Computational Intelligence-based Traffic Signal Timing Optimization

A thesis submitted in fulfillment of the requirements for the degree of
Doctor of Philosophy

Submitted by:

Sahar Araghi
MSc and BSc(Hons) Computer Science

Under supervision of:

– Douglas Creighton
– Abbas Khosravi
– Michael Johnston

Deakin University
Centre for Intelligent Systems Research
May 2015
I am the author of the thesis entitled: “Computational Intelligence-based Traffic Signal Timing Optimization” that is submitted for the degree of: “Doctor of Philosophy in Computer Science”

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

Full Name: Sahar Araghi
Signed: [Signature]
Date: 14/04/2015
I certify that the thesis entitled: “Computational Intelligence-based Traffic Signal Timing Optimization” that is submitted for the degree of: “Doctor of Philosophy in Computer Science” is the result of my own work and that where reference is made to the work of others, due acknowledgment is given. The works are not in any way a violation or infringement of any copyright, trademark, patent, or other rights whatsoever of any person. If the works have been commissioned, sponsored or supported by any organisation, I have fulfilled all of the obligations required by such contract or agreement.

I also certify that any material in the thesis which has been accepted for a degree by any other university or institution is identified in the text.

Full Name: Sahar Araghi
Signed: [Signature]
Date: 14/04/2015
Abstract

Managing road traffic congestion is one of the major challenges in today’s modern cities. Traffic congestion has explicit effects on productivity and efficiency, as well as side effects on environmental sustainability and health. Among different approaches to handle increasing traffic volumes, controlling traffic flows at intersections is recognized as a beneficial technique, to decrease daily travel times.

The objective of traffic signals is to increase road capacity and decrease delays, while ensuring safe travel, at busy intersections. Methods to optimize traffic signal timing, which are currently deployed to manage intersections, rely on mathematical models that do not sufficiently capture the dynamics of traffic at an intersection. Consequently, conventional traffic signal timing methods have achieved some level of success, but have not shown as much as possible success in optimizing vehicles travel time.

Previous research investigating the application of Artificial Intelligence (AI) to traffic flow considered the control of traffic signals by predefined rule-based system and fuzzy rules. These methods usually need a predefined model of traffic conditions in order to predict traffic flow. Alternative centralized techniques have communication and computational overheads, in addition to reliability and real time control issues. While studies have applied AI to control signal timing for an isolated intersections, the techniques explored do not scale well for multi-intersection networks.
The nature of urban traffic is dynamic. In addition, traffic signal timings at one intersection influence the traffic congestion at the neighboring intersection. These facts reinforce the necessity of using hybrid computational intelligence methods to design efficient traffic signal timing controllers. The ability to learn from experience is one of the characteristics of AI methods that makes these methods suitable to address real-world problems. Self-organizing AI methods are robust to dynamic changes of conditions. Furthermore, to overcome excess computational demand in central control design, distributed control systems are a potential option. In distributed models, each intersection can have its own controller with its own traffic data load. Distributed controllers are able to include all benefits of a multi-agent system, which increases reliability, robustness, and accuracy of the system.

In this thesis, a Q-learning AI technique is first investigated and benchmarked. After understanding its drawbacks, an enhanced version of the Q-learning controller is developed. As the next step, different traffic signal controllers with intelligent methods such as Q-learning, neural network, and fuzzy logic systems are designed for an isolated intersection network model and compared under the same test bed conditions.

The results reveal improved performance of the proposed AI-based controllers compared to fixed-time controllers. Among the aforementioned controllers, the best results were achieved using fuzzy control. Therefore, this research focused on improving the quality of fuzzy control through using optimization methods and combined with a neural network. The outcome is an adaptive neuro-fuzzy inference systems (ANFIS) controller with optimal rule base that has better performance than controllers developed previously. Applying type-2 fuzzy logic systems and the recently introduced cuckoo search optimization method led to the development of an efficient traffic signal timing controller, which is extended for a network of multi-intersection.

A distributed AI control system is also considered for a multi-intersection network. The controller of each intersection overcomes partial view issues through learning and
communicating with neighbor intersections. The computational cost of the whole network is divided between different controllers. Optimal type-2 and type-1 ANFIS controllers, a conventional fuzzy controller, and fixed-time controllers are also developed based on proposed distributed control system for multi-intersection traffic signal timing. Optimized type-2 ANFIS controller has revealed its robustness and higher performance under different test conditions compared to the other developed controllers.
I would like to dedicate my thesis to
my kind and supportive husband Sina
List of Publications

Journal papers:


Conference papers:


Research Contributions

This research contributes to developing traffic signal timing controllers aiming to reduce the amount of travel delay times for vehicles in the traffic network. In this way, selected artificial intelligence techniques and optimization methods are studied, extended, and applied. The contribution of this research can be summarized in the following points:

- A comprehensive review of studies in traditional traffic signal timing and also computational intelligence techniques for design of optimized traffic controllers for both single and multi-intersections is done in this thesis. Considering deficiencies of previous works and applying combination of promising AI techniques and new optimization methods leads to creating powerful controllers for traffic signal timing.

- A comparative analysis of computational intelligence techniques including reinforcement learning, neural network, fuzzy logic systems, and neuro-fuzzy systems for traffic light timing for the first time is done in this work.

- Designing a Q-learning controller based on Abdoos et al. [1] research and improving its performance for a single intersection traffic model. Next step is developing optimized neural network, and fuzzy controller for the same traffic model. All controllers in this thesis are designed with flexible cycle time. In previous studies the controllers could consider fixed amount of time as green time extension while the possibility of having continuous green time added to proposed controllers in this work.
- Developing a fuzzy controller with a predefined rule base, then enhancing to type-1 ANFIS and type-2 ANFIS controller. In addition, different optimization methods such as genetic algorithm, simulated annealing, and cuckoo search are used to obtain optimal parameters of the aforementioned controllers. Cuckoo search as one of the most recent introduced meta-heuristic optimization methods for the first time is used in combination with type-2 ANFIS controller in this thesis.

- In the last chapter the developed controllers are applied for a multi-intersection network. The designed controllers in this chapters are distributed controllers that use multi-agent systems for this purpose. Each intersection has its own controller and just use its own and its neighbor intersections traffic data for signal timing. This approach can reduce the data overload on signal timing controllers.

- Developed traffic signal timing controllers in this thesis are able to propose different range of number as traffic signal phase duration and they determine these numbers at the start of each phase. This characteristic provides the option of using timer traffic signals, which is useful for drivers to know how long they have to stay in traffic, while using timer based signal lights is not appropriate for controllers doing timing by extending the current phase or termination of that based on current detected traffic.

- Designing an isolated intersection traffic model and a multi-intersection network model with nine intersection in PARAMICS. In addition, considering different traffic scenarios for both single and multi-intersection network to test the robustness and reliability of the designed controllers.
Contents

Abstract ........................................................................ iv
List of Publications ..................................................... viii
Research Contributions ................................................. xi
List of Tables ............................................................... xvii
List of Figures ............................................................. xviii

1 Introduction ......................................................... 1
   1.1 Overview ...................................................... 1
   1.2 Motivation .................................................... 1
   1.3 Traffic signals and control methods ....................... 2
   1.4 Research objective .......................................... 4
   1.5 Thesis outline ................................................. 5

2 Literature review .................................................... 7
   2.1 Traffic signal control methods ............................... 7
       2.1.1 Overview ............................................... 7
       2.1.2 Fixed-time control .................................... 9
       2.1.3 Traffic actuated control ............................ 10
       2.1.4 Traffic adaptive control ......................... 13
3 Q-learning controller for a single intersection 37

3.1 Overview .................................... 37
3.2 Machine learning ............................... 37
  3.2.1 Reinforcement learning and Q-learning .......... 38
  3.2.2 The Q-learning algorithm .......................... 39
3.3 Q-learning controller based on Abdoos model [1] .......... 41
3.4 Improved Q-learning controller for an isolated intersection .... 43
3.5 Experimental results and discussion .......................... 45
  3.5.1 Introducing the testbed and conditions ............. 46
  3.5.2 Experimental setting for Q-learning traffic signal timing controllers 48
  3.5.3 A comparison of Abdoos method, proposed Simple Agent Q-learning (SAQL) method and fixed-time method .... 49
3.6 Conclusions ..................................... 53

4 Intelligent controllers for a single intersection 54

4.1 Overview ......................................... 54
4.2 Related background .................................. 55
  4.2.1 Neural network .................................. 55
  4.2.2 Fuzzy logic systems (type-1 and type-2) ............... 55
  4.2.3 Adaptive neuro-fuzzy inference system (type-1 and type-2) 58
  4.2.4 Genetic algorithm .................................. 61
4.2.5 Simulated annealing ........................................... 62
4.2.6 Cuckoo search optimization algorithm via Lévy flight ........ 64
4.3 Part 1: NN and FLS controllers ................................ 66
  4.3.1 Design of the NN controller ................................ 66
  4.3.2 Design of the FLS controller .............................. 68
  4.3.3 Fixed-time controller ....................................... 68
  4.3.4 Experimental results and discussion for Part 1 .............. 69
  4.3.5 Conclusion of Part 1 ....................................... 74
4.4 Part 2: ANFIS controller ........................................ 76
  4.4.1 Design of ANFIS controller ................................ 76
  4.4.2 Experimental results and discussion for Part 2 .............. 78
  4.4.3 Conclusion of Part 2 ....................................... 81
4.5 Part 3: Optimized type-2 ANFIS controllers .................... 82
  4.5.1 Design of T2-ANFIS controller with SA, GA, and CS ....... 83
  4.5.2 Experimental results and discussion for Part 3 .............. 85
  4.5.3 Conclusion of Part 3 ....................................... 87
4.6 Conclusions .................................................... 88

5 T2-ANFIS distributed traffic signal controller .................... 90
  5.1 Overview ..................................................... 90
  5.2 Distributed multi-agent systems .............................. 90
    5.2.1 Agent taxonomy .......................................... 92
    5.2.2 Learning in Multi-agent System ........................ 93
  5.3 Optimized T2-ANFIS traffic signal timing controller for multi-intersection traffic network ............................................. 95
5.3.1 Cuckoo search optimization for MAS T2-ANFIS traffic signal controller .................................................. 97

5.4 Experiments and benchmarks ................................................................. 100
5.4.1 Fixed-time controller for multi-intersection network ............... 101
5.4.2 Fixed-fuzzy controller for multi-intersection network .............. 101
5.4.3 T1-ANFIS controller for multi-intersection network ............... 101

5.5 Experimental results and discussion ...................................................... 102
5.6 Conclusions ....................................................................................... 109

6 Conclusions and future works ................................................................. 110
6.1 Conclusions ....................................................................................... 110
6.2 Recommendations for future work ..................................................... 113

References ............................................................................................ 115
List of Tables

3.1 Demand of cars for each zone in first experiments. ........................................ 49
3.2 Demand of cars for each zone in second experiments. ..................................... 49
3.3 Percentage of improvement of SAQL against fixed-time and Abdoos’s method 51
4.1 FLS controller Rule Base .......................................................... 69
4.2 Parameters of NN controller .......................................................... 70
4.3 Parameters of FLS controller .......................................................... 70
4.4 Performance Order Between Controllers based on the Average Delay Time in Seconds per Vehicle .......................................................... 74
4.5 Parameters of CS in T2-ANFIS ....................................................... 85
4.6 Parameters of GA in T2-ANFIS ....................................................... 86
4.7 Parameters of SA in T2-ANFIS ....................................................... 86
5.1 Average delay time per vehicle .......................................................... 107
5.2 Performance against fixed-time Controller .................................................. 108
## List of Figures

2.1 Sample of four different traffic phases ........................................ 8  
2.2 Phase, stage, and cycle relationships .......................................... 9  
2.3 Locations of traffic detectors ..................................................... 11  
3.1 Interaction of agent and environment in reinforcement learning .......... 39  
3.2 The Sutton standard $Q$-learning algorithm [2]. ............................ 41  
3.3 This table shows the state-space defined in [1]. ............................ 42  
3.4 Training process of $Q$-learning controller .................................. 46  
3.5 Intersection with four approaching links ...................................... 47  
3.6 Snapshot of the isolated intersection in PARAMICS. ...................... 47  
3.7 Snapshot of four defined phases in an isolated intersection. ............. 48  
3.8 Comparison of the results of fixed-time, Abdoos and SAQL (scenario 1). .. 51  
3.9 Comparison of the results of fixed-time, Abdoos and SAQL (scenario 2). .. 52  
4.1 Structure of a T2-FLS ............................................................... 57  
4.2 ANFIS structure ..................................................................... 59  
4.3 Structure of a sample T2-ANFIS controller with two inputs and two membership functions for each input. ............................................ 61  
4.4 Genetic Algorithm ................................................................. 62  
4.5 Simulated Annealing ............................................................... 64  
4.6 Cuckoo Search via Lévy Flights ................................................. 65
Chapter 1

Introduction

1.1 Overview

This chapter presents the motivation of the research, the research background, research objectives, main contributions, and the structure of the thesis. It is required to use traffic terminologies in this chapter. Information regarding traffic terminologies is presented at the beginning of chapter 2.

1.2 Motivation

Higher levels of urbanization and increases in the number of vehicles have emphasized the need for efficient transportation systems. Developed cities are unable to reconstruct their road traffic network, therefore, providing real-time control is a necessary part of modern traffic road control systems. Increasing intersection capacity, decreasing delays, and guaranteeing the safety of people and vehicles are some of the main goals of signal control. Improvements in signal control also have the benefit of reducing fuel usage and $CO_2$ emissions by decreasing the stop and delay time of vehicles.

Various use of computational intelligence methods in research and industry provide
1.3 Traffic signals and control methods

Traffic lights, also known as traffic signals, traffic lamps, and signal lights, are signaling devices positioned at or near road intersections, pedestrian crossings and other locations to manage conflicting requirements for the use of road space. Traffic lights allocate right of way to groups of mutually compatible traffic movements during distinct sets of time.

The history of utilizing traffic signals began in 1914 in the USA, but the need to improve their performance began after World War II when road network congestion increased rapidly. Since then, three generations of traffic signal control have evolved. The first generation, the fixed-time method, requires pre-set signal sequences and manual maintenance. The Traffic Network Study Tool [9, 10] is an example of tools for calculating fixed-time plans. The focus of the second generation was to adjust the signal timing based on real-time traffic detection. Traffic inductive loop is one of the main characteristic of this generation. Inductive loop detector is an electromagnetic communication or detection system, which uses a moving magnet and causes an electrical current in nearby wire. These detectors are able to detect vehicles passing or arriving at a certain point, for instance approaching a traffic light, and use this data to adjust the timing of the lights at that intersection. The third-generation is characterized by dynamic decision making and distributed control systems. This generation is fully adaptive and signal timing is optimized progressively [11].
1.3 Traffic signals and control methods

According to the current queue length and information obtained from detectors located within the network, the sequence of signal changes and the associated timing is calculated. The decisions are based on an estimation of the incoming traffic for the next few seconds, as well as the outgoing traffic from the intersection. These systems were made possible by the computational power of standalone microprocessors, allowing operation at separate local sites.

Control methods for signal sequencing can be divided into two groups: non-optimized and optimized. A set of heuristic rules to describe relationships between signal timing and traffic conditions is employed in non-optimized models, such as Sydney Coordinated Adaptive Traffic System (SCATS) [12]. In optimized methods, the goal is to minimize estimated vehicle delays and stops, balance the saturation of approaching links and maximize intersection capacity. On-line and off-line modes are considered for optimized methods. Earlier research in optimization led to offline fixed-time planning. To minimize the average delay time of vehicles for isolated intersections, Webster [13] used unconstrained optimization. Gartner et al. [14], used mixed-integer-programming (MILP) for preparing coordination between a group of intersections for optimizing network control. Split Cycle Offset Optimization Technique (SCOOT) [15], is an example of an on-line optimized system. In this system, optimization routines are applied for cycle, split, and offset respectively and the optimization routines are limited to choose from a predefined set of incremental changes in signal timing.

In order to have an effective signal controller for a large traffic network, distributed control approaches are suitable [5]. A distributed control system can be developed in a multi-agent structure in which each intersection represents an agent in a multi-agent system. A distributed multi-agent system for a large urban traffic network has many benefits compared to a central controller. The advantages of distributed multi-agent systems are reviewed in chapter 5.
In this section, a history about emergence of traffic signal lights and some of the control methods that have been created to optimize their performance was reviewed. Traffic signal control has evolved over time to accommodate increased traffic volumes. Early methods are being replaced with modern methods. These modern methods provide the ability to optimize the traffic network in terms of delays and traffic flow, and recent literature suggests that decentralized control combined with a multi-agent approach and machine learning techniques is a valid area for further investigation. More explanations about traffic signal lights controller are presented in next chapter.

1.4 Research objective

Reinforcement learning ($Q$-learning) is a well knowing method for unknown environments and neural networks (NN) have a reputation for speed and accuracy for approximation. Alternatively, for applications with highly stochastic and uncertain inputs, fuzzy logic systems (FLS) provides a viable solution. In type-1 FLS, uncertainty is not considered in rules because of the assumption of availability of a crisp value for membership grade. Type-2 FLS overcomes this limitation by lower and upper membership functions for the primary membership grade as well as the secondary membership grade associated with each single primary membership function. In addition, multi-agent systems or distributed control systems have many advantages over conventional controllers. They have increased efficiency and speed due to a high level of parallelization, higher reliability and robustness in case of agent failures, reduced total cost of the system, increased re-usability and easy upgrade options and scalability [5]. Multi-agent systems are more flexible for decision making in environments where information is not completely available, therefore, using decision making systems with the minimum required amount of historical data and training is suitable for multi-agent systems.
The objective of this thesis is to introduce a powerful adaptive controller for a multi-intersection traffic network by utilizing a combination of machine learning and optimization techniques. In this thesis, for the first time a comparison between the performance of $Q$-learning, NN, and FLS controllers is done. This comparison is for finding the most suitable method as a traffic signal timing controller. A single intersection is considered for this purpose. After finding the most suitable method, the aim is to improve and extend it for a multi-intersection traffic model. In this research, different optimization methods such as genetic algorithm (GA), simulated annealing (SA), and cuckoo search (CS), which is a new inspired optimization method, are applied to find the optimal parameters of the adaptive distributed intelligent traffic signal controller.

