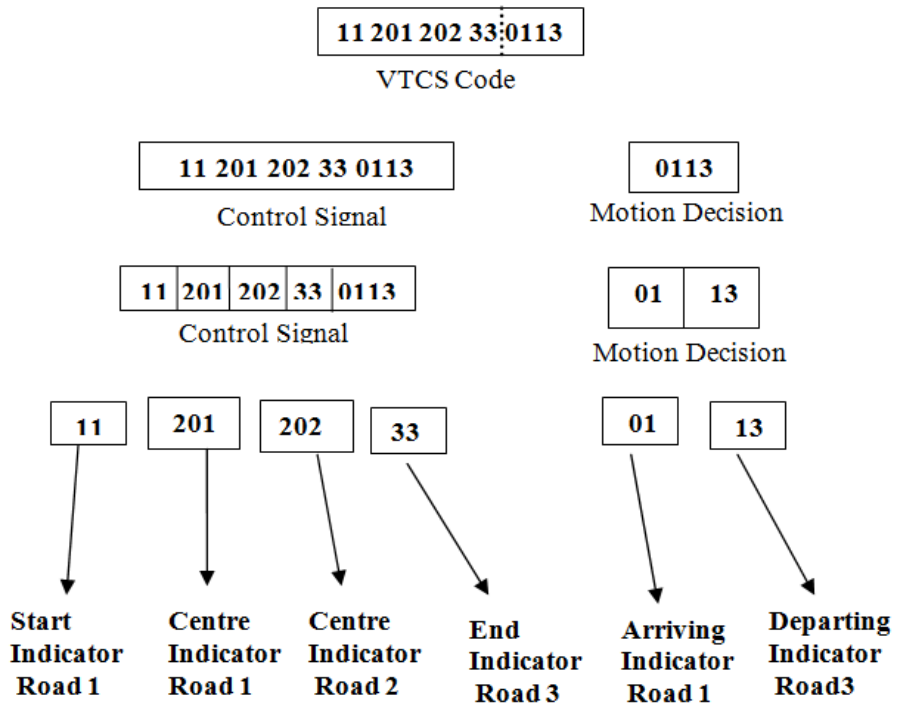
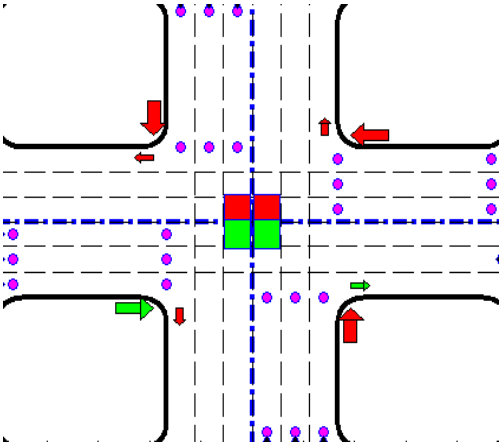


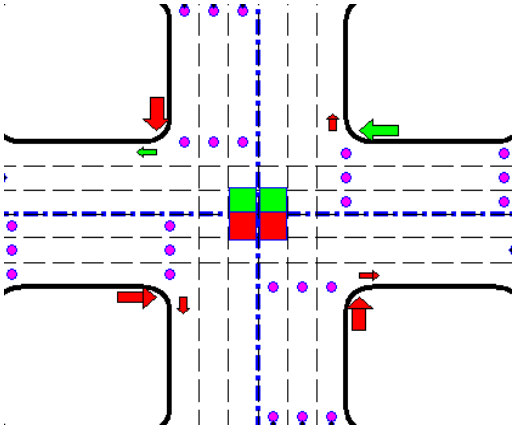
Figure 3.4: Procedure for Decoding a VTCS Code.

3.3 Traffic Phases

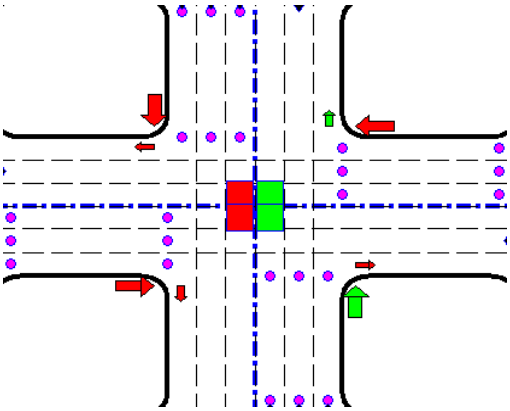
The developed Vehicular Traffic Control System (VTCS) model is characterized with a set of possible movement that does not conflict. These set of movements are termed as traffic phases. In this work, 22 possible phases were considered. This enables a variety of traffic control strategy to be implemented in order to minimize the average waiting time. The developed simulator was used to simulate all the possible traffic phases and presented in Figures 3.5 (1 to 22). The VTCS code for each of the traffic phases is also presented.



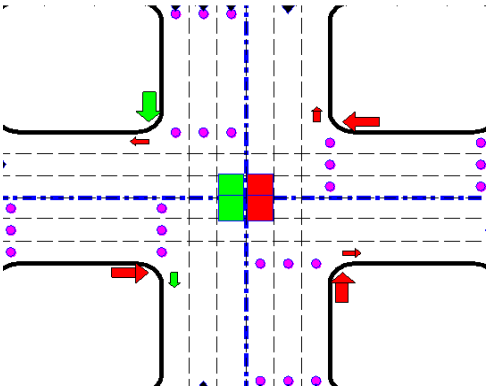
(1) VTCS Code: 14 204 202 33 04 13



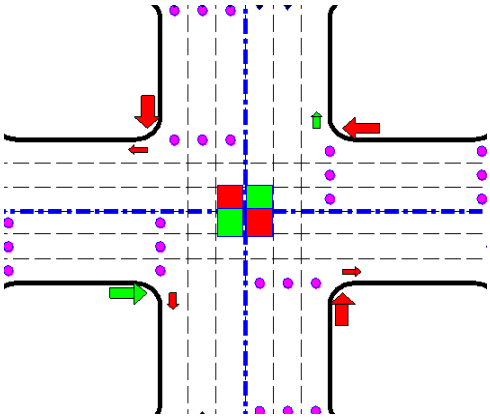
(2) VTCS Code: 13 203 201 34 13 31 0314 0311



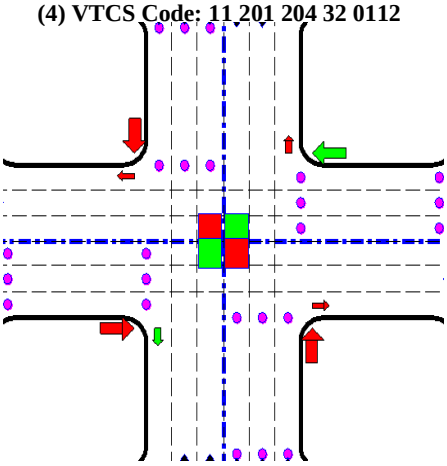
(3) VTCS Code: 12 202 203 31 0211



(4) VTCS Code: 11 201 204 32 0112

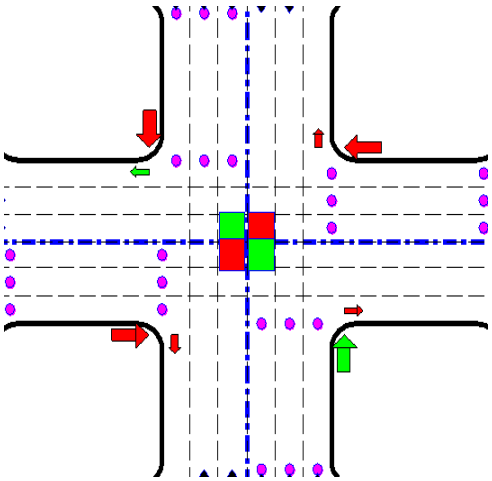


(5) VTCS Code: 14 204 203 31 0411

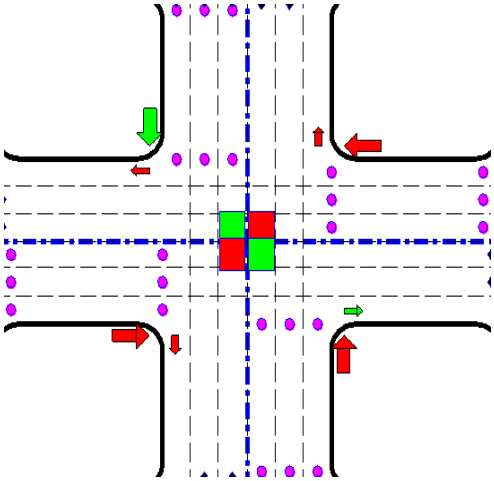


(6) VTCS Code: 11 203 204 32 0312

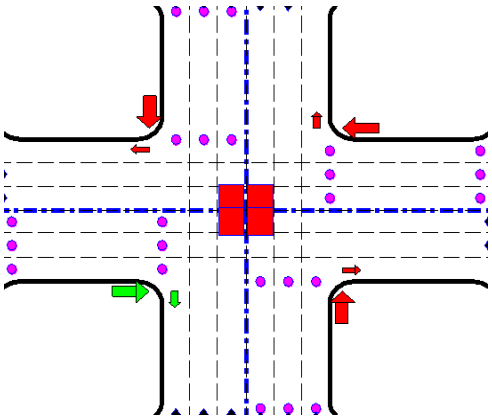
Figure 3.5: Traffic Phases



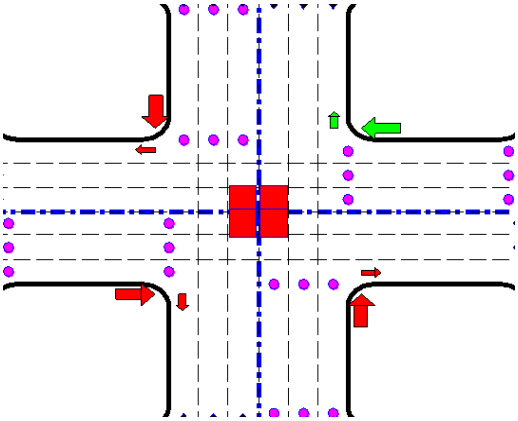
(7) (VTCS Code:12 202 201 34 0214



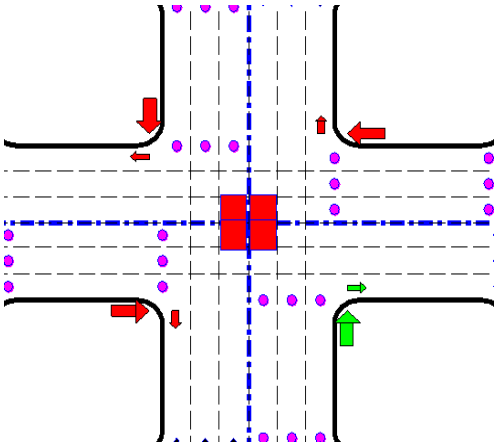
(8) VTCS Code: 11 201 202 33 0113



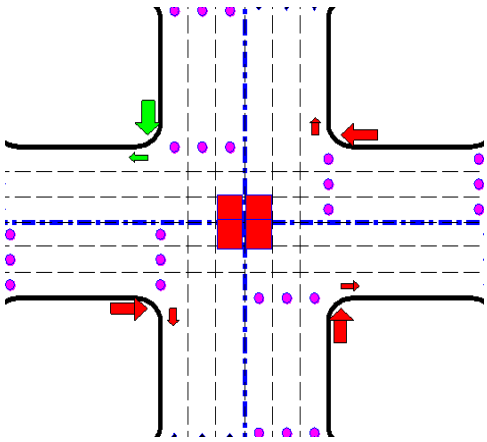
(9) VTCS Code: 14 32 0412



(10) VTCS Code: 13 31 0311

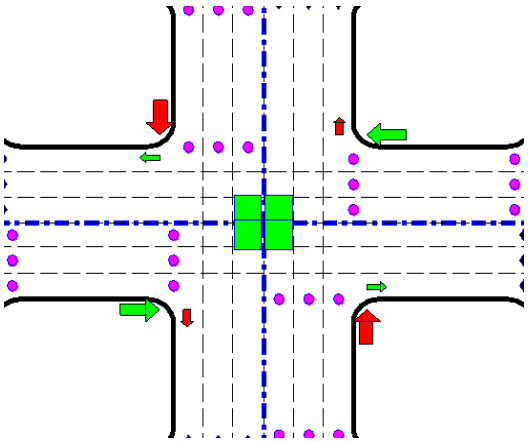


(11) VTCS Code: 12 33 0213

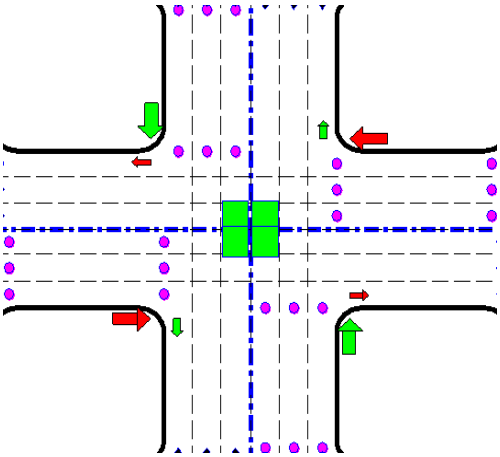


(12) VTCS Code: 11 34 0114

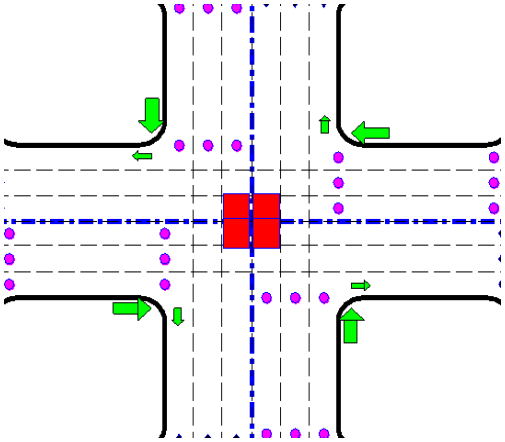
Figure 3.5: Traffic Phases



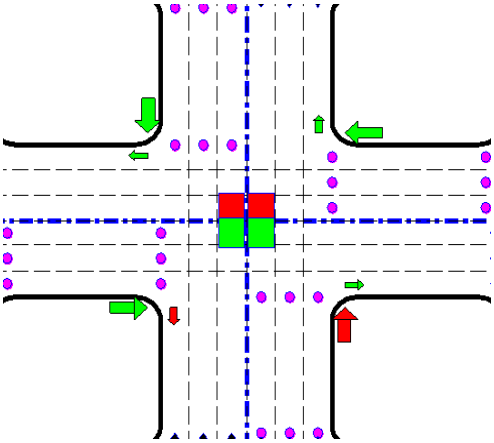
(13) VTCS Code: 13 203 201 34 14 204 202 330413 0314