1.5 Thesis outline

The thesis outline is organized as follows:

- **Chapter 1** gives a brief introduction and background of the controlling traffic signal lights, the research objective, main contributions, and the outline.

- **Chapter 2** reviews the history of different traffic signal timing controllers. After introducing the conventional traffic control systems, the most recent controllers that apply intelligent methods to increase the performance of the traffic signal timing are explained.

- **Chapter 3** introduces a benchmark $Q$-learning controller based on [1]. In addition, the improved version of the $Q$-learning controller is presented in this section named SAQL. Evaluation of the performance of both controllers are done in a single intersection traffic model designed in PARAMICS which is traffic simulator.
1.5 Thesis outline

- **Chapter 4** is composed of three parts:
  
  Part 1 introduces two intelligent controllers for a single intersection. One of these controllers is designed based on NN and the other one is based on FLS. To reach optimal controller GA is used as the optimization methods.

  Part 2 is about designing an adaptive neuro-fuzzy inference system (ANFIS) controller. ANFIS gives the opportunity of using FLS in traffic signal controlling without the need for a pre-defined rule base. Parameters of the ANFIS controller are optimally tuned using GA and ANFIS controller obtains its optimal rule base. GA-FLS, a FLS controller with fixed and predefined parameters, and a fixed-time controller with three different values are also designed and implemented to evaluate the performance of the ANFIS controller.

  Part 3 introduces the improved version of ANFIS controller. In this part a combination of type-2 FLS and ANFIS method is used. Furthermore, to find the suitable optimization method for optimizing the parameters of the ANFIS rules, GA, SA, and CS are used in the structure of the controller.

- **Chapter 5** presents a distributed controller using type-2 ANFIS for a multi-intersection traffic network. A comprehensive evaluation of the proposed controller is done and its performance is compared with type-1 ANFIS, fixed-fuzzy and fixed-time controller, which are all designed for a multi intersection traffic network.

- **Chapter 6** concludes the thesis and provides recommendation for future research work.
Chapter 2

Literature review

2.1 Traffic signal control methods

2.1.1 Overview

In order to propose a new and efficient traffic signal control model, it is necessary to review existing traffic signal controlling methods, which is provided in this section.

The terminologies used in traffic signal control are defined as:

*Green time:* Period of time in which vehicles in a lane are allowed to cross an intersection.

*Link:* A group of adjacent lanes on which traffic forms a single queue.

*Phase:* A set of unique traffic signal movements, where a movement is controlled by a number of traffic signal lights that change colour at one time. Phase is the part of the cycle assigned to a fixed set of traffic movements, when any of these movements change, the phase changes Fig.2.1.

*Stage:* A set of one or more traffic and/or pedestrian phases that receive a green signal during a particular period of the cycle Fig. 2.2.

*Cycle:* The time required for one full cycle of signal indications.
2.1 Traffic signal control methods

Fig. 2.1. Sample of four different traffic phases

**Split:** Total time allocated to each phase in a cycle. It is composed of green time, amber or yellow time and red time.

**Offset:** Time lag between the start of green time for two adjacent intersections to allow free flow of vehicles without facing any red signals.

**Delay:** The total mean stopped time per vehicle for each lane in the road traffic network.

**Saturation flow:** The maximum number of vehicles from a lane group that would pass through the intersection in one hour under the prevailing traffic and roadway conditions if the lane group was given a continuous green signal for that hour.

**Progression:** The time relationship between adjacent signals on a roadway which permits a platoon of vehicles to proceed through the signals at a planned rate of speed.

Based on the architecture type used to adjust required green time for each phase, different classifications exist for traffic signal control [7, 16]. This research considers the following classification for categorizing the implemented traffic signal control to date.

- Fixed-Time
- Actuated
- Adaptive
2.1 Traffic signal control methods

2.1.2 Fixed-time control

In fixed-time control, all green phase durations and their order are fixed. In this method, traffic demand is considered to have a fixed distribution. Then, based on historical data, the related time for traffic signal lights is applied. Installing traffic detectors are not required for fixed-time method and it reduces the cost in comparison to adaptive and actuated methods.

Based on the historical traffic volume, the cycle time is divided to several phases. After each phase, a fixed amount of time is required for clearing the intersection and starting the next phase, called the safety time. For shorter cycle time, the amount of safety time is increased per hour. Therefore, there is lower overall capacity for intersections with shorter cycle times. On the other hand, longer waiting times and longer queues are the consequences of longer cycle times.

In this regards, Webster [13] proposed a formula based on the flow rate of each lane in a link to find an optimal cycle and appropriate duration for green time in each phase. Two classifications are considered for fixed-time control and these classifications are proposed based on the method applied for calculating the green time.

---

Fig. 2.2. Phase, stage, and cycle relationships; mutually compatible phases are grouped into stages, and certain phases may appear in more than one stage, such as phase B in stage 1 and stage 2.
2.1 Traffic signal control methods

- Progression based method that aims to maximize the bandwidth of the progression. PASSER (Progression Analysis and Signal System Evaluation Routine) [17], MAXBAND [18], and MULTIBAND [19] are included in this set.

- Disutility based method, an approach based on minimizing performance measures such as the number of stops and overall travel delay time. TRANSYT-7f [9] and SYNCHRO [20] are based on this method.

Fixed-time control is not capable of adapting to real-time traffic as they cannot adapt to variations in traffic patterns. Furthermore, for events, accidents, and other disturbances that may disrupt traffic conditions, fixed-time methods are not suitable.

2.1.3 Traffic actuated control

The limitation of fixed-time traffic signal controller in coping with sudden variation of traffic and its high dependency to historical data are overcome by real-time vehicle actuated control methods. Traffic-actuated control methods utilize inductive detectors to observe the actual traffic situation. The traffic-actuated controller must have the ability to determine whether the last vehicle of the queue that has formed at the stop line during the red phase has passed. This detection is useful for having efficient termination of green time, and it is performed by measuring the gap between vehicles. The green time is terminated when the gap between vehicles is larger than the threshold maximum gap. However, many traffic actuated controllers extend the green time to ensure that the green phase terminates safely. These extensions continue until the intervals between vehicles are sufficiently long that the termination be more efficient or until a pre-specified maximum green time has been reached.

Usually, four major zones are considered in this method: Zone 1, Zone 2, Option zone and Comfort zone. Zone 1 and Zone 2 are placed very close to the stop line intersection.
Zone run runs from the stop line for 3m. Zone 2 runs for 20m after Zone 1. The Option zone is the overlapping region extending beyond zone 2. The green time of the phase can be effectively terminated if the first vehicle after the stop line exists in this region. Related to the traffic conditions, the traffic flow is considerably slower in the Option zone. The region beyond the Option zone is the Comfort zone. In this area the vehicle flow is steady and is not influenced other than a queue build-up. Extension of green time is closely dependent on the presence of vehicles in any of these zones, however, the priority is different in each of them. Fig. 2.3, shows an example of detector placements for a single lane of an intersection [5, 16].

![Fig. 2.3. Location of detectors as straight lines on the bitumen.](image)

In traffic actuated control, it is necessary to consider a strategy that ensures the coordination between different phases, and this coordination is set based on the same principles used for fixed-time method. The refinement gap extension problem (to ensure sufficient time between signal changes to minimise accidents) and relative delays of traffic flows are the factors for decision making about extending or terminating of the green signal of a
The optimal placement of detectors at an intersection impacts the performance of this method. In addition, by increasing the number of detectors the accuracy of the system is improved. In actuated methods, a pre-specified block period time is considered for extending the green time of a phase, therefore, detection of sparse traffic can have considerable influence on delay time [21]. System D [22], MOVA [23], LHVORA [24], and SOS [25], are samples of actuated traffic control system. In the following, explanations about System D and MOVA are presented.

- **System D**: System D or vehicle actuated [22], employs vehicle detectors to estimate the time when the flow-rate over the stop-line is below the saturation level. According to the output from the detectors, a stage will be extended until no vehicles are detected in critical interval of links [26].

- **MOVA**: The aim of MOVA [23] is dynamic operation at isolated intersections. In this system, the signal timings are generated in each cycle, and the optimization of the objective function is performed by minimizing the delay and stop time in an uncongested situation and maximizing the capacity of lanes in a congested situation. Signal timing in each cycle allows the adjustment of timings according to traffic demands. MOVA uses pairs of upstream detectors to obtain vehicle gaps in order to terminate green extension. Updating the signal plans is performed every half-second. Signal timing control in this method is based on the real-time traffic demand; however it needs a strategy to coordinate the cooperation of intersections in a network. The extension time for the green time of a phase is fixed and it influences the delay time. Furthermore, the position and number of detectors are important items in the accuracy of this method.
2.1.4 Traffic adaptive control

Parameters like time, day, season, weather, and some unpredictable situations such as accidents, special events or maintenance activities are highly influential on traffic load. The aim of traffic-adaptive control systems is to take these elements in account in order to predict more efficient green times. In adaptive traffic control systems, the traffic condition is sensed and monitored continuously and the timing of traffic signals is adjusted accordingly. Adaptive systems, such as SCOOT (Split, Cycle and Offset Optimization Technique), and SCATS (Sydney Coordinated Adaptive Traffic System), have been used from the mid 1970s. These adaptive methods have been successfully applied in various cities around the world.

Adaptive traffic systems can be grouped in three categories [16]. In first category a library of pre-stored signal control plans is applied. These plans are developed off-line on the basis of historical traffic data. Plans are selected by considering the time of day and day of the week, and this selection is done directly by the operator, or by matching the current library in order to have the best suitable plan to fit the measured traffic condition. One of the limitations of the first category is that the registered traffic conditions, which trigger a response may become out-dated or completely changed by the time the system responds.

Systems in second category use an on-line approach where the plans are prepared based on real-time supervision data and predicted values. It is possible to have optimization every five to fifteen minutes, however, in new systems there is no more than one timing plan every ten minutes in order to avoid transition disturbance.

The same strategy as the second category is used in the third category, however, there is a choice made on the frequency with which the signal timing plans are revised. It allows the parameters of signal plans to change continuously in response to the real-time measurement of traffic variables. The amount of improvement obtained by using traffic-adaptive systems compared to fixed-time and traffic-actuated is not necessarily the same
in another adaptive system. The performance of different adaptive systems is completely
dependent on the network geometry and traffic demand chosen in the benchmark study.
Some examples of second category methods are SCATS/GLIDE, SCOOT and MOTION,
while OPAC, PRODYN, RHODES, UTOPIA/SPOT, TUC and HMS are examples of the
third category.

Among the different methods in the adaptive strategy both non-optimized and optim-
ized approaches exist. An explanation of these methods is presented next.

Non-optimized systems

In order to adjust signal timings, non-optimized systems use a set of heuristic rules and
as such real-time adjustments to signal timings are not optimized with respect to perform-
ance measures. In practice, non-optimized systems mechanically match detected traffic
conditions to pre-set heuristic rules. The advantage of these systems is in their simple
implementation and robust control. SCATS is a system in this category.

- SCATS/GLIDE: Sydney Coordinated Adaptive Traffic System (SCATS) [12] was
developed in the early 1970s by the Roads and Traffic Authority of New South
Wales, Australia. The structure of this system was a distributed, three-level, hier-
archical model. To scale the control of a large network, a central computer, regional
computers, and local intelligent controllers are applied. The regional computer has
the ability to perform adaptive control independently without the help of the central
computer. The central computer has the responsibility to monitor the system per-
formance and equipment status. The SCATS control structure enables it to be easily
expanded for various sized traffic areas.

SCATS, via several on-line calculations, chooses a mixture of cycle, split, and offset
from pre-determined sets of parameters. The local controller has maximum freedom
2.1 Traffic signal control methods

to act in traffic-actuated mode. The system calibrates itself automatically based on the data it receives. For controlling, the system is divided into many smaller subsystems, each containing up to 10 intersections. Each subsystem has its own minimum, maximum and geometrically optimum cycle lengths. Subsystems can link together and provide larger systems in order to coordinate larger groups of signals.

For managing the links between subsystems, linking plans are used. In situation of having linked subsystems, the cycle time is extended. Many operations are required for combination of subsystem plans, link plans between subsystems, flexible cycle length, and setting different offsets. For each subsystem, four background plans are stored in the database and the appropriate cycle length and plan is selected independently to fit the traffic demand. In order to reach this aim, a number of detectors are considered in each subsystem and defined as strategic detectors. These detectors are stop-line detectors at key intersections. From the data gathered by strategic detector, various factors are calculated. These factors are useful for decision making about the necessary changes required for the cycle and plan. Strategic options, minimum delay, maximum stops, or maximum throughput are some sample factors which may be chosen for the decision making. SCATS acts as a heuristic feedback system and adjusts signal timings based on traffic flow changes during previous cycles.

SCATS has been widely used in several cities in Australia, New Zealand, USA, China, Singapore, Philippines and Ireland. In Singapore a special version of SCATS is used, named GLIDE (Green Link Determining), and has been adapted for local traffic network structure and requirements.
2.1 Traffic signal control methods

**Optimized systems**

Optimized systems are usually based on the state-space of the control system and sequential decision-making. These methods aim to optimize the generalized control performance during a time period. The sequence of signal changes and associated timings are the items calculated by applying dynamic programming (DP) and utilizing Bellmans equation [27]. Definitions of state variable are an important part in the DP formulation for the traffic problem. In Bell et al.’s work [28], the state of a traffic signal control is defined as a composite of the state of the traffic and the state of controller. The number of vehicles that have formed a queue at the stop line of each link and the others will arrive in the near future make the state of the traffic at an intersection. The state of the controller considers the signals that are green, any changes that will occur in near future, the times at which they will end and the expiry time of any maximum and minimum permitted periods.

This definition imposes a difficulty in implementing Bellmans equation. There will be computational difficulty if all possibilities for each of these state variables are included. Related to this problem, Bell et al. refined their definition. They claimed normal backward dynamic programming techniques are not suitable for applying in real-time control because of a large number of state sequences and calculations commence at the end of the look-ahead where information on arrivals is least certain.

Robertson and Bretherton [15], proposed DYPIC as a DP approach useful for analytical aims. The limitation of implementing DYPIC for engineering purposes is due to the computational difficulty of the DP solution. In regards to this problem, a quadratic function to approximate the exact value function is proposed. The heuristic solution is developed with an approximation function and it adapts with a rolling horizon [29]. The planning horizon is split to a ‘head’ and ‘tail’ period. The ‘head’ is formed based on detected information and the ‘tail’ is based on predicted traffic information. In the next step, an optimal policy
is calculated for the whole horizon, which is only done for the ‘head’ period. Finally, when the ‘head’ period expires, the new information becomes available, and then the process rolls forward and repeats.

Among adaptive control systems, OPAC, PRODYN, and RHODES are based on state-space representation of the system and involve the concept of DP, whereas MOVA and SCOOT apply optimisers to calculate signal timing by considering a set of performance measures. More explanation about, OPAC, UTOPIA, RHODES, and SCOOT are presented in below:

- **OPAC**: OPAC [30] is a distributed real-time traffic signal control system. This system employs an optimal sequential constrained search (OSCO) instead of dynamic programming to plan for the entire horizon and utilize terminal cost to punish the remained queues at the horizon. The horizon is 60 seconds with 10 second as the head and the remainder as the tail. It is reported by Gartner [31] that OPAC had 5-15% improvement against existing traffic-actuated methods. OPAC performance was more significant in high degree saturation. However, using OSCO search reduces the flexibility of decision making in OPAC, and there will be questions because of the long horizon (60 sec) about optimization based on predicted information for that far into the future when the decisions planned for the tail are never implemented.

- **UTOPIA**: UTOPIA [32] is a hybrid control system that combines online dynamic optimization and offline optimization. A system with two levels: area level and a local level are designed for this reason. A reference plan is generated by the area controller and local controllers adopt this reference plan and also coordinate signals in neighbor intersections. A 120 seconds rolling horizon is applied in this control system and the process is repeated every 3 seconds. The AUT (Automatic Updating TRANSYT) module is developed to update the reference plans automatically. In
the next step, AUT prepares data for TRANSYT calculation and starts TRANSYT optimization. Implementing UTOPIA led to 15% increase in average speed of private vehicles and 28% for public transport with priority.

- **RHODES**: Real-time Hierarchical Optimized Distributed Effective System (RHODES) [33], is a hierarchical adaptive traffic signal control system. This system uses dynamic network loading, and network flow and intersection control is done in three layers. An estimation/prediction component and a control component are placed for each level. The real-time detected information is used to estimate link free-flow speed, queue discharge rates, turning probabilities.

  Online adjustment of traffic signal lights with estimation influences the decision making of the control component. Dynamic programming (DP) is employed for the intersection controller and the related DP algorithm is completed when all possible decisions for each phase has been evaluated. Then the sequence of phases and their duration are determined. The network flow controller counts all possible decisions to fix the conflicting demands. Comparison of RHODES with semi-actuated systems reveals 30-50% decrease in delay time. The main deficiency of RHODES is the computational burden on the intersection controller, and it is not clear in the literature how the system handles the computation demand in real-time.

- **SCOOT**: SCOOT [15], is a control system developed in U.K. by the Transport Research Laboratory. This system is a centralized adaptive system that optimizes green time splits, offsets, and cycle length separately. By minimizing the maximum degree of saturation on approaching links to the intersection, split optimizer equalizes saturation in an intersection.

  For calculating the degree of saturation, traffic flow profiles are used. Five seconds before the expiry of the current green stage, the optimizer is run to determine how
2.1 Traffic signal control methods

the stage should be set for the next step, i.e. should it start four seconds earlier, later or remains the same.

Once per cycle the flow profile is used by the offset optimizer to predict the performance measures during a cycle for an isolated intersection. Predictions are useful for evaluating if the offset must reduce by four second, increase by four seconds or remain the same. The cycle optimizer operates on a region of intersections that it is expected controlling of that part lead to a good progressions in the network’s traffic. The optimizer’s focus is on degree of saturation for all links in the region. For the degrees at the ideal level, the Minimum Practical CYcle (MPCY) length is increased by a small fixed step, and for the below ideal level degrees, the optimizer reduce the MPCY by a small fixed step.

Several studies focus on traffic signal timing through mathematical techniques, for example, Jang et al. [34] equalized queue growth rates across links in over-saturated urban roadway networks and thus postponed queue spill backs that form at the localized sections of networks. Chen and Hu [35] focused on finding equilibrium under the interaction of signal setting and traffic assignment. They solved the problem by introducing a bi-level framework. The upper level solved for signal setting parameters based on flow distributions, including cycle length and green splits. The lower level solved for user equilibrium dynamic traffic assignment flows in a traffic network.

Adaptive controllers consider different parameters in their decision making that make them very accurate, however, this accuracy comes at a significant computational cost. Actually, the main drawback of the introduced controllers are in the amount of information need to be communicated to the central controller. In addition, the computational cost required for data mining, extracting the degree of saturation and traffic patterns.
2.2 Computational intelligence in adaptive systems

The aforementioned systems react to different traffic conditions by adjusting control variables. These systems continuously monitor the traffic and they are characterized by a feedback sequence of the control output. However, they do not usually employ machine learning techniques for sequence feedback. Furthermore, these systems do not utilize accumulative information for improving their techniques. Machine learning techniques, when applied to traffic signal control, have the ability to address these issues and improve the traffic control systems. Intelligent adaptive controllers employ machine learning techniques and benefits learning in controlling. These control systems are characterized by including two iterative parts, one that provides feedback to the process and the controller, and the other that performs parameter adjustment.

A review over previous works shows higher popularity of $Q$-learning, NN, and FLS among the other intelligent methods for controlling traffic. In this section a review of previous works in these three categories will be presented. Researchers applied hybrid machine learning methods in some works. It is attempted to present the different works in three separate categories, however, there may be some overlaps between them. Related works to this part are explained in follow:

2.2.1 Reinforcement learning controller for traffic signal timing

Most existing traffic control systems need predefined model of traffic flow to have a short-time prediction of future traffic condition. In $Q$-learning no prespecified model of the environment is required and relationship between actions, states, and environment are learned by interaction with the environment.

In this regard, for the first time Thorpe [36] studied using reinforcement learning for traffic signal control [37]. Thorpe applied SARSA [38] to a traffic control problem. He
evaluated the performance of SARSA with three different representations of an specific state, and used a NN to estimate the reinforcement value. In his study, states were defined by the number and position of vehicles in all directions ending to an intersection. Action for each state was set to change the lights’ color from red to green and vice versa. These features were combined in three different ways. The first representation called vehicle count. In this approach Thorpe considered ten partitions based on the number of cars, and then by considering all pairs of combination of these ten partitions for east-west and north-south directions and considering two possible modes for traffic lights, 200 \((10 \times 10 \times 2)\) states were inputs to the learning agent. For the second representation or fixed distance, Thorpe divided each lane to 110-foot intervals that led to four partitions at each lane. An occupied bit is set to show the absence or presence of vehicles in each partition, which causes having eight components for whole east-west or north-south lane and one component for the traffic light’s color. Totally, there were nine components vector as the input to the NN. Third representation was variable distance, similar to the previous one but with a variable distance for each partition. The distances were set to 50, 110, 220, and 400 feet, and there were again four partitions for each lane and for traffic light. The learning agent in this representation had nine components input similar to fixed distance.

Thorpe set the reward to \(r = -1\) for each step of trial to reach the goal. Evaluation is done in a \(4 \times 4\) network, and during the performance evaluation the best result for total simulation steps required to clear vehicles from the simulation environment belonged to variable partitions, and for the case of minimum travel time fixed distance had the best result.

Wieiring [37, 39] proposed a transition model that estimates waiting time for both green and red lights at each intersection. They applied multi-agent reinforcement learning to control traffic signals. Their method was car-centric; each car estimates its own waiting time and communicates it with the nearest traffic light. For state definition they considered
position and orientation of vehicles in the queue, and their destination address. The action was set to change between red and green phase, and for the reward function, if a car stayed at the same place \( r = 1 \) otherwise \( r = 0 \). In this system the goal was to minimize the overall waiting time, and it learned the assignment function for estimating overall waiting time of vehicles. During their experiments both local and global communication scenarios for better decision making of traffic lights are applied.

Abdulhai et al. [40], applied \( Q \)-learning as a traffic controller. They performed the experiment for an isolated intersection, but had some suggestions for the case of multi-agent in this study.

In the case of single intersection, states are the length of queues on four approaching links to the intersection and the elapsed phase time. Action was defined as extending the current red or green phase or changing to the next one. In this study, reward was considered as a penalty and it was the total delay time between two successive decision by the vehicles in the queues formed behind stop light of four approaching link of the intersection. In addition, a power function is used to approximate balancing of the queue length in order to modifying the reward, which is directly proportional to the queue length in each one second step. This was useful to prevent agent of being indifferent about too long, too short queue or situation of equal-length of queues.

For the case of multiple intersection some other states such as the split between two intersection may be added, and the reward would be the weighted summation of all single intersection by considering highly weighted reward for the main road. Abdulhai et al. have shown that reinforcement learning and especially \( Q \)-learning is a promising approach to build an adaptive traffic signal controller in [40]. The result of experiments for single intersection showed that \( Q \)-learning outperformed the pre-timed controller for variable traffic flows, and either slightly outperformed or was equal to the pre-timed controller for situations of the constant or uniform flows.
In study done by Wunderlich et al. [41], Longest Queue First (LQF) was proposed as a traffic signal scheduling algorithm for an isolated intersection. The LQF algorithm was designed for a signal control problem and the concepts were employed from the field of packet switching in computer networks. This method utilized a maximal weight matching algorithm to minimize the queue sizes at each approaching link and led to a significantly lower average vehicle delay through the intersection. It was proved that LQF was stable and had strong performance under various traffic scenarios.

The authors decided to apply LQF in multi-intersection network in their next study [42]. In a multi-intersection network a phase scheduling decision at one intersection would largely affect the traffic conditions in its neighbor intersections and applying LQF became a more complex task. In this research reinforcement learning is used in order to have the capability to have distributed control as needed for scheduling multiple intersections. In fact, they introduced a novel use of a multi-agent system and reinforcement learning framework to obtain an efficient traffic signal control policy. The focus was at minimizing the average delay, avoiding congestion and intersection cross-blocking. The network contained five intersections and each intersection is governed by an autonomous intelligent agent.

A central agent and an outbound agent were two kinds of agents employed in this work. The outbound agents schedule traffic signals by following the LQF algorithm. These agents provide traffic statistics for the central agent. The central agent learned a value function driven by its local and neighbors traffic conditions. The proposed methodology in their work utilized the $Q$-learning algorithm with a feed-forward NN in order to perform $Q$-value function approximation. In the setting of $Q$-learning method the state is represented as an eight-dimensional feature vector, in which each element represented the relative traffic flow at one of the lanes. For the outbound intersection agent, only local traffic statistics is considered, but the central intersection agent had access to all states of its neighboring intersections, increases the dimensionality of the state space. Action set was defined as the
eight different combination of available phase [41]. The reward was considered in the range from -1 to 1, where positive reward values were obtained if the current delay is lower than the previous time step. The agent received a penalty (negative value) if an increased average delay is observed. Experimental results revealed the higher performance of multi-agent control based on reinforcement learning against LQF governed isolated single-intersection control.

Prashanth and Bhatnagar [43] proposed the feature based reinforcement learning for controlling traffic signals. They also claimed that using feature based state-action algorithms made their method appropriate for using in high-dimensional setting of a multi-intersection network. Authors mentioned their work is different from prior works like Abdulhai et al. [40], that required full state representation and it was not practically possible to implement them. Their method did not require the precise information on elapsed time and queue length. To perform that they divide the queue length in three sets: low, medium, high and put a threshold for elapsed time to check if the detected elapsed time is higher than threshold or less than it takes place in two different groups. They compared the performance of the proposed method against fixed-time, longest queue and also the algorithms proposed in [40] and [44], and the proposed feature based algorithms outperformed all the others.

Abdoos et al. [1] presented an approach for controlling traffic signals in a network of 50 intersections based on $Q$-learning. Each intersection was considered as an agent and the whole network formed a multi-agent system. In their research, the average length of queue in approaching links was the states of $Q$-learning and the number of permutations of the approaching links determined states’ number. During the experiments, they considered intersections with four approaching links, therefore, state space consists of 24 states and different phase splits of the cycle time were the proposed actions in $Q$-learning. Phase split refers to the division of the cycle time into a sequence of green signals for each group of
approaching links. In addition, cycle time set as a fixed value and a minimum green time is adjusted for each phase. Reward was considered as inversely proportional to the average lengths of the queues in the approaching links, which is normalized to remain between 0 and 1.

In their next work [45], they developed a holonic multi-agent system to model a large traffic network. In this study, each intersection had a similar structure to [1] and they presented as homogeneous agents. The result of their research revealed that the performance of the individual $Q$-learning and holonic $Q$-learning is almost the same. The average standard deviation of delay time for holonic $Q$-learning was less than the individual $Q$-learning, which shows that they are clustered more closely in holonic $Q$-learning and are more reliable.

Other studies that Abdulhai had contribution, related to controlling traffic, are [46, 47, 48]. In [48], an adaptive traffic signal controller designed, which was using a multi-agent reinforcement learning approach. Each controller, agent, was responsible to control traffic lights timing around a single traffic junction. EI-Tantawy [48] proposed two possible modes: firstly independent mode, where each intersection controller works independently of other agents; and secondly integrated mode, where each controller coordinates signal control actions with neighboring intersections. They tested the model on a network of 59 intersections in the lower downtown core of the City of Toronto, Canada, for the morning rush hour. Their results showed reduction in the average intersection delay ranging from 27% in mode 1 to 39% in mode 2.

Other researchers also applied reinforcement learning and specially $Q$-learning in developing the traffic controller. For example, in [44], a self-organizing traffic light control method is presented. The phase of a lane is changed to green if the elapsed time during the red phase hits a certain threshold. This is also useful to recognize that the number of cars on the lane are above the threshold and the queue length is indirectly used for signal
configuration. In [49], a context detection reinforcement learning method was proposed, which was able to create a partial model of the environment based on demands. The partial model improved or a new one was constructed through the time to satisfy the demand. Adaptive reinforcement learning controller are proposed in [50, 51] to signalizing a model free traffic environment. Also, Houli et al. [52] used reinforcement learning to propose a multi-objective control algorithm. They predict the overall value of given vehicle’s states by using reinforcement learning.

The challenge for all $Q$-learning controllers is to manage the huge amount of state-action space. One of the solutions to reduce the number of states is categorizing possible states in groups. Although this approach increases the learning rate, limiting the number of states to the number of groups decrease the accuracy of the system. Based on the review, proposed $Q$-learning controllers usually consider the extension of green time as an action in $Q$-learning. Generally the extension time is a fixed period of time, which may repeated until reaching the maximum threshold. This fixed period of time is an assumption causes low efficiency. Considering some predefined numbers as possible green time similar to Abdoos et al.’s [1, 45] studies has the same deficiency. Preparing enough data to train the system or a suitable simulation is the other issues for $Q$-learning controller. $Q$-learning without enough training samples cannot converge to the optimal results. However, $Q$-learning is a beneficial methods to have online learning as it can improve its performance and adopt to the new situations.

### 2.2.2 Neural network controller for traffic signal timing

Adaptive controllers based on NN are recommended in many other studies. For example, Spall and Chin.[3] employed simultaneous perturbation stochastic approximation (SPSA) [53] based gradient estimates with an NN controller for optimizing the system. SPSA
was used for modeling the weight update process of a NN. A function was developed to take the current traffic information and generate the signal timings. In their work the current traffic information was used to solve the current instantaneous traffic issue. The system that was presented by them named S-TRAC and had these advantages: (1) It did not require any system-wide traffic flow model; (2) S-TRAC automatically adapted to long-term changes in the system such as seasonal variations while providing real-time responsive signal commands; and (3) This system was able to work with existing hardware and sensor configurations within the network of interest while additional sensors may help the overall control capability. For S-TRAC they used a feed-forward NN with 42 inputs and two hidden layers. Inputs included: (1) the queue at each cycle termination for 21 traffic queues of the simulation; (2) eleven nodes for per-cycle vehicle arrivals in the system; (3) simulation start time; and (4) the nine outputs from the previous control solution. The output layer of the NN contained nine nodes for each signals split, and for two hidden layers there were 12 and 10 nodes respectively. To evaluate the performance of S-TRAC, a simulation of a nine-intersection network of the central business district of Manhattan, New York was used. They have 10% and 11% improvement for both case of constant arrival rates of and increase in mean arrival respectively against fixed-time method during 90 days.

In study conducted by Chine et al. [54] they apply S-TRAC in a moderately congested network, in Maryland. The interruptions of the traffic flow caused by the traffic signal was evaluated. The result of their evaluation showed an average improvement of 7% with 90% confidence bound equal to ±2.5%.

Yin et al. [55] developed a fuzzy neural model to predict the traffic flows in an urban street network. Their developed model consists of two modules: a gate network (GN) and an expert network (EN). The first one classified data into a number of clusters through fuzzy approach. The EN module specifies the input-output relationship as in a conventional NN approach. In fact, GN groups traffic patterns of similar characteristic into clusters and
EN models the specific relationship within each cluster. The model used an online rolling training procedure. Thier fuzzy neural model had 23% and 30% improvement respectively for offline and online schema against a designed NN model.

Among various method to control traffic lights, Choy et al.'s study [56] is one of the well-known research in this area. In this work, a new hybrid, synergistic approach was proposed that applied computational intelligence concepts to implement a cooperative, hierarchical, multiagent system for real-time traffic signal control of a large-scale traffic network. The problem of controlling the network was divided to various subproblems and each handled with an agent by fuzzy neural decision making capability. At the first, the decision were made by lower-level agents and then they were mediated by higher-level agents.

In [56], a multistage online learning process for each agent was implemented that involved reinforcement learning, weight and learning rate adjustment, in addition of dynamic update of fuzzy relations by evolutionary algorithm. The test bed used for evaluation of the proposed method was a section of the Central Business District of Singapore. The result of the experiments illustrated that the performance of the proposed multiagent architecture against the one used for real-time adaptive traffic control system of the moment had significant improvements. It reduced total vehicle stoppage time by 50% and the total mean delay by 40%.

In another work done by Srinivasan et al. [57], the authors presented an enhanced version of the SPSA-NN system for a multi-agent system and they tested that in more complicated scenarios. The authors claimed that although the SPSA algorithms is a useful method for updating the weight online, whereas the model proposed by Spall et al. [3] had some limitations influence its performance.

Spall et al. used a three-layer NN and relevant traffic variables were used as inputs.
2.2 Computational intelligence in adaptive systems

Based on [57], there were two shortcomings for that system: firstly, the system used heuristic methods to identify the general traffic patterns (morning and evening peaks) and assignment of time periods for patterns. This causes the robustness of the system to come into question for non-periodic traffic patterns. Secondly, a NN was considered for each time period, and the weight were updated only whenever the same traffic pattern and time period was arisen. It may not be possible to respond appropriately to changes of the traffic inside the same time period. Srinivasan et al. improved that method and compared it with the hybrid multiagent architecture presented by Choy et al. [56] and Green Link Determining (GLIDE), which was the existing traffic signal control for the city and is the local version of the SCATS. To evaluate the performance they considered a large traffic network in Singapore Central Business District with 25 intersections. After 15 separate simulations with different seeds, which each was set for three hours, the lowest mean delay belonged to SPSA-NN, Hybrid NN and GLIDE respectively.

In the research done by Teodorovi et al. [58], an intelligent isolated intersection control system was developed. Their model was based on the combination of the NNs and dynamic programming. The proposed system makes real time decisions to extend current green time, and calculating the amount of extension required. They conclude from their experimental tests that the outcome of the proposed NN controller is nearly equal to the best solution.

Chao et al. [59] presented an intelligent traffic light control method based on using NN theory for crossroads. First, the number of passing vehicles and passing time of one vehicle within green light time period were measured in the main-line and sub-line of a selected crossroad. During the next step, the measured data are adopted to construct an estimation method based on extension NN for recognizing the traffic flow of a standard crossroad. They claimed their proposed method can discriminate the traffic flow of a standard crossroad rapidly and accurately.

The work done by Nagare and Bhatia [60] was another attempt to forecast traffic flow
for controlling traffic congestion. It was mentioned that NN introduces some flaws such as flow convergence and the obtained solution is usually local optimal. The idea was that by using combined optimization methods better optimization results can be obtained. They applied three different combined optimization methods in their work to have a comparison between them.

NN controllers have ability to consider different ranges of green time and their training times are usually lower than $Q$-learning controllers, but they are not easily adjustable for new situation. Considering appropriate number of layers and neurons in each layer are the other issues related to NN controllers.