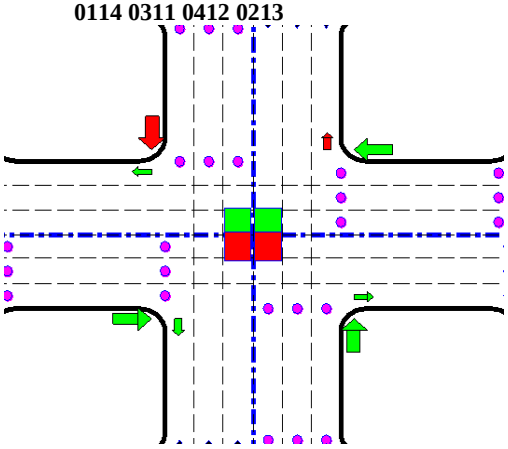
(14) VTCS Code: 11 201 204 32 12 202 203 31 0212 0111



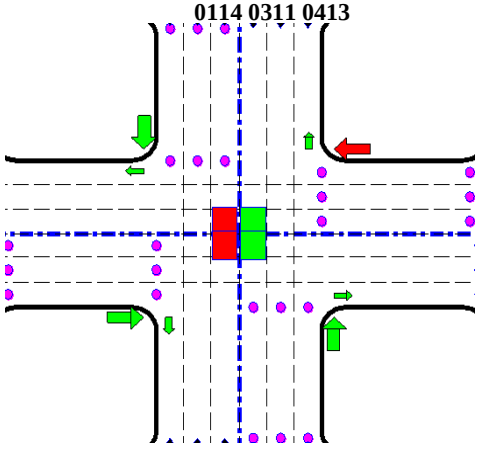
(15) VTCS Code: 11 34 13 31 14 32 12 33 0114 0311 0412 0213



(16) VTCS Code: 11 34 13 31 14 33 0114 0311 0413

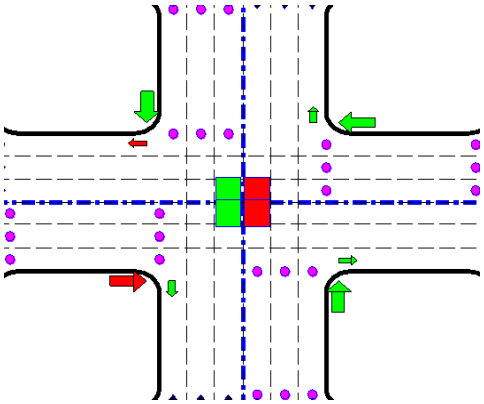


(17) VTCS Code: 13 203 201 34 14 33 14 32 12 33 0314 0413 0213 0412

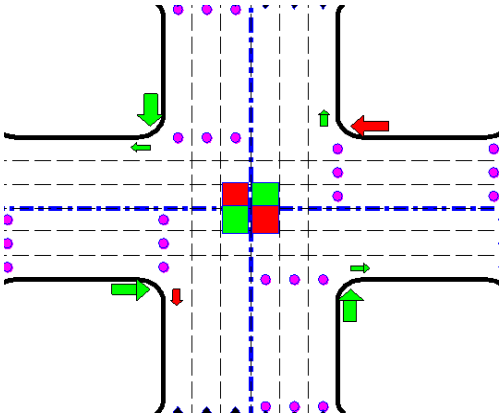


(18) VTCS Code: 12 202 203 31 12 33 11 34 14 32 0211 0114 0412 0213

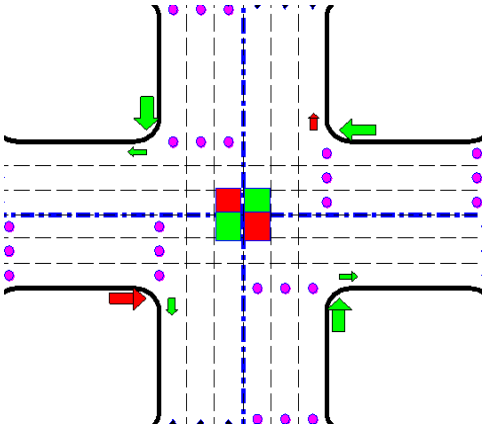
Figure 3.5: Traffic Phases



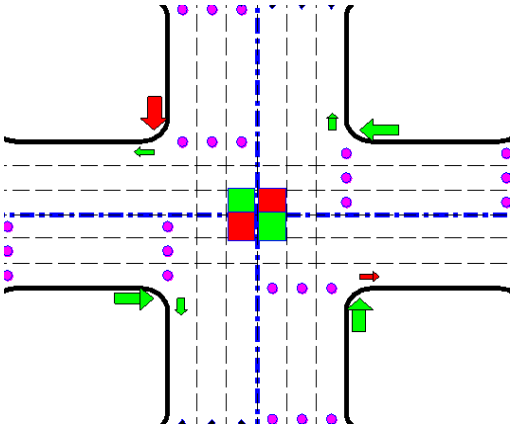
(19) VTCS Code: 11 201 204 32 13 31 12 33 0112 0311 0213)



(20) VTCS Code: 14 204 203 31 11 34 12 33 0411 0114 0213



(21) VTCS Code: 13 203 204 32 11 34 12 33 0311 0114 0213



(22) VTCS Code: 12 202 201 34 14 32 13 31 0214 0412 0311

Figure 3.5: Traffic Phases

In Figure 3.5 (1 to 22), the green light configuration responsible for each of the traffic phases is shown. The VTCS codes are generated using the green light timing from the cross-road intersection in section 3.2.1. It is worth noting that the other form of control signal such as the red light, amber etc. also depend on t_g . These 22 phases serve as a snap-shot of the intersection which can easily be saved for reference purposes (especially during post fault analysis).

3.4 Artificial Bee Colony (ABC) Algorithm based Vehicular Traffic Control System

In this work, ABC algorithm is employed to determine the appropriate green light duration for each of the roads forming the intersection in order to minimize equation 3.3. This will in turn minimize the average waiting time which is dependent on the queue length.

Thus, the objective function can therefore be described as in equation (3.8).

$$F = \min\left(N\left(t_g\right)\right) \tag{3.8}$$

Subject to:

$$0 \le t_g \le t_{g\max} \tag{3.9}$$

$$\delta \in \phi \tag{3.10}$$

$$CS \in S \tag{3.11}$$

where: t_g is the green light duration;

$t_{g\max}$ is the maximum allowable green light duration

δ is the motion decision executed

ϕ is the set of allowable traffic phases

CS is the control signal / green light configuration

S is the set of allowable signals / green light configurations

Therefore, t_g is the control parameter which must be optimized in order to achieve a reliable and effective traffic control. It is worth noting that the other form of control signal such as the red light, amber etc. also depend on t_g . Modern VTCS make use of counters which counts down thereby notifying the waiting vehicles to get ready or not. The absence of green light timing (t_g) denotes the red light. It can be observed from the conventional objective function of ABC algorithm shown in equation (2.2) that, it best suites minimization problem. This is based on the following facts.

1. $if \quad f(x_i) > 0, \quad \frac{1}{(1+f(x_1))} < 1$

and

2. $if \quad f(x_i) = 0, \quad \frac{1}{(1+f(x_1))} = 1$

Meaning that the objective function will have maximum value of unity. For values of $f(x_1)$<0, the objective function becomes greater than zero. This case is rare since the populations are usually set of random values ranging between zero and one

In order, to make ABC a suitable choice for VTCS, it is necessary to restrict the manner in which the population of bees are generated and develop an objective function F_{obj} that could fit into equation (2.2). The ABC based VTCS objective function is developed as follows:

From equation (3.3)

$$N(t_g) = \sum_{i=1}^{n} N_{oi} + \lambda_i t_g - \mu t_g \qquad (3.12)$$

Thus,

$$F_{obj} = N\left(t_g\right) \tag{3.13}$$

But from equation (2.2), when $f\left(x_1\right)$ is greater or equal to zero;

$$F_{obj} = \frac{1}{1+ f\left(x_1\right)} \tag{3.14}$$

Thus: making $f\left(x_i\right)$ the subject and substituting for F_{obj}, yields equation (3.15).

$$f\left(x_i\right) = \frac{1-\sum_{i=1}^{N}\left(N_{OI} + \lambda_i t_g - \mu t_g\right)}{\sum_{i=1}^{n}\left(N_{OI} + \lambda_i t_g - \mu t_g\right)} \tag{3.15}$$

Since N_{OI} is independent of the present green light timing it can be omitted from equation (3.15) to yield:

$$f\left(x_i\right) = \frac{1-t_g\sum_{i=1}^{n}\left(\lambda i - \mu_i\right)}{t_g\sum_{i=1}^{n}\left(\lambda_i \ - \mu_i\right)} \tag{3.16}$$

Let $k = \sum_{i=1}^{m}\left(\lambda i - \mu_i\right)$ (3.17)

Therefore,

$$F\left(x_i\right) = \frac{1-t_g k}{t_g k} \tag{3.18}$$

Let t_g be a function of the population of bees (x_i), i.e.

$$t_g = g(x_i) \tag{3.19}$$

where, $g(x_i)$ is a function of (x_i). Thus, to limit the value of t_g, a maximum value must be defined. In this work, $t_{g\max}$ limits the value of t_g. Since x_i represent the i[th] bee, x_i is a row of np is a set of random real numbers taken in the range (0,1), t_g can be redefined as follows.

Let:

$$g(x_i) = \frac{\sum_{k=1}^{np} x_i}{np} \tag{3.20}$$

Therefore:

$$t_g = t_{g\max} g(x_i) \tag{3.21}$$

Substituting equation (3.20) into (3.21)

$$t_g = \frac{t_{g\max} \sum_{k=1}^{np} x_i}{np} \tag{3.22}$$

As can be observed from equation (3.22), t_g will have a maximum value of $t_{g\max}$. Substituting equation (3.22) back into equation (3.18) yields

$$f(x_i) = \frac{np - kt_{g\max} \sum_{k=1}^{np} x_i}{kt_{g\max} \sum_{k=1}^{np} x_i} \tag{3.23}$$

Equation (3.23) represents the ABC based VTCS objective function.

3.5 Developed Vehicular Traffic Control Algorithm (VTCA)

The developed VTCA is based on ABC algorithm which dynamically schedule green light timing that adapt or suits the condition of vehicular traffic at the intersection. Since the problem of traffic control cannot be easily and accurately predicted, an Adaptive Dynamic Scheduling Algorithm (ADSA) will best suit solution of such problem. ADSA makes use of information from sensors to schedule green light timing. The following sequence of steps can be used to accomplish the ADSA for vehicular traffic control.