Using NN directly as a controller or in combination to other methods for example, as an optimizer have been presented in many works. In some of them both FLS and NN have been used for designing the traffic signal controller, which will be discussed later.

### 2.2.3 Fuzzy logic systems controller for traffic signal timing

The first attempt in applying FLS for controlling traffic signals at a single intersection was done in 1977 by Pappis and Mamdani [61]. Their controller had three inputs and one output. It was designed for a two-phase intersection with random vehicle arrivals and no turning movements. Seven seconds after starting the green time, every 10 seconds the controller decided about extension of the phase or changing that. Actually, the fuzzy rules were developed to evaluate and make decision about the suitable extension of current green phase based on different time duration and by measuring ”degree of confidence”. The extensions were compared with the highest degree of confidence and if none of them had more than 50% of confidence, then the green signal will be terminated immediately. Otherwise, the green time was selected and the process was repeated until maximum acceptable green time is reached.
In the aforementioned work, it was assumed that the vehicle detectors were placed upstream from the intersection to inform the controller about the future vehicles that will arrive, which is useful to predict the future queue length of vehicles at the intersection. To evaluate the system it was compared with the efficient vehicle-actuated method and the result of simulation showed the better performance of FLS controller.

Nakatsuyama et al. [62] applied FLS to control two adjacent intersections with one-way movements. This controller determined the extension or termination of the green signal based on the upstream traffic for the downstream intersection. A FLS controller for freeway ramp metering is developed by Chiu and Chand [63]. The FLS controller presented by Wei et al. [64], was for a set of intersections each of which manages the phase length and sequence dynamically according to its own and neighboring traffic situations.

Favilla et al. [65] also applied FLS to control an isolated intersection with two-way streets. They considered the number of vehicles that had already passed the intersection and the length of the vehicle queue in the red approach as inputs for the fuzzy rules, and the amount of the extension was the output of the FLS. Some additional strategies were considered for adapting the numerical bounds on the input and output as well. Some other studies that proposed FLS controller for a single intersection are [66, 67, 68, 69].

Niittymaki and his colleagues made many contributions in the field of traffic management. They had contributions in simulation area [70, 71, 72, 73, 74], saturation flow in signal-group-controlled traffic signals [75], air quality management [76], prediction of road traffic noise [77], and applying fuzzy controllers in the field of traffic [78, 79, 80, 81, 82, 83, 84, 85, 86, 87]. As a part of Niittymaki and his colleagues’ works we can mention to developing fuzzy rule bases for both choice and sequencing of signal stages to be used [86]. They presented a systematic approach to fuzzy traffic signal control and prepared fuzzy rule based on experts knowledge.
Niittymaki considered the traffic signal programming in two sets: the choice and sequence of signal stages, and the optimization of the relative length of these stages. In this study, the rule bases for both of these problems were introduced and the result of the experiments for the rule bases was promising. Pedestrian friendly signals, separate signals for cyclists, public transport priorities, heavy vehicle priorities, all other priority systems, environmental sensitivity, and general routing aspect were the factors that Niittymaki considered them as effective factors in traffic and rule bases were prepared based on them. In another paper they introduced a Lukasiewicz many-valued logic similarity based fuzzy control algorithm, which had a good statistical results in high density traffic [88].

In the beginning of 90’s, the first application of FLS in a multi-intersection network was published. Chiu and Chand [63] presented a fully distributed system with cooperative local controllers to self-organizing traffic signal control. The parameters of each controller were adjusted by a local controller by considering the local traffic condition and the parameters of the adjacent intersections. For adjusting the standard signal timing parameters a set of fuzzy rules were applied by each local controller. In their proposed system cycle time, phase split and also offset adjustments were considered. Their approach made it possible for the local controllers to have their own cycle time when the coordination is not important.

Lee and Lee-Kwang [89] presented a FLS controller for a group of intersections. Each intersection controlled its own traffic while it had cooperation with its neighbors. The controller used these information and obtained the signal’s optimal time through fuzzy rules.

Each controller had three modules; the green phase observing module, the next phase selection module, and the decision module. The observation module produces the stop degree based on traffic conditions. The stop degree indicates the possibility that the controller should stop the green phase. The next phase module selects one candidate for the
next green phase among all phases except the green phase. It observes the traffic conditions and selects the phase, which is the most urgent among them, and the decision module decides about the time to switch to green phase.

In addition, the controller had capability to manage phase length and phase sequence adaptively to the conditions at the adjacent intersection as well as its own traffic conditions. To test the performance of the controller they developed a simulator for intersection groups. They compared the proposed method with vehicle actuated method under 18 traffic conditions; six traffic plans and three intersection groups. The intersections were two-ways and turning movements were allowed. The proposed method in that research showed better performance for all cases. It showed from 3.5% to 8.4% improvements over the vehicle actuated method in steady traffic conditions. In time-varying conditions, it has improvements from 4.3% to 13.5% were obtained in total average delay time.

Zhang et al. [90] proposed a FLS controller for an over-saturated intersection having two-way streets with left-turning movement. This controller decided on whether to extend or terminate the current green phase. In another work [91], they proposed a two-layer fuzzy control algorithm for traffic control of the network, which is supposed to have large traffic flow and high possibility of congestion.

An FLS controller for an isolated signalized intersection was proposed by Nair and Cai [92]. The aim of the controller was to ensure smooth flow of traffic by reducing the delay time. Most of the FLS controllers attempt to optimize the performance of the network by maximizing traffic flows or minimizing traffic delays under typical traffic conditions. As a result of that, these controllers are not optimal for exceptional traffic cases such as roadblocks and road accidents. In this research, the authors proposed a FLS controller able to control traffic flows under both normal and exceptional traffic conditions. Traffic detector sensors were placed at incoming and outgoing links (lanes) and the controller utilized the information received from them to make near-optimal decisions. They also
developed a simulator to evaluate the performance of traffic controllers under different conditions. Results showed that the performance of their proposed traffic controller was similar to that of conventional FLS controllers under normal traffic conditions and was better than of others under abnormal traffic conditions.

Rahman and Ratrout [93] reviewed FLS controllers in their study. The review covers: applying fuzzy method for two-way single intersection without turning vehicles, single intersection with all possible movements, multiple intersections, phase sequence and time determination, and congested intersection and network. This paper indicated better performance of fuzzy based controller compared to traditional traffic signal controls, specifically during uneven and heavy traffic conditions. Regarding to the similar situations that they recognized in Saudi Arabia, they found the FLS controller as a suitable solution for traffic issue there. They predicted the FLS approach will have a significant contribution in the future approach of transportation management system. In addition, they expected the contribution of FLS controller in the advancement of adaptive traffic signal control by improving the performance of the adaptive controller and the overall decision making process of the transportation management system.

Balaji and Srinvasan [94] and Sabetghadam et al. [95] used type-2 FLS in controlling traffic signals. Non-stationary sensor noise, stochastic nature of drivers behavior, use of rules to control vehicles flow and signals, and use of expert knowledge for mining fuzzy rules from opinions are factors worth to be mentioned to make type-2 FLS more appropriate to be employed in designing such controllers.

In [95], it was mentioned that although computational intelligence based method like NN have been used for designing signal controller, a large training data set with all uncertainties they may contain make it difficult to obtain a proper controller. In their research they developed a multiagent distributed architecture signal control system based on type-2 FLS. All agents were homogeneous and had equal decision making capabilities. An
agent calculated the appropriate green time based on averaged flow rate, queue length, and communicated data from the immediate neighbors, gathered by detectors attached to the intersection. Result of experiments against fixed-time method showed around 40% improvement.

In [94], researchers also used a distributed agent architecture with fuzzy type-2 sets for reducing total delay time. In this study, the proposed method was compared with a hybrid NN based hierarchical multiagent system controller and real time adaptive traffic controller (Glide), which is used in Singapore. They showed that the proposed method had a significant improvement against the benchmarks for both dual and multiple peak traffic scenarios. These authors have other studies in applying multi-agent, NN and FLS in this regard [96], [6], and [16].

Two other studies in this field are [97] and [98]. In the work presented by Wenchen et al. [97], the authors developed two adaptive two-stage fuzzy controllers for traffic signals at isolated intersections. Their controller had online optimization ability. Chiou and Huang [98] proposed a stepwise genetic FLS controller. They considered queue lengths and traffic flows of cars and motorcycles as state variables and extension of green time as control variable. Based on these factors they worked on minimization of total vehicle delays. Through the experimental results they conclude that their proposed signal control model is efficient and robust.

Bi et al. [99], proposed a multi-agent type-2 FLC. In this paper differential evolution is used to optimize the parameters of FLC membership functions and rule base. The network model in this study was composed of eleven intersections. Each intersection was controlled by one separate type-2 FLC and the neighbor intersections communicate with each other. They demonstrated their proposed method can enhance the vehicular throughput rate and reduce delay time and queue length of vehicles.
Experts knowledge about traffic is useful to design a FLS controller. Predefining appropriate membership functions for both inputs and outputs need this knowledge. FLS is a beneficial tool to design a traffic controller as it can describe traffic situations appropriately.

2.3 Conclusions

This chapter reviews different traffic signal timing controllers. All controllers can be categorized in three different groups; fixed-time controllers, actuated controllers, and adaptive controllers. After introducing the conventional traffic control systems the lack of using computational intelligence methods revealed.

In addition, the traffic control systems such as SCATS, MOVA, some other adaptive controllers currently applied in cities usually have problem with the high amount of overhead on the central controller. In addition, the computational cost of data-mining and saturation is considerable.

During last two decades, researchers applying computational intelligence methods in the field of traffic have shown a significant impact to reduce the amount of traffic congestion and travel time. However, by the review done in this chapter and recording to the explanation presented for each part, it is revealed that there is opportunities to explore new techniques based on hybrid computational intelligence, such as NN, FLS, and reinforcement learning. There is also opportunities to improve the controllers and reduce the travel time by using optimization methods, such as SA, GA, and CS, which is a new optimization technique, more explanation about these methods is presented in chapter 4. Considering the possibility of using continues phase extension time is another item that has not been used in previous works. This feature is added to the controllers designed in this thesis.
Chapter 3

Q-learning controller for a single intersection

3.1 Overview

In this section, a Q-learning controller for a single intersection based on one of the recent research in this field by Abdoos et al [1] is designed. Then, we improve this controller and ameliorate some of its deficiencies. Before that a brief related background is presented.

3.2 Machine learning

Machine learning algorithms are broadly categorized in three sets: supervised learning, un-supervised learning and reinforcement learning.

- Supervised learning: In this approach, the algorithm requires a set of desired input-output mapping examples in order to be trained and generate a function to map an input to a desired output. Connectionist learning structures such as neural networks often uses supervised learning to train and update the neural parameters.
• Unsupervised learning: in this learning method, the algorithm tries to find hidden structures in unlabeled data. The labeled training set is not required.

• Reinforcement learning: this is a learning algorithm that learns the right policy according to its observation of the world. Each action has an effect on the environment and the environment provides feedbacks.

The lack of labeled examples and the dynamic nature of most complex applications make supervised learning unsuitable, while the nature of unsupervised learning is not appropriate for dynamic and interactive problems. Reinforcement learning operates based on environmental feedback and is suitable for interactive problems.

3.2.1 Reinforcement learning and Q-learning

Reinforcement learning is a machine learning method that is suitable in unknown environments. In model-free reinforcement learning, the learning algorithm can learn an optimal policy without being able to predict the impact of its actions. In this method, learning progresses based on environmental feedback, the reward or punishment related to the problem.

As a simple term, an agent in reinforcement learning tries to reach a goal through dynamically interacting with its environment. The agent examines different actions in different situations according to the goal and determines the best action or best sequence of actions. For each action the environment provides a feedback useful for the agent to recognize to what extent the action is beneficial to reach the goal [2].

Generally, Markov Decision Process (MDP) [100] is regarded as the mathematical foundation for reinforcement learning. A fully observable MDP is a quadruple $\langle S, A, R, T \rangle$ where $S$ is a finite set of states, $A$ is the set of actions, $T : S \times A \times S \to [0, 1]$ is the state transition function that describes the probability $p(s'|s,a)$ of ending up in state $s'$ when
performing action \( a \) in state \( s \), and \( R : S \times A \rightarrow \mathbb{R} \) is reward function that returns a numeric value after taking action \( a \) in state \( s \). An agent’s policy is a mapping \( \pi : S \rightarrow A \). \( \gamma (0 \leq \gamma \leq 1) \) is the discount factor. The agent aim to find an optimal policy \( \pi \) that maximizes the expected sum of discounted rewards. Eq. 3.1 formulates this definition.

\[
V(s, \pi) = \sum_{t=0}^{\infty} \gamma^t E(r_t|\pi, s_t).
\]  

(3.1)

### 3.2.2 The Q-learning algorithm

Q-learning is an incremental reinforcement learning method. It does not need a model of the environment and learning can be performed online [2]. Fig. 3.1 explains the Interaction of agent and environment in reinforcement learning.

![Fig. 3.1. Interaction of agent and environment in reinforcement learning. Sensory inputs that describe the current state of the environment are received by agent, the agent chooses and executes an appropriate action and receives a reward from the environment.](image)

An agent is an entity that interacts with the environment. It chooses an action based on inputs receives from its sensors and learns on the basis of the effects of its action on the environment. The Q-learning agent at time \( t \) receives a signal from the environment. This signal describes the current state \( s \). Actually, the state is composed of a group of characteristics presenting the current situation of the environment relevant to the problem. State information must have Markov property, meaning that only this information and the
action being taken are needed to predict the environment’s effect. Although it may not be completely true, it is assumed that the process is Markovian. By Markov property the agent does not need to know the history of previous actions or states in its decision making.

In Q-learning the agent selects action $a$ based on the relative value of all possible action in state $s$. This value is the $Q$-value of undertaking action $a$ in state $s$ and lead to transition to state $s'$. This value presented by $Q(s,a)$. The $Q$-value is obtained gradually during the learning and by exploring randomly various possible actions in each state. By performing action $a$ in state $s$, the agent receives reward $r(s,a)$. The obtained reward highly depends on the effect of the action on the environment. During the learning, the agent’s objective is to find the optimal policy that maximize the accumulative reward. Related to the problem, punishment can replace the reward in which the agent aim to minimizes the accumulative punishment over time. One of the other factors that is generally considered in Q-learning is the discount factor $\gamma$ ($0 \leq \gamma \leq 1$). It is applied for bounding the reward, specially in problem domain with continuous episodes, it considers higher value for short term future rewards compared to long term rewards. For updating the $Q$-value in learning process Eq. 3.2 is used.

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r_t + \gamma \max_{a \in A} Q(s_{t+1}, a) - Q(s_t, a_t) \right)$$  \hspace{1cm} (3.2)$$

where $\alpha$ ($0 \leq \alpha \leq 1$) is the learning rate and $\gamma$ ($0 \leq \gamma \leq 1$) is the discount factor. The learning rate needs to be decreased in order to guarantee the convergence of $Q$-function in stochastic environments. In order to formally prove convergence to the optimal policy it is required that each state-action pair is visited an infinite number of times, but in practice the agent will usually converge to the suitable policy as long as each state-action pair visited sufficiently often. For future use of the $Q$-values they can be stored in a $Q$-table, which needs a high amount of memory. Also, it is possible that the $Q$-values are used as the
inputs of a function approximator designed to generalize the $Q$-function. In this case, the approximator can estimate the $Q$-value for not visited state-action pairs by the help of the similar situations. Fig. 3.2 describes the $Q$-learning algorithm step by step.

```
1: Initialize $Q(s,a)$ arbitrarily
2: for all episode do
3:     Initialize $s$
4:     for all step of episode do
5:         Choose $a$ from $s$ using policy derived from $Q$ (e.g., $\epsilon$-greedy)
6:         Take action $a$, observe $r, s'$
7:         $Q(s,a) \leftarrow Q(s,a) + \alpha \left( r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$
8:     $s \leftarrow s'$
9: end for
10: end for
```

Fig. 3.2. The Sutton standard $Q$-learning algorithm [2].

### 3.3 $Q$-learning controller based on Abdoos model [1]

To have a benchmark for future evaluations, Abdoos et al.’s method [1], is implemented for an isolated intersection. Abdoos et al. presented an approach for controlling signal lights in a network of 50 intersections based on $Q$-learning. Each intersection is considered as an agent and the entire system is formed as a multi-agent system. As mentioned in previous section, they improve their model [45], but the basic concept is used for each single intersection was the same in their works. In their method, the average queue length in approaching links defines states of $Q$-learning and the number of permutations of the approaching links form the number of states. Abdoos et al. considered an intersection with four approaching links. Their state space consists of 24 states as shown in Fig. 3.3. In this table $l_i$ represent the lengths of queue in approaching link $i$.

As the actions of $Q$-learning, they considered different phase splits of the cycle time.
3.3 Q-learning controller based on Abdoos model [1]

Fig. 3.3. This table shows the state-space defined in [1].

Phase split refers to the division of the cycle time into a sequence of green signals for each group of approaching links. In addition, Abdoos et al. adjust the cycle time as a fixed value. They set a minimum green time for each phase and the cycle time is divided to a fixed minimum green time and extension time that can be assigned to each phases. The action space was defined by \( <n_p h, t_{min}, n_e x, h_e x > \), where \( n_p h \): number of phases, \( t_{min} \): minimum green time for phases (seconds), \( n_e x \): number of extensions, \( h_e x \): length of each extension (seconds). The efficient cycle length \( \delta \) is then calculated as follows:

\[
\delta = n_p h \times t_{min} + n_e x \times h_e x
\]  

In their configuration, the number of phases is four and the minimum green time for each phase is set to 13 seconds. In addition, four 10-second extension intervals are used for signal timing, it means \( n_e x = 4 \) and \( h_e x = 10 \). In this case the effective cycle time is \( \delta = 92 \) seconds.

All red intervals, which provide a safe transition between two conflicting traffic signal

<table>
<thead>
<tr>
<th>State number</th>
<th>Link order</th>
<th>State number</th>
<th>Link order</th>
</tr>
</thead>
<tbody>
<tr>
<td>State 1</td>
<td>( l_1 \geq l_2 \geq l_3 \geq l_4 )</td>
<td>State 13</td>
<td>( l_2 \geq l_3 \geq l_1 \geq l_4 )</td>
</tr>
<tr>
<td>State 2</td>
<td>( l_1 \geq l_2 \geq l_4 \geq l_3 )</td>
<td>State 14</td>
<td>( l_2 \geq l_4 \geq l_1 \geq l_3 )</td>
</tr>
<tr>
<td>State 3</td>
<td>( l_1 \geq l_3 \geq l_2 \geq l_4 )</td>
<td>State 15</td>
<td>( l_3 \geq l_2 \geq l_1 \geq l_4 )</td>
</tr>
<tr>
<td>State 4</td>
<td>( l_1 \geq l_4 \geq l_2 \geq l_3 )</td>
<td>State 16</td>
<td>( l_4 \geq l_2 \geq l_1 \geq l_3 )</td>
</tr>
<tr>
<td>State 5</td>
<td>( l_1 \geq l_3 \geq l_4 \geq l_2 )</td>
<td>State 17</td>
<td>( l_3 \geq l_4 \geq l_1 \geq l_2 )</td>
</tr>
<tr>
<td>State 6</td>
<td>( l_1 \geq l_4 \geq l_3 \geq l_2 )</td>
<td>State 18</td>
<td>( l_4 \geq l_3 \geq l_1 \geq l_2 )</td>
</tr>
<tr>
<td>State 7</td>
<td>( l_2 \geq l_1 \geq l_3 \geq l_4 )</td>
<td>State 19</td>
<td>( l_2 \geq l_3 \geq l_1 \geq l_4 )</td>
</tr>
<tr>
<td>State 8</td>
<td>( l_2 \geq l_1 \geq l_4 \geq l_3 )</td>
<td>State 20</td>
<td>( l_2 \geq l_4 \geq l_3 \geq l_1 )</td>
</tr>
<tr>
<td>State 9</td>
<td>( l_3 \geq l_1 \geq l_2 \geq l_4 )</td>
<td>State 21</td>
<td>( l_3 \geq l_2 \geq l_4 \geq l_1 )</td>
</tr>
<tr>
<td>State 10</td>
<td>( l_4 \geq l_1 \geq l_2 \geq l_3 )</td>
<td>State 22</td>
<td>( l_4 \geq l_2 \geq l_3 \geq l_1 )</td>
</tr>
<tr>
<td>State 11</td>
<td>( l_3 \geq l_1 \geq l_4 \geq l_2 )</td>
<td>State 23</td>
<td>( l_3 \geq l_4 \geq l_2 \geq l_1 )</td>
</tr>
<tr>
<td>State 12</td>
<td>( l_4 \geq l_1 \geq l_3 \geq l_2 )</td>
<td>State 24</td>
<td>( l_4 \geq l_3 \geq l_2 \geq l_1 )</td>
</tr>
</tbody>
</table>
phases, are set to two seconds. Since there are four phases, 8 seconds are assigned for red time intervals. In this case, the total cycle time will be 100 seconds for a complete signal plan. By considering the maximum number of extension as two, there are 19 possible number of actions in their work.

In addition, reward is inversely proportional to the average lengths of the queues in the approaching links, normalized to remain between 0 and 1.

\[
\text{Reward} = \frac{1}{\text{mean}(\sum_{i=1}^{n} \text{LQ}_i) + 1} \quad (3.4)
\]

where \(i = 1, \ldots, n\) is the number of approaching links, \(\text{LQ}\) is the length of queue in an approaching link, and +1 is to refuse zero in denominator.

### 3.4 Improved Q-learning controller for an isolated intersection

For the second step, it is aimed to improve Abdoos et al.’s method by changing parameters used for decision-making. As mentioned in the previous section, 24 states are considered for controlling the traffic lights. One point for improvement is the way the state-space is defined. In Abdoos work, there is no separate state for approaching links with equal length of queues. For example, in their list of 24 states there is no difference between \(l_1 > l_2 > l_3 > l_4\) and \(l_1 = l_2 = l_3 = l_4\) or \(l_1 = l_2 = l_3 > l_4\). However, these cases cause different situations in the traffic network. A second item is that in their method there is just a comparison between length of queues that forms different states. Measurement for the proportion of this difference has not been introduced. There are the same states for cases that have two links with a small difference in the size of queues and the ones with a large
difference in the queue lengths. This point will be clarified more here. Let $Q_i$ be the queue length in the $i_{th}$ lane ($i = 1, 2, 4$). Consider the case that these two situations of queues exist: $(Q_1 : 4, Q_2 : 5, Q_3 : 2, Q_4 : 3)$ and $(Q_1 : 40, Q_2 : 50, Q_3 : 20, Q_4 : 30)$. The same states are considered for both and similar green times will be chosen. However, it is not reasonable to assign similar green times for these two cases, as the effect of queue length has been completely ignored.

To overcome these shortages and make the learning more accurate, all queues lengths are categorized into three ranges: low, medium and high. Different combinations of these values for approaching links define different states. For example, for an intersection with $k$ approaching links there will be $3^k$ members in the state space. The state space in this method is larger than the one proposed by Abdoos et al., but it is more accurate.

Action sets are a combination of green times for each phase. The cycle time is flexible based on the traffic demand and is not necessarily fixed. For an intersection with four phases, some possible actions are: \{13, 13, 13\}, \{23, 23, 33\}, or \{33, 13, 33\} in which each member of the sets are the green time for related defined phase. It is useful to mention that the numbers are chosen in a way to make the action sets similar to Abdoos et al.’s in order to have a reasonable comparison.

Abdoos et al. used a reward that was inversely proportional to the average length of the queues in the approaching links, normalized to remain between 0 and 1. However, normalizing to remain between 0 and 1 causes each action, even beneficial or regretful, to have a positive reward. It can cause delay in the learning of a $Q$-table, which is reset to zero when starting learning. For improving the reward, the distribution of cars and the number of cars in each cycle is considered in calculation. The distribution of cars varies on an hourly and daily basis. For this reason, traffic queue length also fluctuates over time.

In this thesis, length of traffic queues, number of cars that come to the intersection, are considered as reward. Both negative and positive reward is considered during learning. The
reward is the average difference between the number of cars entering to the intersection in current cycle and length of queues (number of ) in related approach links at the end of the cycle time. Positive reward is obtained when the number of cars that entered the intersection is greater than the final queue length that means most of the entered cars or all of them cross the intersection. The reward is negative for the time that the number of cars that entered to the intersection is less than the final queue length, meaning that the proposed green times for each phase has been insufficient for the cars to cross the intersection.

\[
\text{Reward} = \text{mean} \left( \sum_{i=1}^{n} (N_{in} - N_{remained}) \right)
\]  
(3.5)

where \(i = 1, \ldots, n\) is the number of approaching links, \(N_{in}\) is the number of cars entering to the intersection in current cycle and \(N_{remained}\) is the number of vehicles in related approach links at the end of the cycle time.

The process of interaction between the \(Q\)-learning controller and PARAMICS is presented in Fig. 3.4. The detected queue lengths from the environment (here this information is obtained from PARAMICS as a simulator) are sent to \(Q\)-learning controller and the appropriate green time for each phase are proposed. The proposed green times are selected from the predefined action list in \(Q\)-learning method [101].

\subsection*{3.5 Experimental results and discussion}

To evaluate the proposed methods, it is required to simulate a traffic model. PARAMICS version 6.0 has been chosen as the simulator. PARAMICS is a microscopic traffic simulation developed by Quadstone Ltd [102]. PARAMICS allows the traffic process to be simulated to the level of the individual vehicles. The geometry of traffic, amount of traffic and the maximum capacity of the traffic network can be simulated in this software. In this
3.5 Experimental results and discussion

Fig. 3.4. This figure shows the learning process during each traffic cycle in a Q-learning controller. PARAMICS sends the information such as the number of cars and vehicles queue length in each lane in the current cycle to the controller. The controller computes and sends the appropriate green time for each phase of the next cycle to PARAMICS.

research, the control algorithms are implemented in Matlab. PARAMICS loads the controller and sends it the traffic parameters including flow, queue, number of cars, and receives the control signals for each simulation cycle.

3.5.1 Introducing the testbed and conditions

In order to evaluating the experiments, an intersection with four approaching links and four phases (A, B, C, D) is considered, as shown in Fig. 3.5. The cycle times are divided between these four phases. Zones are the areas that vehicles are released into the intersection. Fig. 3.6 and Fig. 3.7, are snapshots of the designed intersection in PARAMICS. Here, four different entrances to the intersection create four zones.

The simulation is modelled in PARAMICS version 6.0. and all controllers are implemented in Matlab R2011b.

For testing the proposed methods, two benchmarks are considered; one is the fixed-time
method and the second one is based on [1].

**Fig. 3.5.** An intersection with four possible phases: A, B, C, D. Four zones are also specified in the figure. Zone1, Zone2, Zone3, and Zone4.

**Fig. 3.6.** Snapshot of the isolated intersection in PARAMICS.
3.5 Experimental results and discussion

3.5.2 Experimental setting for $Q$-learning traffic signal timing controllers

Two scenarios are considered for evaluating the performance of the proposed method. For the first one, 4,990 vehicles enter to the intersection from all four zones (peak time), and in the second scenario 2,440 is the total number of vehicles (off-peak time). In both cases, the simulation time is set to 10 hours. Table 3.1 and 3.2, show the details of simulation parameters respectively. To make results statistically meaningful and to minimize the effects of random number generators in the simulations, all experiments are repeated ten times and average results are reported. During the evaluation, ten different seeds are set in PARAMICS for each scenario, and all three methods are evaluated in a similar condition.
3.5 Experimental results and discussion

Table 3.1
Demand of cars for each zone in first experiments.

<table>
<thead>
<tr>
<th>Zone</th>
<th>Zone1</th>
<th>Zone2</th>
<th>Zone3</th>
<th>Zone4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zone1</td>
<td>-</td>
<td>1,000</td>
<td>900</td>
<td>0</td>
<td>1,900</td>
</tr>
<tr>
<td>Zone2</td>
<td>400</td>
<td>-</td>
<td>0</td>
<td>100</td>
<td>500</td>
</tr>
<tr>
<td>Zone3</td>
<td>0</td>
<td>1,000</td>
<td>-</td>
<td>300</td>
<td>1,300</td>
</tr>
<tr>
<td>Zone4</td>
<td>500</td>
<td>0</td>
<td>700</td>
<td>-</td>
<td>1,200</td>
</tr>
<tr>
<td>Total</td>
<td>900</td>
<td>2,000</td>
<td>1,600</td>
<td>400</td>
<td>4,990</td>
</tr>
</tbody>
</table>

Table 3.2
Demand of cars for each zone in second experiments.

<table>
<thead>
<tr>
<th>Zone</th>
<th>Zone1</th>
<th>Zone2</th>
<th>Zone3</th>
<th>Zone4</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zone1</td>
<td>-</td>
<td>400</td>
<td>200</td>
<td>0</td>
<td>600</td>
</tr>
<tr>
<td>Zone2</td>
<td>400</td>
<td>-</td>
<td>0</td>
<td>180</td>
<td>580</td>
</tr>
<tr>
<td>Zone3</td>
<td>0</td>
<td>300</td>
<td>-</td>
<td>320</td>
<td>620</td>
</tr>
<tr>
<td>Zone4</td>
<td>340</td>
<td>0</td>
<td>300</td>
<td>-</td>
<td>640</td>
</tr>
<tr>
<td>Total</td>
<td>740</td>
<td>700</td>
<td>500</td>
<td>500</td>
<td>2,440</td>
</tr>
</tbody>
</table>

3.5.3 A comparison of Abdoos method, proposed Simple Agent Q-learning (SAQL) method and fixed-time method

For implementing Q-learning based on [1], 19 fixed actions were considered. It was assumed that the cycle time is fixed and the minimum green time is set to 13 seconds. The possible green times were \{13, 23, 33\}. All combination of these three numbers are not considered in the action list due to limitation of the fixed cycle time, which was set to 100 seconds, and also the limitation on the number of extensions for each phase, which was considered as two times and each time 10 seconds.
In the experiments performed here, three fixed numbers as possible green times is considered for SAQL to make the conditions close to what Abdoos have done, but the number of actions is extend from 19 actions to 81 possible actions by considering all combination of these three numbers and having flexible cycle times. The flexibility of cycle time is useful to set the traffic signal lights based on the traffic conditions. Length of a cycle is variable and is obtained from the sum of green times of all phases plus eight. The later one comes from two seconds safety time required after each phase for clearing. For instance, cycle time for a set like \{13, 13, 23, 13\} is 70 and for this set \{33, 33, 33, 13\} is 120.

The rewards are calculated based on the explanation in the previous section. For both learning methods (This method and Abdoos method), $\epsilon$-greedy is applied to select an action in Q-learning. $\epsilon$ is set to 0.1 in our experiments, that means the best action is chosen 90% of times and an action is selected randomly in just 10% of cases. $\gamma = 0.9$ and $\alpha = 0.1$ are the other parameters used during learning for both methods.

Table 3.1, shows the traffic demands used in testing the three methods: fixed-time method, Abdoos method and the proposed method (SAQL). The results are compared after completing learning for Abdoos and SAQL methods. The same settings in PARAMICS are applied for all three methods. During the first experiment, 4,990 cars enter the intersection in 10 hours (scenario 1). The fixed green time for all phases in the fixed-time method is set to 23 seconds, which is the average of three proposed green time numbers by Abdoos (13, 23, 33).

Fig. 3.8, represents the average delay time for each method during these 10 hours. Cumulative average delay in each cycle of ten hour simulation for fixed-time, Abdoos, and SAQL methods is 49,129 seconds, 49,806 seconds, and 36,275 respectively. These figures indicate that application of the proposed method improves the traffic control process by 26.2% and 27.2% compared to fixed-time and Abdoos methods respectively.

In the second experiment, an intersection with 2,440 cars in 10 hours is considered
3.5 Experimental results and discussion

Fig. 3.8. Comparison of the results of fixed-time, Abdoos and SAQL. Cumulative average delay time in seconds during 10 hours (600 minutes) for each method in first experiments is presented.

Table 3.3
Review of results for two scenarios in 10 hours of simulation. Results show the percentage of improvement of SAQL against fixed-time and Abdoos’s method. These results are the average of ten times tests for each method.

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Fixed-time</th>
<th>Abdoos</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>26.2%</td>
<td>27.2%</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>23.1%</td>
<td>28.8%</td>
</tr>
</tbody>
</table>

(scenario 2). Table. 3.2, shows the demands in this experiment. As before, the performance of the three methods is quantitatively measured and examined. The same settings are considered for the fixed-time method. Simulations are run for 10 hours. Fig. 3.9, represents a comparison of the average delay time for each method during these 10 hours. Cumulative average delay in this simulation for the fixed-time, Abdoos et al.’s, and proposed methods are 29,588 seconds, 31,952 seconds, and 22,748 seconds. Similar to the previous experiment, SAQL outperforms the fixed-time and Abdoos et al.’s methods by 23.1% and 28.8% respectively.
3.5 Experimental results and discussion

Although Abdoos’s method is useful for many cases presented in their work, we improve its effectiveness for situations in which it may not be suitable. Aforementioned experiments are performed for such cases. In the work presented by Abdoos et al. [1], better results for their method in comparison to the fixed method were presented, however, our experiments show that better results are related to the traffic demand. During our experiments we considered situations revealing that there are cases in which we can obtain better results through applying fixed-time instead of Abdoos’s method. Considering fixed cycle time causes limitation for allocating the most appropriate green time for related traffic demand by reducing the number of possible action from action list. In this work, we reduce this limitation by avoiding the limitation of fixed cycle time and extending the action sets in Q-learning.
3.6 Conclusions

In this chapter, a benchmark $Q$-learning controller based on [1] is implemented. In addition, the improved version of the $Q$-learning controller is presented in this section named SAQL. Abdoos’s method is applied to one intersection and we add additional details in implementing our method. Both methods use $Q$-learning, which is a promising approach to control traffic lights and supports flexible rather than fixed cycle times. Our improvement is obtained by considering various state-space and reward functions. Evaluation of the performance of the controllers are done in a single intersection traffic model designed in PARAMICS. Results of experiments show higher performance of the proposed $Q$-learning controller methods compared to benchmark $Q$-learning controller and fixed-time controller.
Chapter 4

Intelligent controllers for a single intersection

4.1 Overview

It should be noted that in most of the previous reviewed studies presented in chapter 2, the timing of a signal is done by extending or terminating the current traffic signal phase. In this case, it is not possible to have an estimation of the end of a phase at the start of that phase, therefore, traffic signals with timer are not suitable for them. Furthermore, usually there are fixed (discrete) rather than continuous extension times. Our proposed intelligent controllers are flexible and can produce different range of values as a traffic signal phase duration. These numbers are determined at the start of each phase. This option gives the opportunity of using timer signal controller that is useful for drivers to know how long they have to stay at stop lights.