1. Input the number of roads n forming the intersection and the simulation time T.
2. Generate all the possible Traffic Phases
3. Store the VTCS codes for each of the traffic phases
4. For road 1:n estimate the queue length and sort the roads in descending order of queue length
5. Store the sorted road as R_s
6. Starting from the first element in R_s, select the most appropriate traffic phase for each road.
7. Determine the optimal green light timing duration for each of the selected phases using ABC
8. Compute the AWT, Cycle time, and Percentage Performance. Store the results

9. While the simulation time is not exhausted, increment the time and repeat step 4 to 8
10. Plot the relevant graph and stop

Let: n_o be the number of road forming the intersection

ϕ be set of all possible traffic phases

T be the total simulation duration

t be an instant of time

n_i be a set of road that have not been served such that $n_i = \{1,2,\ldots n_o\}$ if no road have been serviced

n_s be a set of road that have been served such as that $n_s = \{1,2,\ldots.n_o\}$ if all the road have been serviced

Cc be the number of cycles performed

t_{gmax} be the maximum allowable green light duration

R be the road with maximum queue length

Rs be a set of road that are served in a particular traffic phase such that $R \in Rs$

popsize be the number of artificial bees to be employed;

np be the number of parameters in each of the artificial bees;

maxgen be the maximum number of generation of the artificial bees;

Before presenting the flowchart of the ADSA it is necessary to develop some other sub-algorithms namely:

1. Road Selection Algorithm (RSA)
2. Optimal Green Light Timing Algorithm (OGTA)
3. Intersection Decongestion Algorithm (IDA)

The pseudo code for these three algorithms are presented in algorithms 2, 3 and 4 respectively

Algorithm 2: Road Selection Algorithm

1. Input values for: n_i n_s

2. for k = 1: number of element in n_i

3. Estimate the queue length L(k) of road n_i *(k)*

4. if k = = 1

5. R = n_i *(K)*

6. else

7. R = max (L(K), L (k-1))

8. end

9. end

10. Eliminate R from n_i

11. Add R to n_s

12.Print R and stop

Algorithm 3: Optimal Green Light Timing Algorithm

1. Input values for: np, popsize, maxgen, ϕ_R

2. Evaluate the queue length L on road R

3. if L == 0

4. $t_g = 0$

5. else

6. Determine t_g *using ABC algorithm with equation 3.23 as objective function*

7. end

8. Print t_g and stop

Algorithm 4: Intersection Decongestion Algorithm

1. Input values for: N_o, μ, λ, t_g, R_s, n

2. for k = 1

3. if k is not contained in R_s

4. $N_o(k) = N_o(k) + \lambda(k)\, t_g$

5. else

6. $N_o(k) = N_o(k) + \lambda(k) - \mu(k) t_g$

7. end

8. end

9.Print t_g and stop

The flowchart for the developed algorithm is shown in Figure 3.6

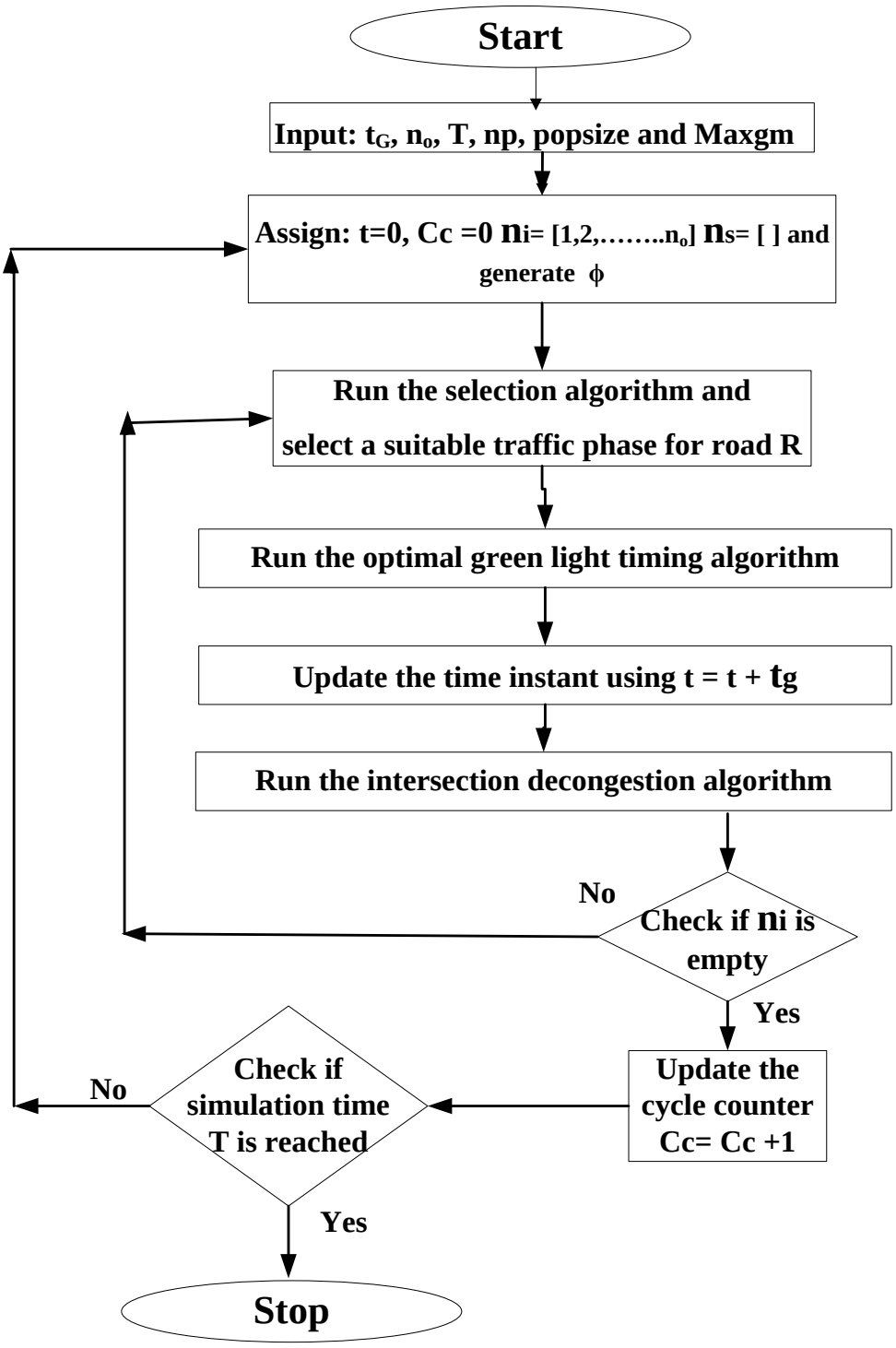


Figure 3.6: Flow Chart for Adaptive Dynamic Scheduling Algorithm for VTCS.

3.6 Vehicular Traffic Control Simulator

In order to ease the development of the vehicular traffic congestion control algorithm, a MATLAB based Graphic User Interface (GUI) fitted with several buttons as shown in Figure 3.7 was developed. Each of the buttons is programmed to carry out a particular function. The GUI shown in Figure 3.7 is designed to handle a cross-road intersection as may be easily observed on the left hand side of the GUI.

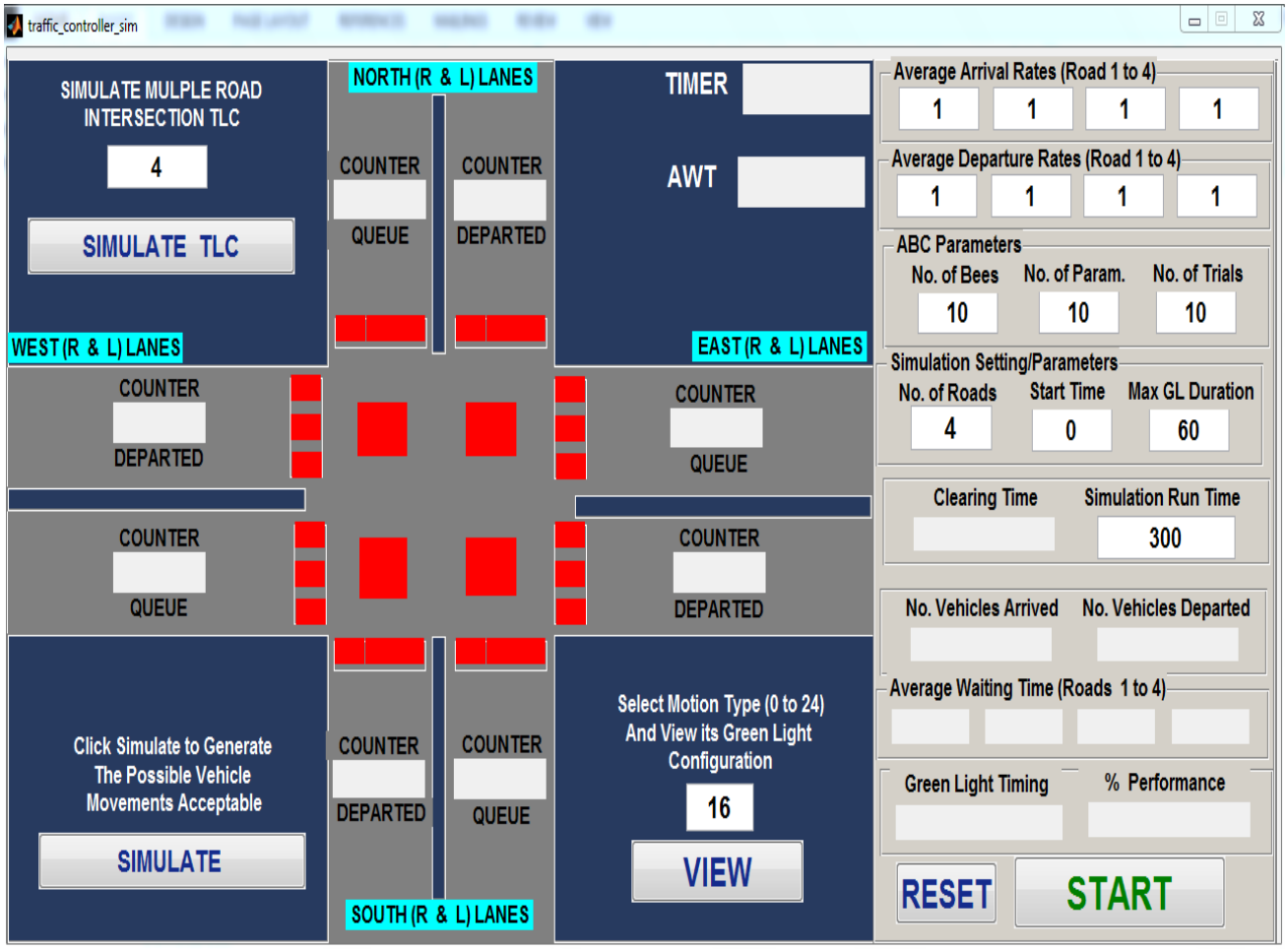


Figure 3.7: Graphic User Interface of the Developed Simulator

The functionalities incorporated in the developed simulator are further described in Table 3.1:

Table 3.1: Functional Buttons of the Developed VTCS Simulator

Button Name	Description
START	Run the ADSA and generate the relevant graphs
RESET	Resets the texts displayed in all the textboxes
VIEW	Generate the green light configuration for a selected traffic phase
SIMULATE	Demonstrate all the possible traffic movement across the intersection
SIMULATE	Simulates other possible road intersection e.g 3- Road, 5 -road, 6 road
TLC	e.t.c

An important metric for measuring the performance of the developed algorithm is the percentage performance of the algorithm or its efficiency (η). This can be estimated using equation (3.24) (Singh *et al.*, 2009).

$$\eta = \left(\frac{\sum_{i=1}^{n} L_{DI}}{\sum_{i=1}^{n} (L_I + L_{DI})} \right) \times 100\% \tag{3.24}$$

where:

n is the total number of roads

L_I is the queue length of the ith road

L_{DI} is the total number of vehicles that have departed from road i throughout the simulation duration.

3.7 Mode of Communication between Wireless Sensor Detectors and ABC Algorithm

In this work, wireless sensor detectors formed the basis of the type of sensor used for vehicular detection. The sensors responsible for detection of arrival and departures on the lanes are known as Reduced Function Devices (RFD's). These sensors gather the necessary data accurately (vehicle detection, queue length) on each lane which is transferred to Full Function Devices (FFD). The detection of vehicles is also dependent on the maximum authorized green light timing which the ABC algorithm optimizes in order to minimize the Average Waiting Time (AWT). The First Pan Coordinator (FPC) which works as a base station makes use of the polls data gathered by the Full Function Devices (FFD) through the Reduced Function Devices (RFD). The FPC enhance access in modifying traffic light behavior and timing. Using this approach the appropriate Traffic Phase is achieved thereby minimizing congestions at vehicular intersections.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Introduction

In this section, the developed simulator was used to simulate a variety of vehicular traffic control in order to demonstrate its effectiveness. A number of graphs describing the condition of the intersection during simulation are generated, presented and discussed. Furthermore, the effectiveness of the developed algorithm is also demonstrated through the simulation of the scenario presented in the work of (Erwan *et.al*, 2015) and comparing the results.

4.2 Simulation

A set of simulation parameter settings were selected as shown in Table 4.1 in order to demonstrate the peculiar results that could be generated by the developed simulator. In the table, three sets of scenario settings are shown which are further discussed.