During last chapter $Q$-learning controller is presented. One of the limitation in $Q$-learning controller is deficiency in providing different range of values as green time. This chapter moves from discrete to continues green time and includes three parts. During each part, different techniques are used in design of the traffic controllers.
4.2 Related background

4.2.1 Neural network

The basic concept of NN were originally obtained from the way biological nervous systems work. A NN is an information processing technique well-known because of its excellent approximation and learning capabilities. A NN is a universal approximator that for any nonlinear mapping can approximate various degree of accuracy [103], using this feature NNs are able to recognize hidden patterns from imprecise and complicated data. In a better word for a problem that is too complicated to be considered by either traditional data mining methods or humans, NN can be a suitable option. In different fields of engineering and science, NNs have been broadly used for control, modeling, prediction and classification problems.

In the case of supervised learning, NNs are usually trained by minimizing an error-based cost function. The parameters of the NN can be optimally adjusted for situations with unknown expected values, by the using and minimizing the cost function. To obtain the optimal set of parameters, the global optimization methods such as simulated annealing (SA) [104] or genetic algorithm (GA) [105], are suitable.

4.2.2 Fuzzy logic systems (type-1 and type-2)

In early 1975, fuzzy set theory was proposed by Zadeh [106, 107, 108]. FLS is a suitable method to represent the vagueness and uncertainties of the linguistic phrases. Actually, it is possible to handle inexact data and uncertain information by fuzzy sets. Using fuzzy theory instead of crisp set theory provides the ability to implement the real-world scenarios in more details. One of the most important feature of FLS is the ability to include an expert’s knowledge in their design. Additionally, they are transparent, which makes them
more understandable by operators compared to black box NN models. A FLS maps the inputs to the output of the system. In situation of no fuzziness in the definition of a cluster or class of objects, there is just a simple two-valued characteristic function, zero and one, but by fuzzy set this domain is extended to the range of whole numbers between zero and one. To represent some linguistic values the system domain is divided in fuzzy sets, for example we can define low, medium, and high traffic flow. Then, membership functions are used to show the degree of dependency to each fuzzy set. Each input value may belong to more than one fuzzy set. Similar situation can be considered about the output space. Associating numerical values to fuzzy sets is fuzzification and defuzzification is the name of the opposite process. The logic of the system is define by if-then rules in fuzzy inference.

In 2001, Mendel argued that type-1 FLSs (T1-FLSs) do not have the ability to directly handle rule uncertainties [109]. As an extension to the concept of ordinary fuzzy sets, Zadeh introduced type-2 fuzzy sets [106]. The membership functions used in type-2 fuzzy sets are also fuzzy. This fact makes it possible for a type-2 FLS (T2-FLS) to model and handle uncertainties of measurements and rules.

Wu and Mendel [110] summarized the most appropriate situations for applying T2-FLS as:

- Non-stationary noise associated with the sensor measurements. This noise cannot be fully expressed mathematically.

- A stochastic data generating mechanism that cannot be correctly approximated by mathematical distribution functions like Gaussian or Poisson distribution.

- The knowledge base used to construct the rule base for the FLS is mined from a series of if-then questionnaires put forward to experts.
4.2 Related background

**T2-FLS and IT2-FLS**

General T2-FLSs are computationally demanding due to their inference and type reduction components. To alleviate this problem, interval T2-FLS (IT2-FLS) are introduced a decade ago [111]. The secondary membership function in IT2-FLS is either zero or one instead of including the whole range between zero and one. Both T2-FLS and IT2-FLS consist of five parts: fuzzifier, inference engine, rule base, type-reducer, and defuzzifier. Fig. 4.1 shows the structure of the T2-FLS.

**Fuzzifier:** This part maps the crisp inputs to fuzzy sets.

**Rule base:** Rule base includes expert knowledge and is expressed in the form of ”if-then”. Mamdani and Takagi- Sugeno-Kang (TSK) [112] are two types of fuzzy inference systems. Regarding to these two types, two different kinds of rule definition exist:

\[
\text{Mamdani Rule : If } x_1 \text{ is } F_1 \text{ and } x_2 \text{ is } F_2, \ldots, x_n \text{ is } F_n \text{ then } y \text{ is } G \quad (4.1)
\]
4.2 Related background

**TSK Rule:** If $x_1$ is $F_1$ and $x_2$ is $F_2$, ..., $x_n$ is $F_n$ then $y = f(x_1, x_2, ..., x_n)$ \hspace{1cm} (4.2)

In Eq.4.3 $G$ is a fuzzy function and in Eq.4.4 $f$ is a linear function.

**Fuzzy inference:** Fuzzy inference is a key part in FLS process. It has the capability of simulating human decision making by performing approximate reasoning to achieve a desired control strategy.

**Type-reducer and defuzzifier:** After the process of fuzzy inference the result is T2-FLS or IT2-FLS. Type-reducer convert the type-2 output to type-1 output. The crisp value is the result of defuzzifier part.

### 4.2.3 Adaptive neuro-fuzzy inference system (type-1 and type-2)

Adaptive Neuro-Fuzzy Inference System (ANFIS) [113] is made of the combination of FIS and NN. In ANFIS, the parameters of FIS are tuned by applying neural learning rules. Back propagation is the method that usually have been applied during training by NN. ANFIS is based on first order Sugeno FIS that adaptively learns and modifies the rules of the system. More explanation is available in the study done by Jang [113]. An example of a simple ANFIS structure with two inputs and two membership functions for each input is presented in Fig. 4.2. In this figure, circle is for a fixed node and square indicates an adaptive node. It is assumed that the examined FIS has two inputs and one output. Typical rule set for this system can be expressed as follows:

**Rule 1:** If $x$ is $A_1$ and $y$ is $B_1$, then $f_1(x) = p_1 x + q_1 y + r_1$ \hspace{1cm} (4.3)

**Rule 2:** If $x$ is $A_2$ and $y$ is $B_2$, then $f_2(x) = p_2 x + q_2 y + r_2$ \hspace{1cm} (4.4)

where $x$ and $y$ are the crisp inputs, and $A_i$ and $B_i$ (for $i = 1, 2$) are the linguistic labels
The ANFIS model presented in Fig. 4.2 has five layers. Description about each layer is provided below:

**Layer 1**: Every node $i$ in this layer is an adaptive node with a node function such as Eq. 4.5. ($O_{l,i}$ is the output of the $i$th node of layer $l$).

$$O_{1,i} = \mu A_i(x), \quad i = 1, 2 \tag{4.5}$$

where $x$ is the input to node $i$, and $A_i$ is the linguistic label, including small, medium, large, related to the current node function. In better words, $O_{1,i}$ is the membership grade for each fuzzy set member ($A_1, A_2, B_1, B_2$).

**Layer 2**: The outputs of this layer represent the firing strength of each rule. A minimum value of two input weights is selected in this layer.

$$O_{2,i} = \mu A_i(x) \times \mu B_i(x), \quad i = 1, 2 \tag{4.6}$$
Layer 3: The outputs of this layer are called normalized firing strengths. The \( i \)th node calculates the ratio of the \( i \)th rules’ firing strength to the sum of all rule’s firing strengths.

\[
O_{3,i} = \bar{W}_i = \frac{w_i}{w_1 + w_2}, \quad i = 1, 2
\]  

(4.7)

Layer 4: Linear functions of input signals are calculated in this layer. Each node \( i \) in this layer is an adaptive node.

\[
O_{4,i} = \bar{W}_i f_i = \bar{W}_i (p_i x + q_i y + r_i), \quad i = 1, 2
\]  

(4.8)

where \( \bar{W}_i \) is the normalized firing strength of layer 3, \( p_i, q_i, r_i \) are the parameters of this layer and referred as consequent parameters.

Layer 5: This layer is made of a single node labeled sum. The output of this layer is the overall output and summing all incoming signals.

\[
\text{overall output} = O_{5,i} = \sum_i \bar{W}_i f_i = \frac{\sum_i W_i f_i}{\sum_i W_i}
\]  

(4.9)


There are some concerns regarding the design of type-1 ANFIS that still prevail in the case of type-2 ANFIS, including selecting the operator to use for intersection and union, the format of the rules and defuzzification method. The type-2 ANFIS structured designed by John and Czarnecki has an additional layer as compare to type-1 ANFIS. This extra layer is for defuzzification. Based on John and Czarnecki [114]’s research, type reduction process described by Karnik and Mendel [115] is adopted for proposed type-2 ANFIS structure. This process is applied to reduce the type-2 set to a type-1 set and then defuzzify that set in the normal way to produce a crisp value. The obtained crisp value will be compared with
the target output and the error will be calculated for back propagating through the network for parameter modification. In practice, the output of each layer is a type-2 set and the last layer does the type reduction. An example of type-2 ANFIS with two inputs is presented in Fig. 4.3.

### 4.2.4 Genetic algorithm

Genetic Algorithm (GA) has been theoretically and empirically shown to offer consistent performances in complex search spaces [116]. The philosophies of the GA derive from natural genetic processes. It involves terms such as gene, chromosomes, offspring, generation, crossover, and mutation. The GA begins with a randomly generated population of chromosomes as the initial solution. A fitness or evaluation function is considered to calculate the objective value of a given problem. The algorithm progresses by selecting some chromosomes according to certain selection criteria and rejecting the remaining solutions. Subsequently, GA develops offspring by employing a reproduction process. This process
4.2 Related background

named crossover and mutation that recombine and develop a new generation. Among the
process a number of generations are iterated until the algorithm converges to the best solu-
tion, or reaches the stopping criterion specified for the given problem.

The GA has different characteristics as compared with other optimization techniques
[117, 118, 119, 120, 121, 122, 123, 124, 125, 126]. Firstly, the GA performs a stochastic
search, rather than a deterministic search. This is an effective way in finding the optimum
solution in most of complex systems. Secondly, the GA considers different points in the
search space simultaneously that increases the chance of finding the global optimum solu-
tion, instead of trapping in a local one. Thirdly, the GA does not need information about
the structure or parameters of the problem [117]. Therefore, the efficiency and robustness
of the GA provides a good opportunity to apply this method as an optimization tool for
parameter tuning of the input and output fuzzy sets.

Pseudo code for GA presented in Fig. 4.4.

1: begin
2: Generate initial population of individuals
3: Evaluate the fitness of the individuals
4: while (termination criteria) do
5: Select the best individuals to be used by GA operators
6: Evaluate the fitness of new individuals
7: Replace the worst individuals by the best new individuals
8: end while
9: Postprocess results and visualization
10: end

Fig. 4.4. Pseudo code for standard GA algorithm.

4.2.5 Simulated annealing

Simulated Annealing (SA) is a universal probabilistic metaheuristic method for the global
optimization problem in a large search space. This method is often used when the search
space is discrete. SA was independently described by Kirkpatrick, Gelatt and Vecchi in 1983 [127] and by Cerny in 1985 [128].

The name and inspiration of SA come from annealing in metallurgy. Annealing is a technique containing heating and controlled cooling of a material lead to increase the size of its crystals and reduce their deficiencies.

The idea of slow cooling is applied in the SA algorithm as a slow decrease in the probability of accepting worse solutions while exploring the solution space. SA improves an initial solution iteratively by selecting a neighbour solution and comparing according to a fitness function. By this strategy, it finds the best solution over time and this process repeats until convergence. Transition to poorer solutions is allowed with the purpose of avoiding getting stuck in local optimums or flat regions. SA has a number of advantages in comparison to traditional mathematical optimization techniques. For example, it can be used for optimization of any cost function, either continuous or discontinuous. The complexity and dimensionality of the cost function does not place any restriction on SA performance in finding the globally optimal solution. More description about SA can be found in [129].

Pseudo code for SA is presented in Fig. 4.5. SA starts from a state $s_0$ and continues to reach a maximum of $k_{max}$ steps or until a state with an energy of $e_{min}$ or less is found. During the process, the neighbor $(s)$ randomly generates a neighbor of a given state $s$. The random value in the range $[0, 1)$ is chosen and the annealing schedule is defined by the temperature. $E()$ is the energy (goal) function and the acceptance probability function is $P()$. 
4.2 Related background

4.2.6 Cuckoo search optimization algorithm via Lévy flight

Cuckoo search (CS) is a new heuristic search algorithm. In this method three different rules are considered:

First: Each cuckoo lays an egg, choose a host nest and dispose it there.

Second: The best nest, nest with higher quality of eggs, will carry to the next generation.

Third: The number of host nests are fixed. The probability of egg-discovery by host bird is $p_a \in [0, 1]$. The host bird in cuckoo search can throw out the parasitic egg or abandon that nest for a better new nest. In fact, $p_a$ is the probability of substituting the nest by new nest, which are the new random solutions.

Pseudo code for CS is presented in Fig. 4.6 [130].

During generating a new solutions $X^{(i+1)}$ for the $i$th cuckoo the lévy flight is performed.

$$\left(X_i\right)^{(t+1)} = X_i^{(t)} + \alpha \oplus \text{Lévy}(\lambda)$$

where $\alpha > 0$ is the step size. This parameter related to the scales of the problem of interests. Most of the time $\alpha$ is considered as unity. The aforementioned equation is the stochastic equation for random walk. Generally, random walk is Markov Chain and its next location
only depends on its current location. In Eq. 4.10, the symbol $\oplus$ is entrywise multiplication (exclusive OR). The entrywise product in CS is similar to those applied in Particle Swarm Optimization (PSO) algorithm. However, the random walk via Lévy flight is superior as its step length is much longer in the long run. Applying random walk by Lévy flight is for the time that the random step length is drawn from a lévy distribution. The distribution has an infinite variance and mean, it also has a power-law step size of a heavy tail Eq. 4.11.

$$Lévy \sim u = t^{-\lambda}, \quad (1 < \lambda < 3), \quad (4.11)$$

To speed up the local search, Lévy walk should generate a substantial of the new solution around the best solution obtained so far. In addition, some of the new solutions should be generated far enough from the current solution. This group will be generated by far field randomization to assurance of not be trapped in a local optimum.

Based on Yang and Deb’ claim [130], in a quick look there are some similarities
between CS and hill-climbing compared to some large scale randomization. However, there are significant differences: First of all, CS is a population based algorithm similar to GA and PSO, while it uses some sort of elitism and/or selection similar to harmony search. The second point is that CS is more efficient in randomization. In CS the step length is heavy tailed and any large step is possible. The third fact is in the number of parameters need to be tuned, which are less than GA and PSO. This factor make CS more appropriate for wider class of optimization problems.

4.3 Part 1: NN and FLS controllers

In this section, two intelligent controllers for a single intersection are introduced. One of these controllers is designed based on NN and the other one is based on FLS. To reach optimal controller we use GA optimization methods.

4.3.1 Design of the NN controller

The developed NN controller uses a feed-forward network. It is GA-based NN (GA-NN), which uses GA to find the optimal parameters for NN controller. It consists of four input neurons, ten neurons in the hidden layer, and four neurons in the output layer. In every cycle the length of queues from PARAMICS are fed to the NN and the proposed green times for each cycle are generated and provided to PARAMICS. After each period of simulation, when a model simulation time is finished, the total average delay time, from the first cycle to the last cycle, is calculated and sent to Matlab as the cost function of the optimization process. Based on the cost function and by using GA optimization method, new weights for the NN are generated and weights are updated accordingly. This process is repeated until there is no further improvement for several iterations or after reaching a maximum
number of iteration set in GA options. Upon termination, the NN parameters are set to the optimal set of weights.

It is important to note that the NN model is indirectly trained here, as the desired targets, which are green times in this problem, are unknown during training. For each generation, new populations with a new set of parameters are developed for the NN model. PARAMICS uses these temporary NN models as the brain for controlling traffic lights. At the end of each simulation, it returns the calculated average delay per vehicle as the cost function to Matlab. The decision on whether to accept or discard the current solution (the NN set of weights) is made in the GA optimization method. This process allows for the optimal adjustment of the NN parameters even if the desired targets are unavailable or unknown. Fig. 4.7 shows the training process.
4.3.2 Design of the FLS controller

The FLS controller is also a GA based controller (GA-FLS). FLS controller obtain its optimal parameters after training. Length of queues at approaching links are fed as inputs of the FLS controller. The output of the FLS controller is the proposed green time for each phase. Therefore, for an intersection with four approaching links we need four FLS controllers each for estimating the appropriate green time for related link. Membership functions for all inputs and the output of the FLS controllers are considered identical. Each input and the output has three membership functions named small, medium, large. The membership functions are Gaussian whose sigma and mean are optimized during training. Design of the FLS controller is similar to the NN controller in Fig.4.7.

Eleven rules are defined for the FLS controller. Queue length of vehicles at current link (CL), next link (NL), second next link (2NL), and third next link (3NL) are the factors considered in the rule base definition. For example, we define the second rule as this:

If CL is medium, NL is small, 2NL is small, and 3NL is small then Green Time is medium.

Table 4.1 shows the rule base used for FLS controller.

4.3.3 Fixed-time controller

Fixed-time controller or pre-timed controller is usually used as a benchmark for evaluating the performance of proposed controllers. In fixed-time controller a constant amount of time is set for each phase. The constant pre-defined time for each phase causes the least flexibility for fixed-time controller to adapt traffic demands. A simple structure is considered for fixed-time controller. In this structure we set equal time for all green phases.
Table 4.1
FLS controller Rule Base. In this table S stands for small, M for medium, L for large, and \( \sim \) is for negation (e.g., \( \sim S \) means not-small, which could be either medium or large).