The Table 4.1, 4.2 and 4.3 consists of seven columns which comprises of the Average Arrival Rate (AAR), Average Departure Rate (ADR), ABC parameters and other parameters which includes Start time, green light timing, t_{gmax} and simulation duration. Equal values of AAR and ADR rates were considered using five (5) vehicles per minute for scenario one. While, scenario two considered unequal values of AAR and ADR. Simulation at 1800s was also made at scenario three.

Table 4.1: Simulation Parameter Settings for the Developed Scenarios

Scenario 1 (Constant Arrival & Departure Rates)						
Road SN.	**AAR**	**ADR**	**ABC par.**	**Value**	**Other parameters**	**Value**
1	5/60	5/60	*Popsize*	10	Start Time	0
2	5/60	5/60	*Maxgen*	10	Tgmax	50 - 150s
3	5/60	5/60	*Np*	10	Sim. Duration	300
4	5/60	5/60				
Scenario 2 (Variable Arrival & Departure Rates)						
Road SN.	**AAR**	**ADR**	**ABC par.**	**Value**	**Other parameters**	**Value**
1	1(5/60)	4(5/60)	*Popsize*	10	Start Time	0
2	2(5/60)	3(5/60)	*Maxgen*	10	Tgmax	50 - 150s
3	3(5/60)	2(5/60)	*Np*	10	Sim. Duration	300
4	4(5/60)	1(5/60)				
Scenario 3 (Arrival and Departure Rates at 0.2 & 0.5 respectively)						
Road SN.	**AAR**	**ADR**	**ABC par.**	**Value**	**Other parameters**	**Value**
1	0.2	0.5	*Popsize*	50	Start Time	60
2	0.2	0.5	*Maxgen*	20	Tgmax	120
3	0.2	0.5	*Np*	10	Sim. Duration	1800
4	0.2	0.5				

4.2.1 Scenario 1: Constant Average Rate of Arrival (AAR) and Departure (ADR)

This scenario is intended to simulate a case in which AAR equals ADR across the cross-road. In this scenario, the maximum allowable green light duration is varied between 50 and 150 seconds and simulated over 300 seconds. The values for AWT were noted and plotted against the maximum allowable green light duration as shown in Figure 4.1. Similarly, the values for percentage performance were noted and plotted against the maximum allowable green light duration as shown in Figure 4.2.

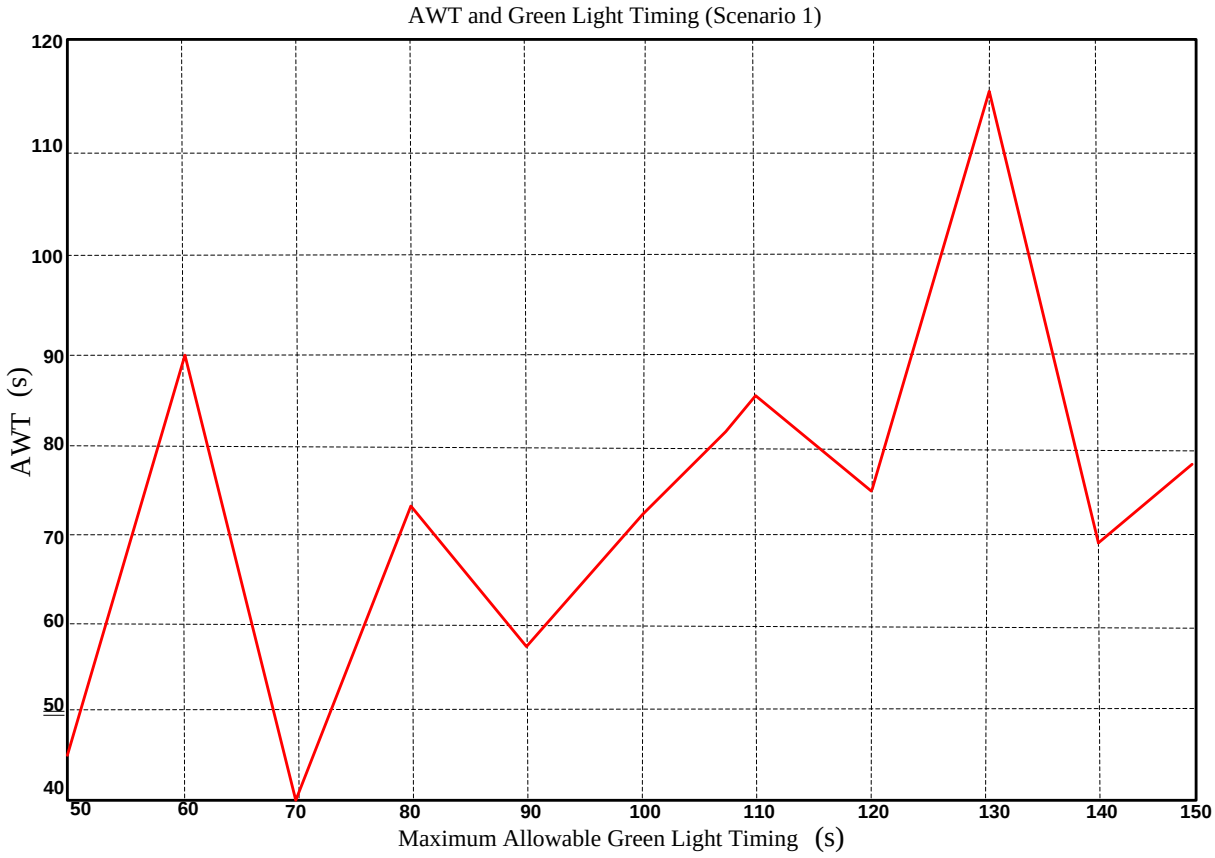


Figure 4.1: Variation of AWT with T_{gmax} (Scenario one)

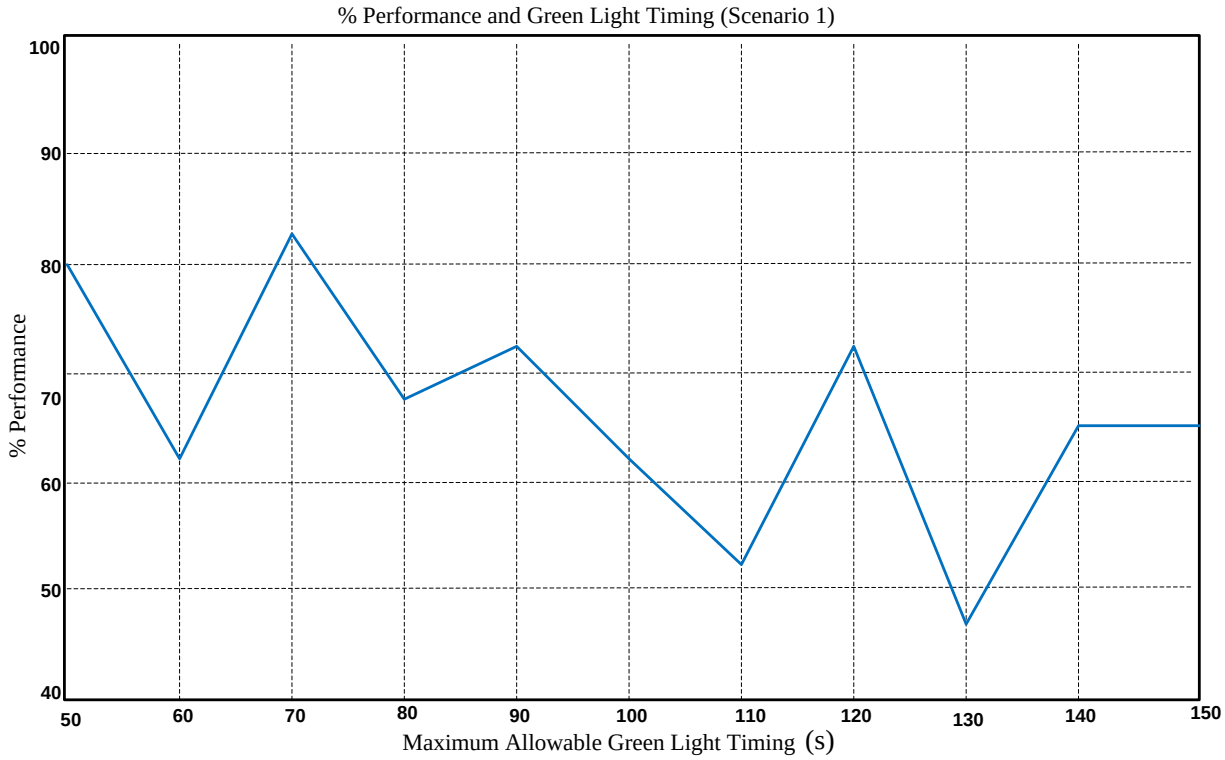


Figure 4.2: Variation of Percentage Performance with T_{gmax} (Scenario one)

A vehicular traffic controller is designed with the aim of maintaining the average waiting time within a safe maximum. The lower the AWT, the better the controller. AWT may vary depending on the condition of the intersection at any given instant.

As may be observed in Figure 4.1, the points corresponding to T_{gmax} equal to 60 and 130 marks the points at which the AWT have peak values. The curve of the AWT has a Zig-zag shape which implies that the AWT varies randomly with change T_{gmax}. This is due to dynamic nature of green light duration allocated to various roads forming the intersection. It can be also observed that despite the wide variation in AWT, the value of AWT was only higher than the selected T_{gmax} once. (i.e at AWT = 90s , corresponding to T_{gmax} = 60s).

On the other hand, the percentage performance is inversely proportional to the AWT (i.e decreases with AWT). The percentage performance of the algorithm has a minimum and maximum value of approximately 47 and 83 respectively.

4.2.2 Scenario 2: Variable Average Rate of Arrival (AAR) and Departure (ADR)

This scenario is intended to simulate a case in which AAR and ADR are unequal across the cross-road. In this scenario, the maximum allowable green light duration is also varied between 50 and 150 seconds and simulated over 300 seconds. The values for AWT and percentage performance of the algorithm were also noted and plotted against the maximum allowable green light duration as shown in Figure 4.3.

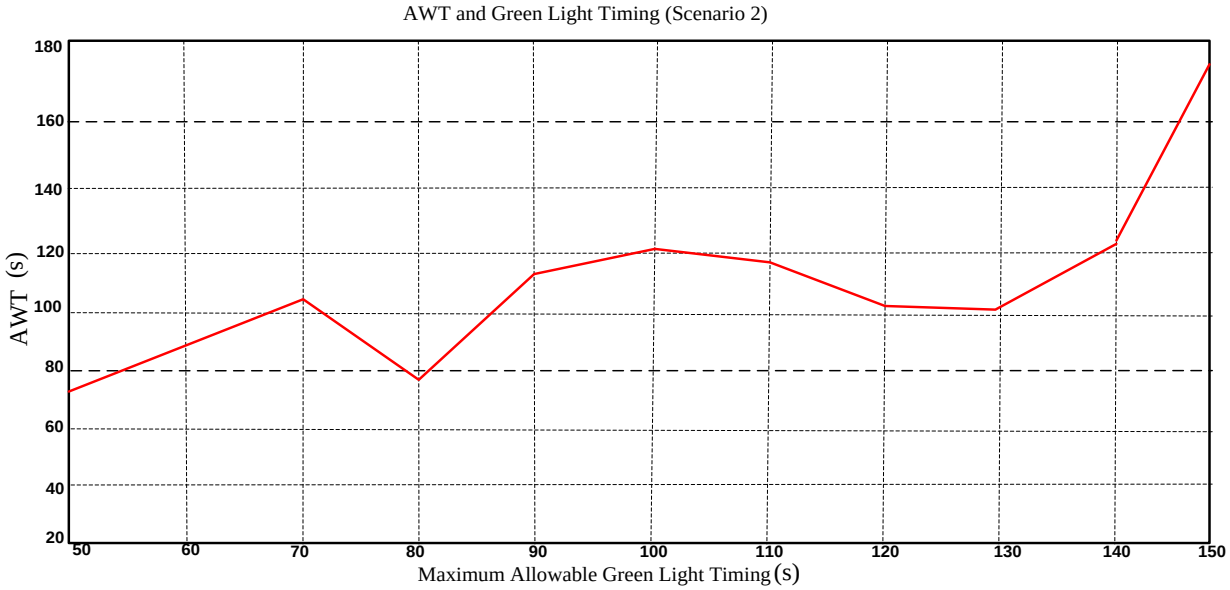


Figure: 4.3 Variation of AWT with T_{gmax} (Scenario 2)

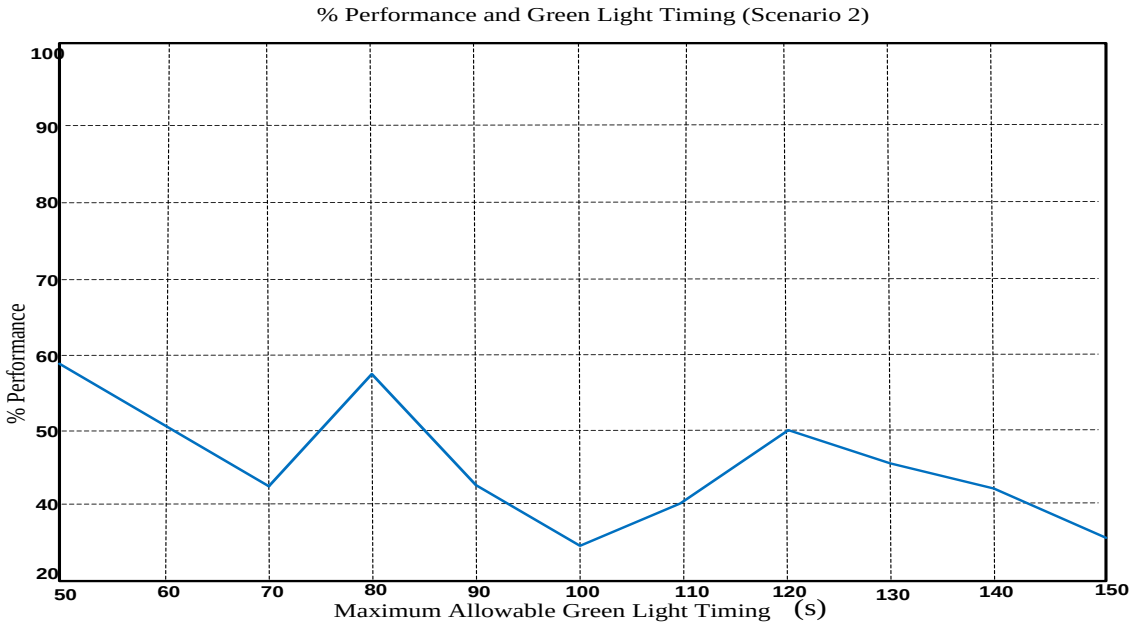


Figure: 4.4 Variation of Percentage Performance with T_{gmax} (Scenario 2)

It could be observed in Figure 4.3, that the variation of Average Waiting Time (AWT) curve is less compared to that of Figure 4.1. This may be associated with the pattern of the Average Arrival Rate (AAR). A peak value for AWT occurs at T_{gmax} equal to 150s which is the only point that deviates widely from a certain range of control points.

If this peak point omitted, the controller AWT time can be approximate to vary between 70s and 120s. The performance of the developed algorithm is relatively poor in this scenario due to high values of AWT. The maximum percentage performance in this scenario corresponds to the minimum percentage performance in scenario 1.

4.2.3 Scenario 3

This scenario is intended to generate and discuss all the features of the developed simulator. The maximum green light duration was kept at 2 minutes (120 seconds) and simulated over 30 minutes (1800 seconds). The simulation is initialized with t = 60 seconds, implying that vehicles were

allowed to queue-up for 1 minute across the entire roads. The AAR was kept at 12 vehicles per minute while the ADR was 30 vehicles per minutes, and across the entire roads respectively. Figure 4.5 to 4.9 were obtained and are further discussed as follows.

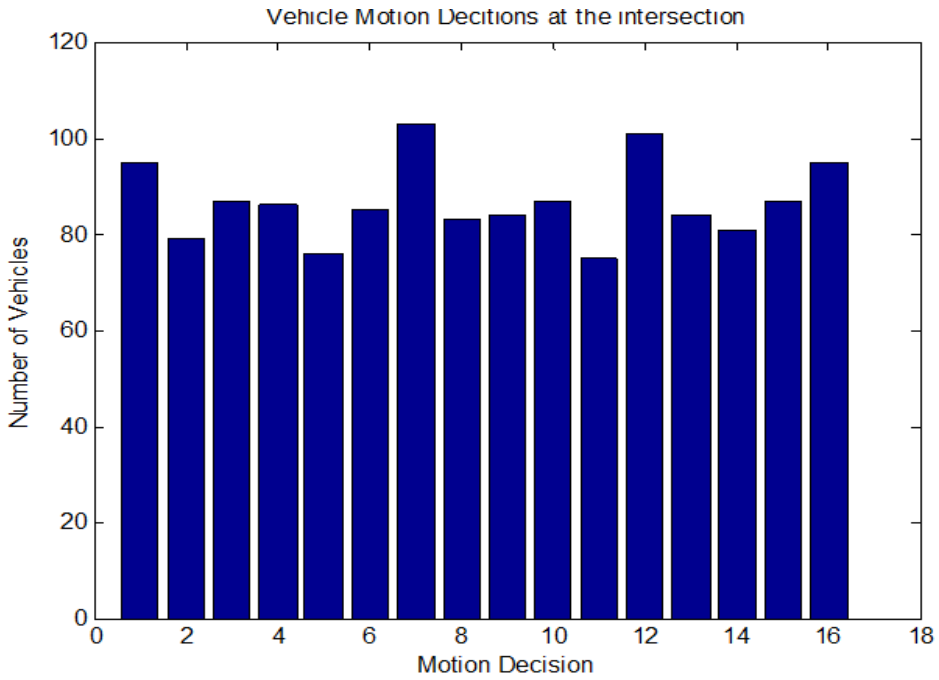


Figure 4.5: Vehicles Motion Detections at the Intersection

Figure 4.5 is a bar chart showing the number of vehicles that executes each of the 16 possible moves within the cross - road intersection. The motion decision δ of the 16 possible movements are shown in Table 4.2

Table 4.2: Motion Decision for the 16 Possible Movements

MD (Serial number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
MD Code	0111	0112	0113	0114	0211	0212	0213	0214
MD (Serial number	9	10	11	12	13	14	15	16
MD Code	0311	0312	0313	0314	0411	0412	0413	0414

It can be observed that, the highest movement undergone by the vehicles corresponds to the seventh motion decision from Figure 4.5.

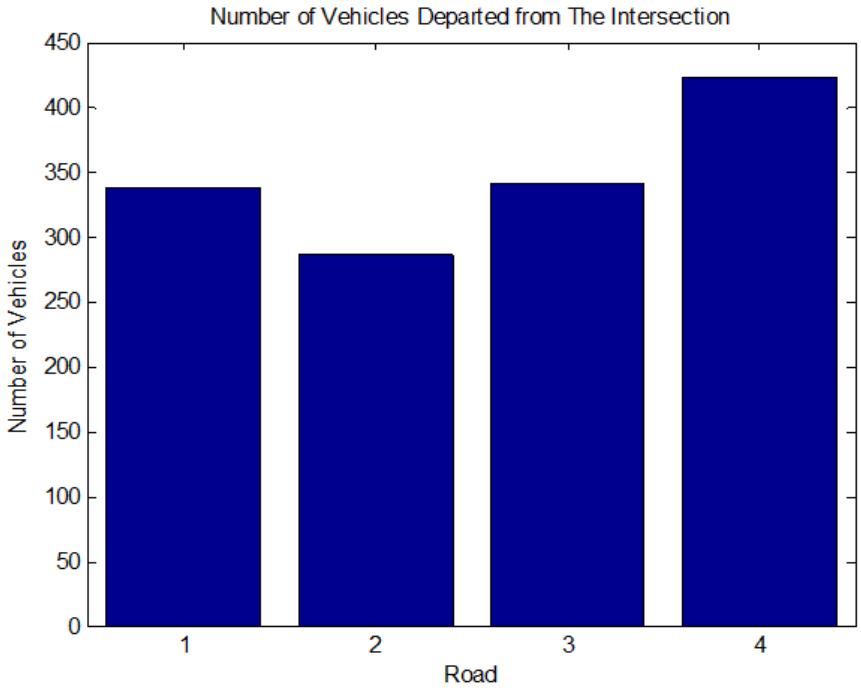


Figure 4.6: Number of Vehicles Departed at the Intersection

Figure 4.6 illustrates the number of vehicles that have departed through the cross - road of the intersection throughout the simulation period. These values are useful in evaluating the performance of the developed algorithm. Figure 4.6 also shows the road that decongested the intersection most (i.e. road 4)

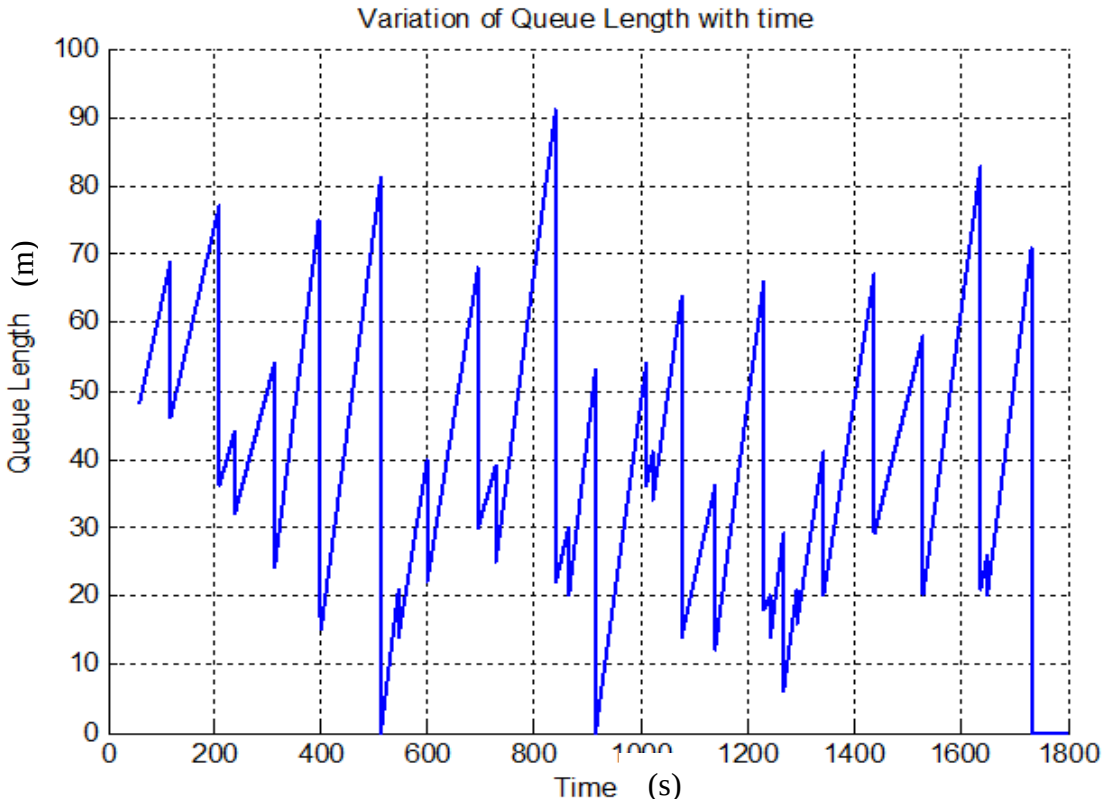


Figure 4.7: Variation of Queue Length with Time

Figure 4.7 shows the variation of queue length with simulation time. It could be observed that a zero green light duration was generated by the controller approximately 500, 900 and 1700 seconds which corresponds to a cleared condition. The ADSA is designed to generate T_{gi} of zero whenever road i is having no vehicle on it. This means that, roads with no vehicles are skipped during simulation for improvement in performance. It can also be observed that the green light duration is always less than the allowable maximum (120s). This shows that the generated t_g is only allowed to vary between 0 and t_{gmax} depending on the condition of the traffic.

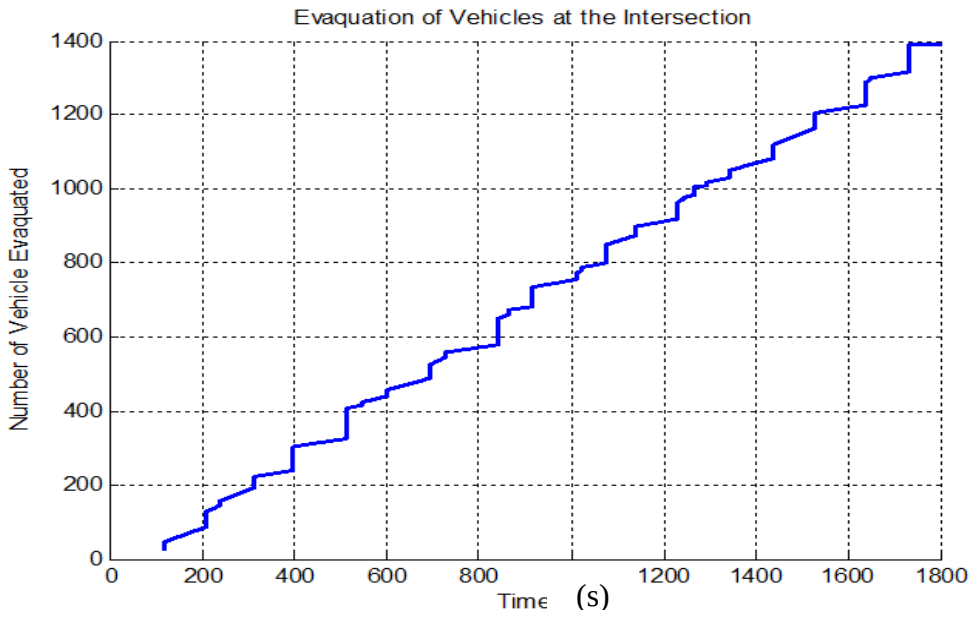


Figure 4.8: Evacuation of Vehicles At The Intersection

Figure 4.8 illustrates the cumulative sum of all the vehicles evacuation from the intersection. It can be observed that the curve in the figure has a step - like structure which results from the green light allocation over a cycle period. This step –like structures marks the beginning and ending of the cycle duration. The peak of the curve indicates that at the end of the simulation time of 30 minute, almost 1400 vehicles were cleared from the intersection.