<table>
<thead>
<tr>
<th></th>
<th>CL</th>
<th>NL</th>
<th>2NL</th>
<th>3NL</th>
<th>Green Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>S/M/L</td>
<td>S/M/L</td>
<td>S/M/L</td>
<td>S</td>
</tr>
<tr>
<td>2</td>
<td>M</td>
<td>S</td>
<td>S</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>3</td>
<td>M</td>
<td>( \sim S )</td>
<td>( \sim S )</td>
<td>( \sim S )</td>
<td>S</td>
</tr>
<tr>
<td>4</td>
<td>L</td>
<td>( \sim L )</td>
<td>( \sim L )</td>
<td>( \sim L )</td>
<td>L</td>
</tr>
<tr>
<td>5</td>
<td>L</td>
<td>L</td>
<td>( \sim L )</td>
<td>( \sim L )</td>
<td>M</td>
</tr>
<tr>
<td>6</td>
<td>L</td>
<td>( \sim L )</td>
<td>L</td>
<td>( \sim L )</td>
<td>M</td>
</tr>
<tr>
<td>7</td>
<td>L</td>
<td>( \sim L )</td>
<td>( \sim L )</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>8</td>
<td>L</td>
<td>( \sim L )</td>
<td>L</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>9</td>
<td>L</td>
<td>L</td>
<td>( \sim L )</td>
<td>L</td>
<td>M</td>
</tr>
<tr>
<td>10</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>( \sim L )</td>
<td>M</td>
</tr>
<tr>
<td>11</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

4.3.4 Experimental results and discussion for Part 1

For testing different controllers, similar traffic network model as previous section is considered, an intersection with four approaching links and four phases. The cycle times are divided between aforementioned four phases.

Two different scenarios are considered for evaluating the performance of the controllers:

1. Five hour simulation with 5,500 vehicles (peak load); and
2. Five hour simulation with 3,000 vehicles (non-peak load).

Similar structure are considered for NN controller and FLS controller. Table 4.2 and 4.3 show the parameters set for these controllers during our experiments.

The NN controller has four inputs (number of approaching links to the intersection) and
4.3 Part 1: NN and FLS controllers

Table 4.2
Parameters of NN controller

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of generation</td>
<td>300</td>
</tr>
<tr>
<td>Population size</td>
<td>20</td>
</tr>
<tr>
<td>Number of layers</td>
<td>3</td>
</tr>
<tr>
<td>Number of inputs</td>
<td>4</td>
</tr>
<tr>
<td>Number of neurons in hidden layer</td>
<td>10</td>
</tr>
<tr>
<td>Number of outputs</td>
<td>4</td>
</tr>
<tr>
<td>Range of inputs</td>
<td>0 - 50</td>
</tr>
<tr>
<td>Range of output</td>
<td>0 - 100</td>
</tr>
</tbody>
</table>

Table 4.3
Parameters of FLS controller

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of generation</td>
<td>300</td>
</tr>
<tr>
<td>Population size</td>
<td>20</td>
</tr>
<tr>
<td>Number of inputs</td>
<td>4</td>
</tr>
<tr>
<td>Number of outputs</td>
<td>1</td>
</tr>
<tr>
<td>Number of membership function for inputs</td>
<td>3</td>
</tr>
<tr>
<td>Number of membership function for output</td>
<td>3</td>
</tr>
<tr>
<td>Range of inputs</td>
<td>0 - 50</td>
</tr>
<tr>
<td>Range of output</td>
<td>0 - 100</td>
</tr>
</tbody>
</table>

For the FLS controller we have four inputs (number of approaching links to the intersection) and one output in our simulation for a four-way intersection. We consider four FLS controllers for controlling the intersection with four phases. Each FLS controller proposes the appropriate green time for one phase. The proposed green time with both NN controller and FLS controller are in range of 0 to 100 seconds.

Fig. 4.8 and 4.9 show the NN controller and FLS controller parameters optimization trend during training respectively. Scenario one with 5,500 vehicles is applied for training.
processes for both controllers. Both cost functions quickly converge to their global minimum in less than 50 iterations. The optimal set of parameters for NN controller and FLS controller remains almost unchanged after this iteration.

Two fixed-time controllers are developed as benchmarks. To have reliable comparisons, two different times for fixed-time controller are considered. The first one has 30 seconds green phase time and the second one is set to 70 seconds for each of the predefined phase.

During the evaluation a $Q$-learning controller, with similar structure but different setting, presented in previous chapter is also designed. The designed $Q$-learning controller has 81 states for a single intersection with four approaching links. The queue length in each approaching link is categorize in three groups; small corresponds to 0 to 4; medium relates to 5 to 12; and more than 12 is considered as the large category. In this controller, 10, 20, 30, 50 are acceptable green times. Therefore, we have 256 actions for the intersection with flexible cycle time. The cycle length is calculated by summation of an allocated green time.

![Best fitness Mean fitness](image-url)
4.3 Part 1: NN and FLS controllers

for each green phase plus two-second safety time after each green phase for clearing the intersection.

For the $Q$-learning controller with $\epsilon$-greedy method, a decreasing $\epsilon$ between 0.9 and 0.1 is considered, $\alpha$ (learning rate) is set to 0.1, and $\gamma$ (discount factor) is set to 0.9. The reward function is defined as below:

$$\text{Reward} = \frac{1}{\text{mean}(\sum_{i=1}^{4} d_i) + 1} \quad (4.12)$$

where $i = 1, \ldots, 4$ is the number of approaching links, $d$ is the calculated delay time for each link, and $+1$ is to refuse zero in denominator.

For both scenarios, we consider different seed numbers for each run of training and testing. Fig. 4.10 and 4.11 show the accumulative delay of the intersection in 10 runs for scenario one and two respectively. Each simulation lasts five hours.
The results show a better performance of intelligent controllers compared to both fixed-time controllers. Fixed-time controller has not flexibility to adapt to traffic demand. In average, by aiming to have less amount of delays for vehicles in the traffic network, the best result belongs to FLS, NN, and Q-learning controllers in respect. These controllers have 74%, 71%, and 66% improvement against designed fixed-time controller Table. 4.4.

It should be noted that FLS and NN controller can propose and use all the integer numbers between 0 and 100. However, the difficulty of Q-learning in handling huge amount of data forced us to limit the action list to 10, 20, 30, and 50 during the experiments. Putting this condition leads reducing the Q-learning controller performance.
4.3 Part 1: NN and FLS controllers

![Graph showing delay time vs minute for different controllers]

Fig. 4.11. The results of experiments for scenario two with 3,000 vehicles. Accumulative total delay for an intersection for ten simulation runs with different seed numbers in five hours (300 minutes) simulation.

Table 4.4
Performance Order Between Controllers based on the Average Delay Time in Seconds per Vehicle

<table>
<thead>
<tr>
<th>Method</th>
<th>Scenario 1</th>
<th>Rank in Scenario 1</th>
<th>Scenario 2</th>
<th>Rank in Scenario 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fixed-Time 30</td>
<td>25.45</td>
<td>4</td>
<td>10.08</td>
<td>4</td>
</tr>
<tr>
<td>Fixed-Time 70</td>
<td>29.54</td>
<td>5</td>
<td>21.94</td>
<td>5</td>
</tr>
<tr>
<td>Q-learning</td>
<td>9.88</td>
<td>3</td>
<td>6.85</td>
<td>3</td>
</tr>
<tr>
<td>NN</td>
<td>7.72</td>
<td>1</td>
<td>6.74</td>
<td>2</td>
</tr>
<tr>
<td>FLS</td>
<td>9.86</td>
<td>2</td>
<td>2.74</td>
<td>1</td>
</tr>
</tbody>
</table>

4.3.5 Conclusion of Part 1

In this part, four controllers are designed and implemented: Q-learning, NN, FLS, and fixed-time for an intersection with four approaching links and four different traffic signal
phases. The designed NN and FLS controllers are GA based. It means their parameters are not predefined and they obtained the optimal parameters during training time. It is attempted to develop and evaluate all the controllers in similar conditions.

The implemented intelligent controllers, especially NN and FLS controller, are flexible. It means these controllers are able to propose different range of number as traffic signal phase duration and they determine these numbers at the start of each cycle. This option gives the opportunity of using timer signal controller useful for drivers to know how long they have to stay in traffic. However, most of the previous controllers just have the ability to extend or terminate the current traffic phase.

The results of experiments for all designed controllers indicate FLS and NN controllers have the best performance. $Q$-learning controller also has a close performance to NN and FLS controllers. All intelligent controllers significantly outperform two fixed-time controllers. However, for having accurate conclusion about each of these controllers, there are some factors important to consider: Fixed-time controller may have a good performance in some cases, if the predefined time matches the traffic condition, but it is not useful for unpredictable situations of urban traffic. In addition, it needs experts’ knowledge to set the appropriate time for each phase. Appropriately defining of states, actions, and reward function are issues for $Q$-learning controller. Furthermore, $Q$-learning is not useful to handle a huge amount of states and actions. It does not need supervision during learning period, which is beneficial for controlling unpredictable urban traffic. FLS and NN are suitable methods because of their speed and accuracy in learning and approximating hidden patterns in a huge amount of data. The performance of both methods is sensitive to their initialization and training process.
4.4 Part 2: ANFIS controller

In this part, by designing an adaptive neuro-fuzzy controller it is attempted to improve the performance of the fuzzy controller.

All previous designed controllers are cycle based, meaning the controller proposed the appropriate green time for all phases in the cycle at the beginning of the cycle. During the experiment, it is figured out that having phase based controller can be more efficient. The traffic congestion will change over the time, therefore, proposing the green time at the start of a phase for the current phase can be more matchable with the current traffic situation. In this regard, all the controller after this are designed phase-based.

4.4.1 Design of ANFIS controller

ANFIS is made of the combination of both NN and FLS. One of the difficulties in applying FLS for a system is how to define the appropriate rule base to obtain the best efficiency of the FLS. In ANFIS system, first order Sugeno model fuzzy system modifies the rules and adaptively learns to reach the optimal parameters for the rule base.

In this part, an ANFIS controller is designed for a four-way intersection. The structure of the controller is similar to the previous ones designed in this thesis. The only difference is in being phase-based instead of being cycle-based. This controller has four inputs and one output. The queue lengths of vehicles at each approaching link of the four-way intersection make the inputs of the controller and the output of the system is the proposed green time for the next phase of the cycle. At the end of each phase the detected length of queues of all the approaching links are sent to the ANFIS controller and the controller sends the green time for the next phase. During the training GA evaluates the performance of the controller with different parameters until reach the optimal parameters for the ANFIS controller. Average delay time of a complete run of a simulation is considered as the cost function for the GA.
Fig. 4.12. The figure shows the process of ANFIS training. ANFIS parameters are updated after each round of simulation through GA optimization method.

It means GA aimed to reduce average delay time of the whole network in finding optimal parameters Eq. 4.13.

\[
\text{cost function} = \sum_{i=1}^{n} \frac{d_i}{k}
\]  

(4.13)

where \( i = 1, \ldots, n \) is the number of phases executed during the simulation time, \( d \) is the calculated delay time for each phase, and \( k \) is the number of cars released in each simulation scenario.

The implemented ANFIS controller is for intersection with four approaching links. It has four inputs and three membership functions are considered for inputs named small, medium, and large. The design and training process of ANFIS controller are presented in Fig. 4.12.
4.4 Part 2: ANFIS controller

4.4.2 Experimental results and discussion for Part 2

The experiments of this part are also evaluate at the aforementioned single intersection. Here, the performance of the ANFIS controller compare with a fixed-time controller with three different values, a fixed-FLS controller with predefined rules and membership functions, and a GA-FLS controller presented in previous part.

As it is mentioned before, three membership functions for each input of the ANFIS controller is considered. The range of the each membership function presented in Fig. 4.14. As the figure shows small is a trapezoidal function with the ranges: $[-1, -1, 10, 20]$, medium is triangular one with the range: $[10, 20, 30]$, and large is also trapezoidal function with the range: $[20, 30, 500, 500]$.

We consider a scenario with 800 cars in one hour simulation time. Fig. 4.13 shows the convergence trend of GA to find the optimal parameters for ANFIS controller. It shows that the optimization algorithm finds the optimal membership parameters in about 20 generations.

Similar membership functions are designed for GA-FLS controller Fig. 4.14. GA-FLS has fixed parameters for its four inputs but parameters of the output membership functions are adjusted using the training method. In the case of fixed-FLS parameters of all inputs and output are fixed and pre-defined. The parameters of inputs membership functions are similar to ANFIS and GA-FLS Fig.4.14, and the parameters of the output for fixed-FLS presented in Fig. 4.15.

Parameters of GA-FLS are optimally tuned using GA with the purpose of minimizing delay times. The GA used has 30 populations for optimizing seven parameters of eleven output membership functions. Fig. 4.16 shows the parameters of output membership functions after training with GA.

Fixed-time or pre-timed controller is usually used as a benchmark for evaluating the
performance of designed controllers. The designed fixed-time controller in this part uses equal time for all green phases. The designed fixed-time controller has three different values as green phase time: 20, 40, and 60 seconds values. This is done to have a more comprehensive comparison by considering three different values fixed-time controller.

The performance of each controller is evaluated by considering the total delay time of the intersection. Fig. 4.17 shows the accumulative delay time of the intersection in a one-hour simulation utilizing each controller.

The diagram illustrates that the ANFIS controller has a better performance than GA-FLS, fixed-FLS, and fixed-time controller. Second best controller is GA-FLS and then we have the fixed-FLS controller. Fixed-time controller has a different result for each green phase value. These differences prove that the efficiency of the fixed-time controller is
Fig. 4.14. Membership functions of inputs for ANFIS, GA-FLS, and fixed-FLS controllers. $Q_1$ is for the number of vehicles on a lane.

variable and highly depends on the traffic conditions.

Fig. 4.18 shows the total delay that impose to the intersection per vehicle in each hour. The bar chart also shows that the best performance is achieved by ANFIS controller. It shows the amount of delay per vehicle per hour for the intersection and these delay times are presented in seconds. FLS controllers have totally better performance than fixed-time controller. GA-FLS, which is a version of fuzzy controller with optimized output membership function parameters obtains better result than fixed-FLS, and ANFIS controller that has optimized rule base obtains the best results between other controllers.
4.4 Part 2: ANFIS controller

Fig. 4.15. Membership functions of output for fixed-FLS controller.

### 4.4.3 Conclusion of Part 2

In this part, an ANFIS controller is implemented and its performance is examined for controlling traffic signals for an isolated intersection. ANFIS gives the opportunity of using FLS in traffic signal controlling while there is no need to pre-defined rule base. Parameters of the ANFIS controller are optimally tuned using GA and ANFIS controller obtains its optimal rule base. The purpose of tuning and optimization is to minimize the total delay in the network. GA-FLS, a fuzzy controller with fixed and predefined parameters, and a fixed-time controller with three different values are also designed and implemented to evaluate the performance of the ANFIS controller. All the designed controllers are phase-based. Trapezoidal and triangular membership functions are considered for queue lengths in fuzzy controllers. Results of the experiments for the simulation scenarios show the better
4.5 Part 3: Optimized type-2 ANFIS controllers

As the next step to improve the fuzzy controllers performance, it was decided to apply IT2-FLS in design of ANFIS controller. In addition, to find the suitable optimization method for optimizing the parameters of the ANFIS rules, SA, GA and CS are used in the structure of the controller. The performance of T2-ANFIS controller in combination with each of these optimization methods is evaluated in almost similar testing situation.

![Membership functions of the GA-FLC output after optimizing the parameters.](image)

**Fig. 4.16.** Membership functions of the GA-FLC output after optimizing the parameters.

performance of the ANFIS controller compared to two others fuzzy controllers and fixed-time method.

4.5 Part 3: Optimized type-2 ANFIS controllers
4.5 Part 3: Optimized type-2 ANFIS controllers

4.5.1 Design of T2-ANFIS controller with SA, GA, and CS

Since presenting type-1 and type-2 ANFIS, different structures have been proposed for ANFIS controller by researcher to fit their case study. Here, our proposed IT2-ANFIS controller at each intersection has six layers. Explanation about each layer is presented below:

Layer 1: This layer is for fuzzification, where the inputs are clustered to different regions. Inputs of the controller are the number of vehicles in the queue and the number of vehicles is received from the neighboring intersection in each approaching link of the intersection. To reduce the complexity of the controller, two categories are considered, named “small” and “large”. Each category contains type-2 Gaussian membership functions with fixed mean and variable variance. Nodes of this layer produce the upper and
lower bound values corresponding to each input. The reason for considering Gaussian membership function is based on the suggestion in Wu and Mendel’s research [131].

**Layer 2:** Each rule of the controller is represented by a node in this layer. The output of this layer is product of the firing value of the inputs. A sample structure of the rule base for the designed controller is presented below:

\[
\text{If } Q_1 \text{ is (small/large) and } Q_2 \text{ is (small/large) and } Q_3 \text{ is (small/large) and } Q_4 \text{ is (small/large) then GreenTime} = aQ_1 + bQ_2 + cQ_3 + dQ_4 + e
\]

Based on the results of the Wu and Mendel’s research [131], it would be better to consider \(a, b, c,\) and \(d\) equal to zero to reduce the computational complexity. In this situation, for IT2-ANFIS we just need to find the optimal value for variable ‘\(e\)’ in each rule.

**Layer 3:** The output of this layer is the firing value of each rule.

**Layer 4:** This layer is for normalization.

**Layer 5:** Type reduction is done in this layer.
4.5 Part 3: Optimized type-2 ANFIS controllers

Table 4.5
Parameters of CS in T2-ANFIS

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of generation</td>
<td>100</td>
</tr>
<tr>
<td>Population size</td>
<td>20</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>16</td>
</tr>
<tr>
<td>Step size</td>
<td>0.1-0.9</td>
</tr>
<tr>
<td>probability of egg-discovery ( (P_a) )</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Layer 6: The output of the controller, which is the proposed green time for next green phase, is calculated in this layer.

Similar process as previous part (part 2) is considered for training the T2-ANFIS controller with optimization method. However, here three different optimization methods are applied for training and reaching the optimal rule base. By applying SA, GA, and CS in the same test condition, a comparison between their performance is presented during next subsection.