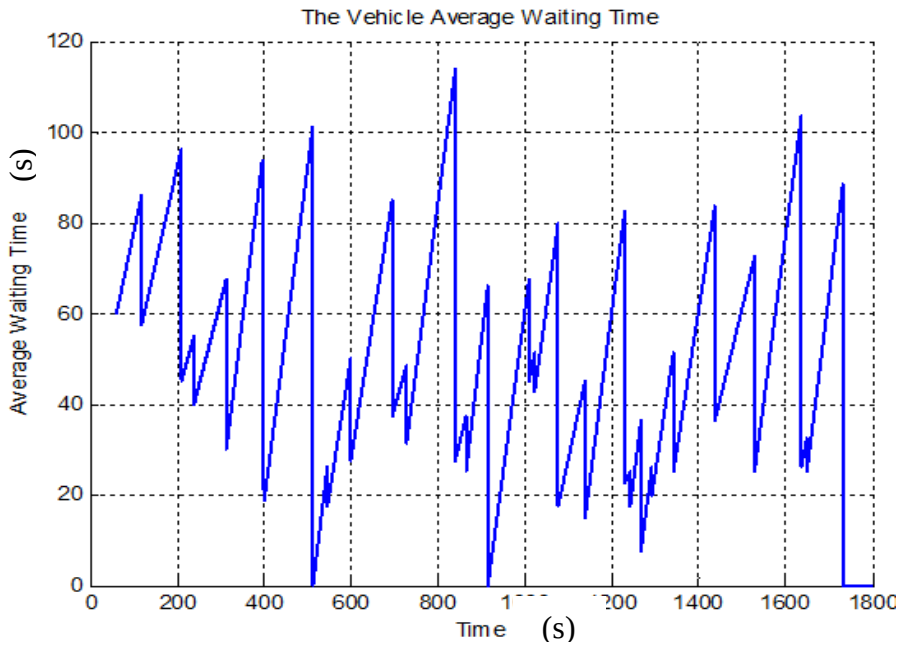


Figure 4.9: Vehicles Average Waiting Time

Figure 4.9 shows the variation of the AWT with change in time instances (simulation time). Figure 4.9 can be observed to have almost exact form as Figure 4.7. This is based on the fact that the AWT is fully controlled by the green lights scheduled for the individual roads forming the intersection. The AWT is highly sensitive to green light duration and queue length. It could be observed from Figure 4.9 that the average Waiting Time (AWT) is maintained within a safe margin of 0 and 115 seconds.

Some of the important output parameters generated by the developed simulator were recorded at the end of Scenario 3. These parameters are presented in Table 4.3.

Table 4.3: Simulation Results for Scenario 3

PARAMETER	VALUE
T_{gmax}	120s
AWT	38s
Clearing Time	1640s
Simulation Duration	1800s
Total Departure after 1800s	1382 vehicles
Total Queue Length at t = 1800s	22 vehicles.
Performance	98.43%
Start Time	60s

4.3 Validation

In order to demonstrate the validity of the developed ADSA for vehicular traffic control, the scenario simulated in the work of Erwan *et.al,* 2015 was replicated using the developed simulator. The results of AWT and Traffic Queue were generated and plotted against those obtained in the work of Erwan *et.al,* 2015. Figure 4.10 and 4.11 shows a comparison of AWT and Traffic Queue respectively.

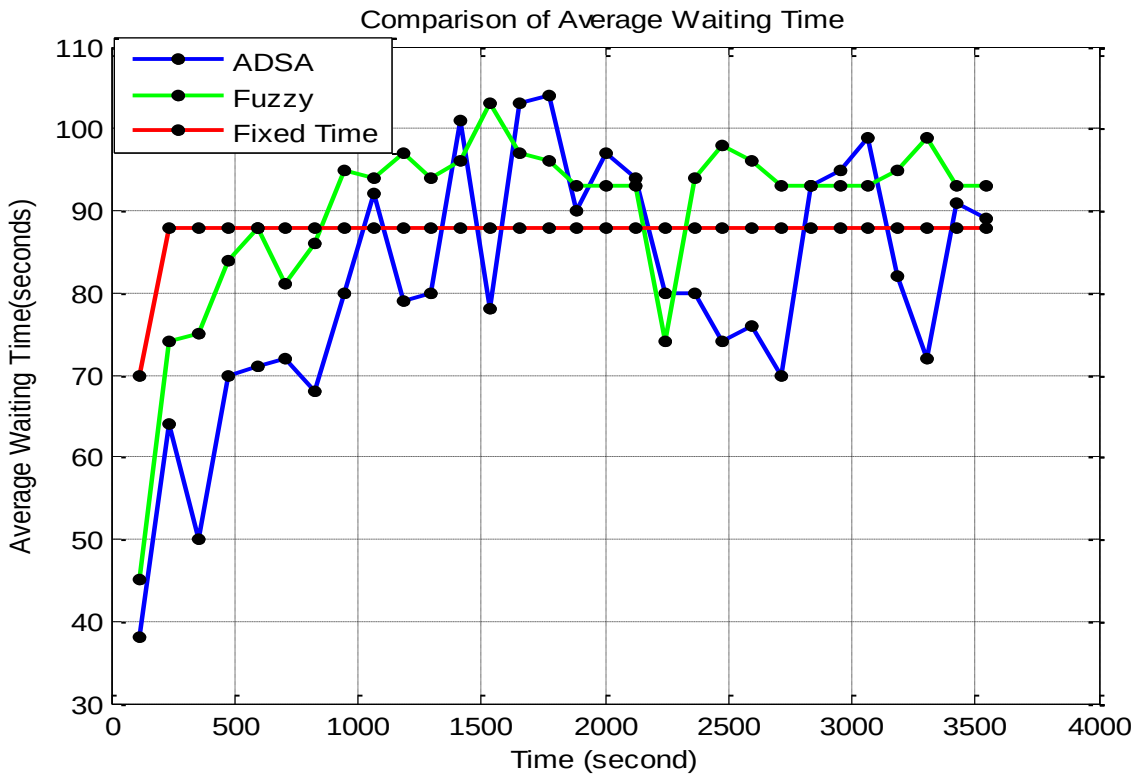


Figure 4.10: Comparison of AWT for Fuzzy, Fixed-Time, and ADSA Based Traffic Controllers

The conventional Fixed Time traffic light controllers are characterized by their constant AWT regardless of traffic condition. This can easily be observed from the 'red' colour shown in Figure 4.10. The Fuzzy logic approach presented in the work of Erwan *et.al*, 2015, yielded an AWT represented by the 'green' colour shown in Figure 4.10. On the other hand, the developed ABC based Adaptive Dynamic Scheduling Algorithm (ADSA) yielded the 'blue' colour. The wide variation observed in the AWT of the ADSA is due the large dependency of the developed ADSA approach to traffic condition. The scenario simulated made use of thirty points (30) on the curve. Out of the 30 scenarios for AWT, the developed ADSA performs better than fuzzy on 23 points (76.67%), while fuzzy performs better on 5 points (16.67%). On the remaining 2 points (6.67%) their results are similar. The overall performance show that the developed ADSA is better than the fuzzy-based approach.

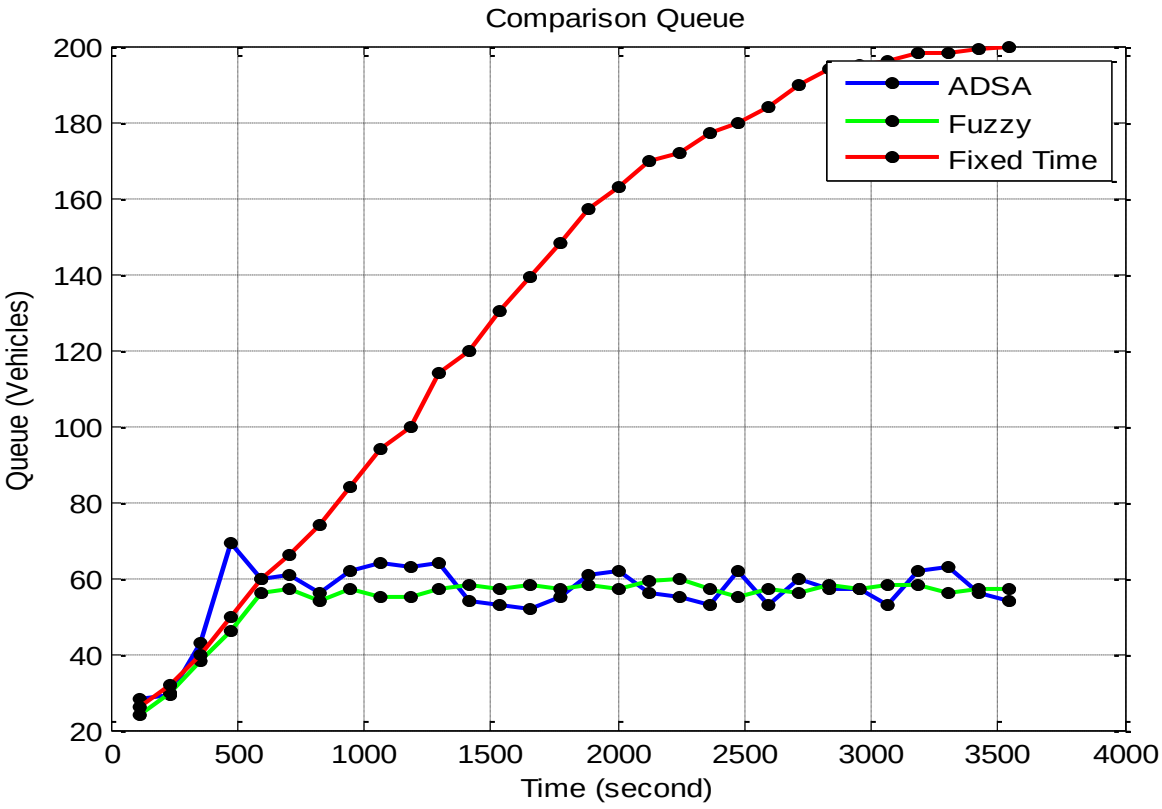


Figure 4.11: Comparison of Queue for Fuzzy, Fixed-Time, and ADSA Based Traffic Controllers

The continuous increase in queue length common to Fixed-Time controllers (as represented by the 'red' colour shown in Figure 4.11) is usually mitigated by using variable-time controllers. From Figure 4.11, it could be observed that both the ADSA and fuzzy based controllers have a relatively fixed queue length as compared to the Fixed-Time controllers. This may be associated with the proportionate green light time scheduled by the variable-time controllers. The scenario simulated made use of thirty points (30) on the curve. Out of the 30 scenarios for traffic queues, the developed ADSA performs better than fuzzy on 16 points (53.33%), while fuzzy performs better on 7 points (23.33%). On the remaining 7 points (23.33%) their results are similar. The overall performance show that the developed ADSA is better than the fuzzy-based approach.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

In this work, the conventional ABC algorithm has been improved in order to suit application in vehicular traffic control (VTC). An ABC based Adaptive Dynamic Scheduling Algorithm (ADSA) for VTC have been developed. A mathematical model and control strategy for a cross -road intersection has been developed to suit application in traffic green light controller design. A VTC simulator was developed using MATLAB GUI development environment. In order to further demonstrate the validity and the effectiveness of the developed simulator, a comparison between the two approaches (Fixed-Time and fuzzy - based approach) presented in the work of (Erwan *et.al*, 2015) and the developed ADSA showed that the developed ADSA outperformed the fuzzy – based approach with 76.67% performance for AWT and 53.33% for queues. The results clearly expressed that the developed ADSA method has been successful to outperform the fuzzy-based approach.

5.2 Significant Contribution

This research work offers the following significant contributions to the existing body of knowledge:

1. Development of an Artificial Bee Colony (ABC) based Adaptive Dynamic Scheduling Algorithm (ADSA) that schedules appropriate traffic control signal (green light timing) duration based on the condition of the vehicular traffics at an intersection.
2. Development of a MATLAB based Graphics User Interface (GUI) that could be used in Vehicular Traffic Control System (VTCS) analysis.

3. Development of an adaptive model for road intersections and a suitable VTCS coding for traffic signal representation and motion control, and finally carrying out a comparative study between the developed approach, the conventional Fixed-Time based approach, and the Fuzzy logic based approach presented in the work of (Erwan *et.al*, 2015). The results clearly showed that the developed ADSA method has been successful to outperform the fuzzy-based approach with a 60% (76.67, 16.67%) for Average Waiting Time AWT and 30% (53.33, 23.33%) for the traffic queues as performances.

5.3 Limitations

The following limitations were encountered in the course of this work:

1. The developed control strategy was based on standard roads and therefore could not be practically implemented on any of the nearby local roads.
2. The breakdown of vehicles on the lanes were not considered.
3. The type of vehicular traffic controllers used in this research has not yet being installed at an intersection across Nigeria. Therefore, practical data could not be acquired for comparison.

5.4 Recommendations for Further Work

The following are some useful recommendations that may serve as guide to the advancement of this research work:

1. The developed ADSA approach can be implemented on microchips and integrated into a micro scale traffic controller designed and coded using the developed technique to aid a more practical scenario implementation.

2. A hybrid technique based on ADSA and Fuzzy could be developed while incorporating more realistic assumptions and built into java based GUI environment to guarantee robustness and high sensitivity to changes in traffic condition.
3. The developed vehicular traffic control simulator should be replicated with java, while incorporating traffic motion controllers and compiled into real software package that can run on various PCs in order to aid students in simulating any desired traffic control scenario.

REFERENCES

Alam, J. & Pandey, M. (2014). Advance Traffic Light System based on Congestion Estimation using Fuzzy Logic. *International Journal of Emerging Technology and Advanced Engineering. (IJETAE)*, 5(1), 38-43

Aljaafreh, A. & Oudat, N. (2014).Optimized Timing Parameters for Real-time Adaptive Traffic signal Controller. *16th International Conference on Computer Modelling and Simulation*, 243-247.

Anokye, M., Abdul-Aziz, A., Annin, K., & Oduro, F. T. (2013). Application of Queuing Theory to Vehicular Traffic at Signalized Intersection in Kumasi-Ashanti Region, Ghana. *American International Journal of Contemporary Research, 3*(7).

Arrobo, G. (2012).Improving the throughput and reliability of wireless sensor networks with application to wireless body area networks (Unpublished doctoral dissertation). P2. Retrievedfromhttp://scholarcommons.usf.edu/cgi/viewcontent.cgi?article=5475&context=etd

Collotta, M., Bello, L. L., & Pau, G. (2015). A novel approach for dynamic traffic lights management based on Wireless Sensor Networks and multiple fuzzy logic controllers. *Expert Systems with Applications*, *42*(13), 5403-5415.

Collata M., Guiffre, T., Pau, G., & Scata, C. (2014). Smart Traffic Light Junction Management Using Wireless Sensor Networks. *WSEAS Transactions on Communications*, 13(1), 648-658

Cosariu, C., Prodan, L., & Vladutiu, M. (2013). *Toward traffic movement optimization using adaptive inter-traffic signaling.* Paper presented at the Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium on.

Ding, D. (2011). Modeling and simulation of highway traffic using a cellular automaton approach.

Erwan, P. Wahyungorro, O. & Sulistyo, S. (2015). *Design and Simulation of Adaptive Traffic Light Controller using Fuzzy Logic Sugeno Method.* International Journal of Scientific Research Publications, 5(4), 1-6

Ezell, S. (2010). Explainig international IT application leadership: Intelligent Transportation Systems. *The Information Technology and Innoative Foundation (ITIF),* 8.

Faye, S., Chaudet, C., & Demeure, I. (2012). *A distributed algorithm for adaptive traffic lights control.* Paper presented at the Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on.

Faye, S., Chaudet, C., & Demeure, I. (2013). *Multiple intersections adaptive traffic lights control using Wireless Sensor Networks.* Institute Mines - Telecom, Paris Tech,CNRS LTCI UMR 5141, Paris, France, 2(2), 1-20.

Gundogan, F., Karagoz Z., Kocyigit , N., Karadog, A., Coylan, H., Murat,Y. (2014).An Evaluation of Adaptive Traffic Control System in Istanbul, Turkey. *Journal of Traffic and Logistics Engineering,* 2(3),

Holland, J.H. (1973). Genetic algorithms and the optimal allocation trials, *SIAM J. computing,* 2(2), 88-105

Karaboga, D., & Akay, B. (2009). A comparative study of artificial bee colony algorithm. *Applied Mathematics and Computation, 214*(1), 108-132.

Karaboga, D., & Basturk, B. (2007). A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. *Journal of global optimization, 39*(3), 459-471

Karaboga, D., Gorkemli, B., Ozturk, C., & Karaboga, N. (2014). A comprehensive survey: artificial bee colony (ABC) algorithm and applications. *Artificial Intelligence Review, 42*(1), 21-57

Karpis, O. (2013). Wireless sensor networks in intelligent transportation systems. *International Journal of Modern Engineering Research (IJMER), 3*, 611-617.

Kotwal, A. R., Lee, S. J., & Kim, Y. J. (2013). Traffic Signal Systems: A Review of Current Technology in the United States. *Science and Technology, 3*(1), 33-41.

Kumar, K., & Kumar, R. (2012). A Textbook on Neural Networks and fuzzy logic, ISBN 978-93-80027-78-4, pp. 5

Ma'en Saleh, A. A., & Al-Oudat, N. (2014). Hierarchal Scheduling Algorithm for Congestion Traffic Control Using Multi-Agent Systems. *International Journal for Advanced Computer Research*, 4(17), 915-921

Mckenney, D. (2011). Distributed and adaptive traffic light control within a realist traffic formulation, thesis submitted for Masters in Computer Science Carleton University, Ottawa Ontario, 5-30

Mohan, A., & Rani, J. L.(2013). An Intelligent Algorithm for Traffic Signal Scheduling.*International Journal of Chemical, Environmental & Biological Sciences , 1(1),* 119-124

Odeh, S. M. (2013). Management of an intelligent traffic light system by using genetic algorithm.

Osigwe, C., Oladipo F., Onibere, E. (2011). Design and simulation of Intelligent Traffic Control System. *International Journal of Advances in Engineering & Technology (IJAET)*, 1(5), 46-57

Punetha, D., Jha, P., & Singh, S. (2014). An Intelligent Transportation System with Computer System Architecture Approaching Safe Passage in Adaptive Traffic Light. *International Journal of Computer Networks and Wireless Communications (IJCNWC)*, *4*(2), 143-149.

Salehi, M., Sepahvand, I. & Yarahmad, M. (2014). A Traffic Lights Control System Based on Fuzzy Logic,*International Journal of u- and e- Service, Science and Technology*, *7(3)*, 27-34.

Samadi S.,Rad, A., Kazemi, F., & Jafarian, H. (2012).Performance Evaluation of intelligent Adaptive Traffic Light Control System: A Case Study. *Journal of Transportation Technologies*,*2*(03), 248-259.

Shanmugasundaram, S. & Banumathi, P. (2015). A Study on Single Server Queue in Southern Railway using Monte Carlo Simulation, *Global Journal of Pure and Applied Mathematics (GPAM)*, 11(1), 4-10

Sinhmar, P. (2012). Intelligent traffic light and density control using IR sensors and microcontroller. *International Journal of Advanced Technology & Engineering Research*, *2*(2), 30-35.

Singh,Y., Mittal, P.,(2013).Analysis and Designing of the proposed Intelligent Road Traffic Congestion Control System with Image Mosiacking Technique,*International Journal of IT, Engineering and Applied Sciences Research* (IJIEASR),2(4), 27-32.

Singh, L., Tripathi, S., & Arora, H. (2009). Time optimization for traffic signal control using genetic algorithm. *International Journal of Recent Trends in Engineering*, *2*(2), 88-96.

Singh, Y.P. (2014). A Novel Approach for Automatic Control of Road Traffic Congestion using Image Mosiacking Technique. *International Journal of Emerging Technology and Advanced Engineering. (IJETAE),* 4(1), 382-389.

Stavrouslakis P. (Ed.) (2004). Neuro- fuzzy and fuzzy neural applications in telecommunications Springer, *signals and communications technology*, ISBN 3- 540-40759-6, 5-15

Subramaniam, S., Esro, M., & Aw, F. (2012). Self-Algorithm Traffic Light Controllers for Heavily Congested Urban Route. *WSEAS Transactions on Circuits and Systems, 11*(4), 115-124.

Sundarapandian, V. (2009). Queuing Theory: Probability, Statistics and Queuing Theory. PHI Learning. ISBN 8120338448, 23

Tanner, M. (1995). *Practical queueing analysis*: McGraw-Hill Companies.

Tubaishat, M., Shang, Y., & Shi, H. (2007). *Adaptive traffic light control with wireless sensor networks.* Paper presented at the Proceedings of IEEE Consumer Communications and Networking Conference.

Wang, F., Ye, C., Zhang, Y., & Li, Y. (2014). Simulation Analysis and Improvement of the Vehicle Queuing System on Intersections Based on MATLAB. *Open Cybernetics & Systemics Journal, 8*, 217-223

www.marsh.com retreived on 18th 09,2015

www.intelligenttechniquesinmechatroics.comretreived on 18th 09,2015

Yousef, K. M., Al-Karaki, M. N., & Shatnawi, A. M. (2010). Intelligent Traffic Light Flow Control System Using Wireless Sensors Networks. *J. Inf. Sci. Eng., 26*(3), 753-768.

Zadeh, L.A, (1973)."Outline of a new approach to the analysis of complex systems and decision processes". *IEEE Trans. Systems, Man and Cybernatics* SMC-3, 28-44

Zhou, B., Cao, J., Zeng, X., & Wu, H. (2010). *Adaptive traffic light control in wireless sensor network-based intelligent transportation system.* Paper presented at the Vehicular Technology Conference Fall (VTC 2010-Fall), 2010 IEEE 72nd.

APPENDIX A

Sub-M-Files for Artificial Bees Algorithm

(a) Employed Bees

```
function [ new_pop ] = employed_bee( pop )
%INITIALIZE Summary of this function goes here
%   Detailed explanation goes here
[I,J]=size(pop);
new1_pop=pop;
new2_pop=pop;
fori=1:length(I)
for j=1:length(J)
if i+1<=I
      new1_pop(i,j)=pop(i,j)+(rand*(-1)^randi(I+J))*(pop(i,j)-pop(i+1,j));
elseif i-1>=1
         new1_pop(i,j)=pop(i,j)+(rand*(-1)^randi(I+J))*(pop(i,j)-pop(i-1,j));
end
if j+1<=J
      new2_pop(i,j)=pop(i,j)+(rand*(-1)^randi(I+J))*(pop(i,j)-pop(i,j+1));
elseif j-1>=J
         new2_pop(i,j)=pop(i,j)+(rand*(-1)^randi(I+J))*(pop(i,j)-pop(i,j-1));
end
new_pop=(new1_pop+new2_pop)/1;
end
```

```
end

end
```

b) Initialization

```
functionnew_pop = initialize( pop )

%INITIALIZE Summary of this function goes here

%   Detailed explanation goes here

[I,J]=size(pop);

new_pop=pop;

fori=1:length(I)

for j=1:length(J)

new_pop(i,j)=min(pop(i,:))+rand*(max(pop(i,:))-min(pop(i,:)));

end

end

end
```

C) Onlooker Bees

```
Function[new_pop ] =onlooker_bee( fit_best,pop,Queue_length,average_arate,average_drate)

%ONLOOKER_BEE Summary of this function goes here

%   Detailed explanation goes here

[I,~]=size(pop);

fit=objt_abc(pop,Queue_length,average_arate,average_drate);

fori=1:length(fit_best)
```

```
iffit_best(i)==0
fit_best(i)=1;
end
end
Probs=(0.9*fit/fit_best)+0.1;
Probs=ceil(Probs)-Probs;
popi=ceil(Probs*I);
new_pop=pop;
fori=1:length(popi)
ifpopi(i)==0
popi(i)=randi(I);
end
new_pop(i,:)=pop(popi(i),:);
end
[ new_pop ] = employed_bee(new_pop );
end
```

D) Scout Bees

```
function [ new_pop ] =onlooker_bee(fit_best,pop,Queue_length,average_arate,average_drate)
%ONLOOKER_BEE Summary of this function goes here
%   Detailed explanation goes here
[I,~]=size(pop);
fit=objt_abc(pop,Queue_length,average_arate,average_drate);
```

```
fori=1:length(fit_best)
iffit_best(i)==0
fit_best(i)=1;
end
end
Probs=(0.9*fit/fit_best)+0.1;
Probs=ceil(Probs)-Probs;
popi=ceil(Probs*I);
new_pop=pop;
fori=1:length(popi)
ifpopi(i)==0
popi(i)=randi(I);
end
new_pop(i,:)=pop(popi(i),:);
end
[ new_pop ] = employed_bee(new_pop );
end
```

APPENDIX B

Main M-Files ABC Algorithm

```
function [ fit_best ]=ABC( popsize,npar,ntry_max,plot_choice,...
Queue_length,average_arate,average_drate )
% popsize=10;
% npar=10;
% ntry_max=10;
% plot_choice=1;
% Queue_length=100000;
% average_arate=10;
% average_drate=5;
%ABC Summary of this function goes here
%   Detailed explanation goes here
pop=rand(popsize,npar);
pop = initialize( pop );
fit=objt_abc(pop,Queue_length,average_arate,average_drate);
[fit_best]=min(fit);
Best=fit_best;
ntry=0;
whilentry<ntry_max-1
ntry=ntry+1;
[ new_pop ] = employed_bee( pop );
   fit_new=objt_abc(new_pop,Queue_length,average_arate,average_drate);
```

```
fori=1:length(fit_new)
iffit_new(i)<fit(i)
pop(i,:)=new_pop(i,:);
fit(i,1)=fit_new(i,1);
end
end
   [fit_bestTg]=min(fit);
   Best=[Best,fit_best];
ew_pop ] = onlooker_bee(fit_best,pop,Queue_length,average_arate,average_drate );
   fit_new=objt_abc(new_pop,Queue_length,average_arate,average_drate);
fori=1:length(fit_new)
iffit_new(i)<fit(i)
pop(i,:)=new_pop(i,:);
fit(i,1)=fit_new(i,1);
end
end
   [fit_bestTg]=min(fit);
   Best=[Best,fit_best];
[ new_pop ] = scout_bee( pop);
   fit_new=objt_abc(new_pop,Queue_length,average_arate,average_drate);
fori=1:length(fit_new)
iffit_new(i)<fit(i)
pop(i,:)=new_pop(i,:);
```

```
fit(i,1)=fit_new(i,1);
end
end
    [fit_bestTg]=min(fit);
    Best=[Best,fit_best];
end
ifplot_choice==1
figure
plot(1:length(Best),Best,'-r','linewidth',2)
title('ABC Optimization Curve')
xlabel('Generation')
ylabel('cost')
end
end
```

APPENDIX C

Objective Function for Traffic

```
function [fit_trafficTg]=objf_traffic(pop,Queue_length,average_arate,average_drate)
%OBJ_TRAFFIC Summary of this function goes here
%   Detailed explanation goes here
U=average_drate;
N=Queue_length;
L=average_arate;
X=sum(pop,2);
```

```
fit_traffic=(L*U-N*L-U*N*X)./(N*L+U*N*X);

Tg=N*(1-X)/U;

end
```

APPENDIX D

Objective Function for ABC

```
function [fit Tg]=objt_abc(pop,Queue_length,average_arate,average_drate)

%OBJT_ABC Summary of this function goes here

%   Detailed explanation goes here

[fit_trafficTg]=objf_traffic(pop,Queue_length,average_arate,average_drate);

fit=fit_traffic;

forpopi=1:length(fit_traffic)

iffit_traffic(popi)>=0

fit(popi,1)=1/(1+fit_traffic(popi));

else

fit(popi,1)=1+ abs(fit_traffic(popi));

end

end

fit=Queue_length/average_drate-fit;

end
```

APPENDIX E

The Traffic Controller M-Files (Proposed ADSA)

```
function [ traffic_info,Clearing_time,Time,AWT,VEHICLES,NO_VEHICLES,...
no_vehicles EVAQ ] = traffic_controller( ...
no_roads,~, ...
average_arate,average_drate,...
                    ~,sim_duration,To,Tg_max...
,popsize,npar,ntry_max)
% no_roads=4;
trafficL_config=[4 3];
traffic_moves=motion(no_roads,trafficL_config);
% average_arate=5*rand(1,no_roads);
% sim_duration=200;
% To=0;
% Tg_max=20;
G_light_factor=1;
% average_drate=average_arate*100;
% popsize=10;
% npar=5;
% ntry_max=10;
%The traffic information consists of the following;
%A total of no_roads+1 cells with each cell containing four inner cells
```

```
%the first no_roads cells represents: the traffic information of the
%individual roads.
%The remaining one cell represents set of cycle informations;
%A road traffic information is described as follows:
% 1  No_vehicles available in the ith road at time 't';
% 2  No_vehiclesevaquated from road i at the jth green light time
% 3  thejth green ligth time;
% 4  thejth average waiting time

% The Cycle information are as follows:
% 1  the cycle time;
% 2  the cycle number
% 3  The average green ligth time
% 4  TheNo_vehiclesevaquated;
clc
Time=[];
Clearing_time=[];
if To==0
   To=max(1./average_arate);
end
time=To;
traffic_info=cell(1,4,no_roads+1);
[ Vehiclesno_vehicles ] = generate_vehicles(no_roads,traffic_moves, ...
```

```
average_arate,To, 0 ); % Generate vehicles
for road=1:no_roads;
traffic_info{1,1,road}=no_vehicles(road);
traffic_info{1,4,road}=no_vehicles(road)/average_arate(road);
end
k=0;
EVAQ=zeros(1,no_roads);
Time=[Time,time];
[ VEHICLES NO_VEHICLES ] = generate_vehicles(no_roads,traffic_moves, ...
                    average_arate,0, 0 );


while time<sim_duration
for cycle = 1
Tcycl=0;
      k=k+1;
Cycle_evaq=0;
G_allocated=1;
Tg=feval('G_light',no_vehicles(1),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
iftime+Tg<=sim_duration
time=time+Tg;
else
time=sim_duration;
```

```
Tg=sim_duration-time;
end
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(2)=EVAQ(2)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
    Time=[Time,time]; G_allocated=2;
```

```
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(1)=EVAQ(1)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time];
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end
if cars==0
```

```
Clearing_time=time;
break
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%~~~  [3  4]  and  [4  3]
G_allocated=3;

Tg=feval('G_light',no_vehicles(3),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
iftime+Tg<=sim_duration
time=time+Tg;
else
time=sim_duration;
Tg=sim_duration-time;
end
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(4)=EVAQ(4)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
```

```
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=4;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(3)=EVAQ(3)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
```

```
        traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
    Time=[Time,time];
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end
if cars==0
Clearing_time=time;
break
end
rd=no_roads+1;
traffic_info{1,1,rd}=[traffic_info{1,1,rd},Tcycl];
traffic_info{1,2,rd}=[traffic_info{1,2,rd},k];
traffic_info{1,3,rd}=[traffic_info{1,3,rd},Tcycl/no_roads];
traffic_info{1,4,rd}=[traffic_info{1,4,rd},Cycle_evaq];
end
for cycle = 2
Tcycl=0;
```

```
        k=k+1;
Cycle_evaq=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%~~[1 3] , [3 1] and
[4 1]
G_allocated=1;

Tg=feval('G_light',no_vehicles(1),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
iftime+Tg<=sim_duration
time=time+Tg;
else
time=sim_duration;
Tg=sim_duration-time;
end
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(3)=EVAQ(3)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
```

```
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=3;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(1)=EVAQ(1)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
```

```
            traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=4;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(1)=EVAQ(1)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
          traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
          traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
           traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
           traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
```

```
end
      Time=[Time,time];
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end

if cars==0
Clearing_time=time;
break
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%~~[2 4] , [3 1] and
[4 2]
G_allocated=2;

Tg=feval('G_light',no_vehicles(2),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
iftime+Tg<=sim_duration
time=time+Tg;
else
time=sim_duration;
Tg=sim_duration-time;
end
```

```
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(4)=EVAQ(4)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
    Time=[Time,time]; G_allocated=3;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
```

```
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(1)=EVAQ(1)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=4;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(2)=EVAQ(2)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
```

```
            traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
            traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
             traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
             traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
        Time=[Time,time];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end
if cars==0
Clearing_time=time;
break
end
rd=no_roads+1;
traffic_info{1,1,rd}=[traffic_info{1,1,rd},Tcycl];
```

```
traffic_info{1,2,rd}=[traffic_info{1,2,rd},k];
traffic_info{1,3,rd}=[traffic_info{1,3,rd},Tcycl/no_roads];
traffic_info{1,4,rd}=[traffic_info{1,4,rd},Cycle_evaq];
end
for cycle = 3
Tcycl=0;
        k=k+1;
Cycle_evaq=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%~~[3 2] , [2 3] and
[1 4] G_allocated=3;

Tg=feval('G_light',no_vehicles(3),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
iftime+Tg<=sim_duration
time=time+Tg;
else
time=sim_duration;
Tg=sim_duration-time;
end
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
```

```
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(2)=EVAQ(2)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=2;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(3)=EVAQ(3)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
```

```
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=1;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(4)=EVAQ(4)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
```

```
        traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
    Time=[Time,time];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end
if cars==0
Clearing_time=time;
break
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%~~[4 1] , [1 4] and
[2 3]
G_allocated=4;

Tg=feval('G_light',no_vehicles(4),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
```

```
iftime+Tg<=sim_duration
time=time+Tg;
else
time=sim_duration;
Tg=sim_duration-time;
end
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(1)=EVAQ(1)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
```

```
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=1;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(4)=EVAQ(4)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
          traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
          traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
           traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
           traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=2;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
```

```
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(3)=EVAQ(3)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time];
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end
if cars==0
Clearing_time=time;
```

```
break
end
rd=no_roads+1;
traffic_info{1,1,rd}=[traffic_info{1,1,rd},Tcycl];
traffic_info{1,2,rd}=[traffic_info{1,2,rd},k];
traffic_info{1,3,rd}=[traffic_info{1,3,rd},Tcycl/no_roads];
traffic_info{1,4,rd}=[traffic_info{1,4,rd},Cycle_evaq];
end
for cycle =4
Tcycl=0;
    k=k+1;
Cycle_evaq=0;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%~~[1 4] , [2 3] [3
1]                    and                    [4                    2]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
G_allocated=1;

Tg=feval('G_light',no_vehicles(1),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
iftime+Tg<=sim_duration
time=time+Tg;
else
```

```
time=sim_duration;
Tg=sim_duration-time;
end
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(4)=EVAQ(4)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
```

```
        Time=[Time,time]; G_allocated=2;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(3)=EVAQ(3)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
            traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
            traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
             traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
             traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
        Time=[Time,time]; G_allocated=3;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(1)=EVAQ(1)+vevaq;
```

```
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=4;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(2)=EVAQ(2)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
```

```
if road==G_allocated
            traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
            traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
       Time=[Time,time];
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end
if cars==0
Clearing_time=time;
break
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%~~[1 1] , [2 2] [3
3]                    and                    [4                    4]
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%
```

```
G_allocated=1;


Tg=feval('G_light',no_vehicles(1),Tg_max,G_light_factor,popsize,npar,ntry_max,average_arate,
average_drate,no_vehicles );
iftime+Tg<=sim_duration
time=time+Tg;
else
time=sim_duration;
Tg=sim_duration-time;
end
Tcycl=Tcycl+Tg;
[ Vehicles,no_vehicles ] = deploy_cars(Vehicles,no_vehicles,no_roads,traffic_moves, ...
average_arate,Tg, time );
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(1)=EVAQ(1)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
```

```
            traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
            traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
      Time=[Time,time]; G_allocated=2;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(2)=EVAQ(2)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
          traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
          traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
            traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
            traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
```

```
end
end
    Time=[Time,time]; G_allocated=3;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(3)=EVAQ(3)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
    Time=[Time,time]; G_allocated=4;
[ vevaq, VEHICLES,NO_VEHICLES,Vehicles ] = evaquate_cars(...
Vehicles ,Tg,average_drate,...
```

```
G_allocated,VEHICLES,NO_VEHICLES );
EVAQ(4)=EVAQ(4)+vevaq;
Cycle_evaq=Cycle_evaq+vevaq;
no_vehicles(G_allocated)=no_vehicles(G_allocated)-vevaq;
for road=1:no_roads
        traffic_info{1,1,road}=[traffic_info{1,1,road},no_vehicles(road)];
        traffic_info{1,4,road}=[traffic_info{1,4,road},no_vehicles(road)/average_arate(road)];
if road==G_allocated
         traffic_info{1,3,G_allocated}=[traffic_info{1,3,G_allocated},Tg];
         traffic_info{1,2,G_allocated}=[traffic_info{1,2,G_allocated},vevaq];
else
traffic_info{1,3,road}=[traffic_info{1,3,road},0];
traffic_info{1,2,road}=[traffic_info{1,2,road},0];
end
end
     Time=[Time,time];
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
cars=0;
for K=1:no_roads
cars=cars+traffic_info{1,1,K}(end);
end
if cars==0
```

```
Clearing_time=time;
break
end
rd=no_roads+1;
traffic_info{1,1,rd}=[traffic_info{1,1,rd},Tcycl];
traffic_info{1,2,rd}=[traffic_info{1,2,rd},k];
traffic_info{1,3,rd}=[traffic_info{1,3,rd},Tcycl/no_roads];
traffic_info{1,4,rd}=[traffic_info{1,4,rd},Cycle_evaq];
end


end

Plots=[];
T=traffic_info;
for road=1:no_roads
ifisempty(Plots)~=1
while length(T{1,4,road})<length(Plots(end,:))
T{1,4,road}=[T{1,4,road},0];
end
while length(T{1,4,road})>length(Plots(end,:))
Plots(end,end+1)=0;
end
end
```

```
    Plots=[Plots;T{1,4,road}];
end
    AWT=sum(sum(Plots))/(no_roads*length(sum(Plots)));


end
```