4.5.2 Experimental results and discussion for Part 3

In this part, CS, GA, and SA are three optimization methods used in combination with T2-ANFIS controller to optimize the parameters and reach the optimal rules. Each of these optimization methods has abilities that make them a powerful technique. Many studies have been done to compare the performance of different optimization methods and review their weak and strength points eg. [132, 133]. Among different optimization methods CS [130] shows a promising performance in finding optimal parameters without being trapped in local optima. This optimization method is also more suitable for problems with huge number of parameters. Here, after applying CS for the first time in optimizing the traffic controller, its performance is compared with two other well-known optimization methods. The setting of each method is presented in Tables. 4.5, 4.6, and 4.7.
4.5 Part 3: Optimized type-2 ANFIS controllers

Table 4.6
Parameters of GA in T2-ANFIS

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of generation</td>
<td>100</td>
</tr>
<tr>
<td>Population size</td>
<td>20</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>16</td>
</tr>
<tr>
<td>Mutation</td>
<td>0.01</td>
</tr>
<tr>
<td>Crossover</td>
<td>0.8</td>
</tr>
</tbody>
</table>

Table 4.7
Parameters of SA in T2-ANFIS

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum number of generation</td>
<td>100</td>
</tr>
<tr>
<td>Population size</td>
<td>20</td>
</tr>
<tr>
<td>Number of parameters</td>
<td>16</td>
</tr>
<tr>
<td>Maximum rejection</td>
<td>20</td>
</tr>
<tr>
<td>Maximum success</td>
<td>20</td>
</tr>
<tr>
<td>Maximum tries</td>
<td>20</td>
</tr>
<tr>
<td>Initial temperature</td>
<td>5</td>
</tr>
<tr>
<td>Stop temperature</td>
<td>1e-10</td>
</tr>
</tbody>
</table>

As it is mentioned before, in order to reduce the computational cost of experiments, two membership functions are considered for the inputs of the T2-ANFIS. The range of membership functions are such as presented in Fig. 5.5. The input, which is the number of vehicles on a link, is set between zero to 40. Mean (m), standard deviation 1 ($\sigma_1$) and 2 ($\sigma_2$) for the Gaussian membership functions of all inputs are:

Membership Function 1: $m : 0, \sigma_1 : 13.5891, \sigma_2 : 20.3837$

Membership Function 2: $m : 40, \sigma_1 : 13.5891, \sigma_2 : 20.3837$

Traffic scenario with 1,200 cars in one hour of simulation, used here for evaluation. The diagram Fig. 4.20 presents the accumulative delay time for the intersection.

As it is clear from the diagrams, the less amount of delay or best performance is for the
4.5 Part 3: Optimized type-2 ANFIS controllers

T2-ANFIS optimized by CS, the second place is for GA, and then we have the SA. The accumulative delay for the intersection per vehicle in one hour simulation for CS is 5.35 seconds, and it is 9.61 seconds for GA, and with using SA there is 38.12 seconds delay.

4.5.3 Conclusion of Part 3

In this part, in order to increase the accuracy of ANFIS controllers, IT2-FLS is applied in design of traffic signal timing controller. Based on the research done by Wu and Mendel [131] and their explanation regards benefits of choosing Gaussian membership function in design of type-2 fuzzy controller, Gaussian membership functions are set for the designed signal timing controllers. To reduce the computational cost of the training, the number of membership functions are limited to two for each input. In addition, with the purpose of
finding the most suitable optimization technique, three different optimization methods are used to find the optimal parameters for the T2-ANFIS controller. CS is one of the recent introduced optimization techniques that has revealed its superiority compared different optimization methods in different application. Here, the results show its superiority against GA and SA in almost similar testing conditions in designed traffic signal timing controllers.

4.6 Conclusions

This chapter is composed of three parts, and during that the focus was on finding the most suitable learning and optimization techniques to design a traffic signal timing system. In
part 1, a comparison between $Q$-learning, NN, and FLS traffic signal controller is presented. After obtaining the best result for the FLS controller, during part 2, the quality of FLS controller is improved by using ANFIS method. In part 3, IT2-FLS and three different optimization methods; GA, SA, and CS, are included to make a powerful traffic signal controller. Among them, timing with ANFIS and CS system impose less amount of delay to the intersection. In addition, all the experiments of this chapter are done in a phased-based simulation model in which the timing is done after finishing each phase and can be more accurate than a cycle-based model with timing at the end of each cycle.

Considering the obtained results in this chapter, a timing system applying CS and T2-ANFIS will be use for a network of multi-intersection in the next chapter.
Chapter 5

T2-ANFIS distributed traffic signal controller

5.1 Overview

Following on from the successful performing of the CS-based T2-ANFIS controller for a single intersection, this chapter presents a distributed controller using T2-ANFIS for a multi-intersection traffic network. A comprehensive evaluation of the proposed controller is done and its performance is compared with T1-ANFIS, fixed-fuzzy and fixed-time controller, which are all designed for a multi intersection traffic network.

5.2 Distributed multi-agent systems

An agent is anything that can perceive its environment through sensors and act upon that environment through actuators [134]. A rational agent is an agent that always tries to optimize its performance. Generally, autonomous agents are ones whose decision making relies to a larger extent on their own perception rather than on knowledge provided at design time.
A system that consists of a group of agents that can potentially interact with each other is called a Multi-Agent System (MAS), and the corresponding subfield of AI that deals with principles and design of MAS is called distributed AI.

For a single agent the environment that the agent interacts with is either static or dynamic. In MAS, because of the existence of other agents, the environment is dynamic. In a single agent system, it is assumed that the agent knows its own actions but not necessarily knows how the world is affected by the specific action. Also in MAS, the levels of knowledge of each agent about the world state and the other agents in the environment can be different. An agent may not have any knowledge about other agents action sets and their current perceptions, but generally in MAS, each agent must also consider the knowledge of each other agent in its decision making.

MAS interaction is considered as a form of communication. This communication is a two-way process in which agents send and receive messages. In this regard, the language and protocols for sending and receiving messages is important.

MAS has many special benefits and they have been widely adopted in many applications. Some of the advantages of using MAS technology in large systems [135], are listed in the following:

- Increase in the speed and efficiency of operations because of parallel computation and asynchronous operation;
- Increase in the reliability and robustness of the system when one or more agents fail;
- Increase in the scalability and flexibility. Different kinds of agents with different abilities can be introduced to the system;
- Reduced cost because implementing individual agents costs less than a centralized architecture; and
Agents have a modular structure and they can easily be reused without any major modifications and also they can be upgraded more easily than a monolithic system.

Although, there are many benefits in using multi-agent systems, some challenging issues are also considerable:

- Divide a problem to subtasks and allocate them to agents;
- Handle the distributed perceptual information;
- Implement decentralized control and efficient coordination among agents;
- Design efficient multi-agent planning and learning algorithms;
- Representation of knowledge and enabling agents to reason about the actions, plans, and knowledge of other agents;
- Enable agents to negotiate and resolve conflicts;
- Assign roles to agents; and
- Ensure coherent and stable system behavior.

Generally, it is not necessary to use MAS for all applications. MAS can be beneficial in cases where interaction between different people or organizations is necessary or in conflicting situations or where common goals are desired in the environment.

### 5.2.1 Agent taxonomy

MAS are divided in two categories based on the internal architectures of individual agents, homogeneous structure and heterogeneous structure.
Homogeneous structure

In this structure, all agents have a similar internal architecture. The local goals, sensor capabilities, internal states, inference mechanism and possible action states are related to the internal architecture [136]. In this structure, the differences of the agents are in their physical locations and the state in which an action is performed. Agents receive inputs from different parts of the environment. There may be some overlap in the sensory inputs, however, in a typical distributed environment the overlap of sensory inputs is rarely present [96].

Heterogeneous structure

Agents may have different capabilities, structures or functionalities in this architecture [137]. Using heterogeneous architecture is more appropriate for modeling applications closer to real-world [138]. In this architecture, agents can have different possibly conflicting goals. For example, in a predator-prey game, the prey and the predator have opposite objectives.

5.2.2 Learning in Multi-agent System

Building or modifying the belief structure in an agent based on the knowledge-base, input information, consequences or actions needed to achieve the local goal is known as learning [139]. Based on this definition there are two different classes of learning processes, active learning and leaning based on consequence.
### Active learning

The process of analyzing observations to create a belief or internal model of the corresponding situated agent’s environment is called active learning. This process can be performed by a deductive, inductive or probabilistic reasoning approach [16].

Deductive learning, prepares a deductive interface to explain a particular instance or state-action sequence. In this approach, what an agent learns is not new information because all are implied or deduced from the original knowledge-base, which can be very useful. The local goal of each agent contributes towards a part of the knowledge-base. Since the uncertainty and irregularity related to the agent knowledge is normally ignored, this method of learning is not suitable for real-time applications.

In inductive learning method, learning is based on the observation of state-action pairs. These observed state-action pairs are samples of some general rules or theories gained without a teacher or a reference model. This kind of learning is effective in situations where the environment can be presented as generalized statements. The deficiency of this learning method is at the beginning of the operation where sufficient data pertaining to the agent may not be available.

The probabilistic learning approach assumes that the agent’s knowledge-base or the belief model is represented as probabilities of occurrence of events. To predict the internal state of the agent, its observation of the environment is utilized. Bayesian learning is one the best example of probabilistic learning.

### Leaning based on consequence

In previous methods, learning was related to understanding the environment based on updating the belief model and analyzing patterns in sample observation. In consequence
learning, learning is done by evaluating the desirability of the performed action. Reinforcement learning is the learning method that performs this way. The agent starts learning through trial and error during its interactions with the environment. This learning is done without the help of any external teacher or supervisor who knows the solution.

5.3 Optimized T2-ANFIS traffic signal timing controller for multi-intersection traffic network

The traffic signal controller is designed and trained for four-phase intersections with four approaching links. The test bed is a network of nine intersections as shown in Fig. 5.1.

The proposed traffic signal controller is a distributed controller. Distributed controllers have priorities to central ones that urged us to use them. In a distributed model we can consider each intersection as an agent that controls traffic of its environment through its own controller while having collaboration with its neighbor agents (controllers). The controller sends its traffic data to its neighbors and receives their traffic data. Having distributed controllers makes it possible to benefits its properties previously mentioned.

T2-ANFIS controller for MAS has a similar structure at each intersection as presented in last chapter.

In the proposed distributed model, each intersection is considered as an agent in a network of intersections or a MAS. Each intersection controls its own traffic by aiming to reduce the delay time of the whole traffic network. All agents in the network are homogeneous. The designed control system does not need a central controller for cooperation. Each controller receives the traffic data from its neighbors and set the suitable green time for the next phase based on its current traffic condition and the incoming vehicles from the neighbors. The queue length in each approaching link of the intersection and the number
5.3 Optimized T2-ANFIS traffic signal timing controller for multi-intersection traffic network

Fig. 5.1. A network of nine intersections. Zones are the areas where vehicles are released to the intersection. Twelve zones are identified in this figure.

of vehicles coming from the neighbor intersections are used as inputs of the controller. The output of the controller is the proposed green time for the next traffic phase in each intersection.

The controller is designed to be able of estimating the exact time for each green phase at the start of that phase. The controller is not working just based on extension or termination of the current phase. This feature is useful for installing timer traffic signal lights, that give the ability of having an estimation of the end of current green or red phase to drivers. By using timer based traffic signal lights at intersections, drivers can adjust vehicles’ speed appropriately and do just necessary breaking or accelerating that leads to a reduction in the amount of $CO_2$ emission of cars.
5.3 Optimized T2-ANFIS traffic signal timing controller for multi-intersection traffic network

5.3.1 Cuckoo search optimization for MAS T2-ANFIS traffic signal controller

FLS is an efficient tool to design a traffic controller as it can describe traffic situations appropriately. Experts knowledge in the field of traffic is useful to design an FLC. Predefining appropriate membership functions for both inputs and outputs and also appropriate rule base need this knowledge. However, there may always be the possibility of using non-optimized parameters and rules. Here, instead of relying on expert knowledge we use ANFIS to design an optimized and flexible traffic signal controller.

The designed controller needs to be trained to obtain its optimal parameters. Different optimization methods are introduced to date that their efficiency varies in different case studies.

CS as one of the new optimization methods, introduced by Yang and Deb [130], shows its promising performance in different application [133, 140]. Also, it shows its superiority in traffic signal controlling for a single intersection against SA and GA, presented in chapter 4, part 3. Here, we apply CS to optimize the parameters of the proposed controller. In this regard, the average delay time of the network per vehicle is chosen as the cost function for the optimization Eq. 5.1.

\[
\text{cost function} = \frac{\sum_{i=1}^{p} \sum_{j=1}^{n} d_j}{\sum_{i=1}^{p} k_i} \tag{5.1}
\]

where \( i = 1, \ldots, p \) is the number of intersections, \( j = 1, \ldots, n \) is the number of phases executed during the simulation time, \( d \) is the calculated delay time for each phase, and \( k \) is the number of cars released in each simulation scenario.

During the optimization it is attempted to reach the minimum possible delay time for vehicles in the traffic network. CS proposes some parameters for the traffic controllers.
5.3 Optimized T2-ANFIS traffic signal timing controller for multi-intersection traffic network

After a complete run of traffic simulation with new parameters, the delay of whole traffic network per vehicle will be calculated and sent to the CS algorithm. In the next step, CS proposes some new values for the controllers and this process iterates until reaching the optimal parameters or a stop point set for the CS optimization process. Fig. 5.2 shows the diagram of these process.

The algorithmic form of the training process is provided in Figs. 5.3. These operations are repeated for every single controller in the traffic network. In fact, in a network of nine intersections, there are nine controllers. The training process is done for all of them separately but at the same time in a same traffic network. In this situation, we train a distributed controlling system with a unique aim. This aim is defined for the system by the cost function. The goal in training of the system is finding the minimum delay time for the vehicles traveling in the traffic network.

The proposed controller is designed for a network of intersections. Therefore, it is important to consider traffic congestion of neighbor intersections in timing of the current
5.3 Optimized T2-ANFIS traffic signal timing controller for multi-intersection traffic network

Fig. 5.3. This Algorithm shows the training process for optimizing the parameters of each controller in multi-intersection network.

1: begin
2: generate initial population for CS algo members.
3: repeat until completing CS process and finding the optimal parameters (controllers) with less delay time (details of CS presented in [132]);
4: run the traffic simulation and signal timing for each member of the population;
5: for each member of the population do
6: repeat until the end of the simulation time
7: receive the traffic data of each intersection
8: propose green time for the next phase of the intersection;
9: end
10: calculate the average delay time of the whole network per vehicle;
11: send the delay time to CS algorithm;
12: end for
13: end
14: set the optimal parameters for the adaptive controllers;
15: end

intersection’s signals. In this regard, the inputs of the controller are the length of queues at approaching links and the number of vehicles coming from neighbor intersections. These values are fed as inputs of the controller and the output is the estimated green time for the next phase.

Here, each intersection has its own cycle length. A maximum cycle length is set for intersections, however, there is no necessity to have equal cycle length for all intersections. In most of the studies [6, 45, 48], fixed cycle time is considered for controllers especially when they are designed for a multi-intersection network. Considering fixed cycle length makes the managing of the system easier, while by having flexible cycle length the efficiency of the system will be increased. Appropriate cycle length can reduce the delay time in traffic network. After each phase, and before starting the next phase a fixed amount of time is required for clearing the intersection, called safety time. For shorter cycle time, the amount of safety time is increased per hour. Therefore, there is lower overall capacity for intersections with shorter cycle times. On the other hand, if we have longer cycle
5.4 Experiments and benchmarks

A traffic network of nine intersections is designed in a traffic simulator (PARAMICS) to evaluate the performance of the proposed controller. All intersections are four-way with four phases Fig. 5.1. The simulation model is set up in PARAMICS version 6.8. and all controllers are implemented in Matlab R2011b.

Three different scenarios with two different seeds for each of them are considered for evaluating the performance of the controllers:

1. fifteen minute simulation with 680 vehicles;
2. fifteen minute simulation with 950 vehicles; and
3. fifteen minute simulation with 1,360 vehicles.

Scenarios are simulated in fifteen minutes to reduce the computation cost of the experiments. Different benchmarks have been considered in traffic simulation studies due to lack of a standard benchmark. In this section, we design three controllers for comparison. These benchmarks are named fixed-time controller, fixed-fuzzy controller, and T1-ANFIS controller. A brief explanation of each is presented in follow. It is attempted that the controllers benefit similar situations and evaluations for all controllers are done in similar traffic conditions.
5.4 Experiments and benchmarks

5.4.1 Fixed-time controller for multi-intersection network

In numerous studies, fixed-time controller is one of the benchmark for evaluating the performance of traffic controllers. In fixed-time controller a constant amount of time is set for each phase. The pre-defined time for each phase offers the least flexibility for fixed-time controller to adapt to traffic demands. A simple structure is considered for fixed-time controller. In this structure, equal time is set for all green phases. Here, we consider three different values for fixed-time controllers. These three values are 20, 50, and 80 seconds for each green phase. Different varieties of fixed-time controller is useful for more accurate comparison.

5.4.2 Fixed-fuzzy controller for multi-intersection network

The second designed controller is a fixed-fuzzy controller. This controller contains fuzzy logic control method type "Mamdani" with predefined values for membership functions and rule base. Similar to proposed T2-ANFIS controller, fixed-fuzzy controller has two Gaussian membership functions for each input. The parameters of the membership functions are also similar to T2-ANFIS. The output of the controller also contains two Gaussian membership functions set to be in the range of 0-100. This output is the proposed green time for the next traffic signal phase.

5.4.3 T1-ANFIS controller for multi-intersection network

The structure of the T1-ANFIS controller is very similar to T2-ANFIS. As it is clear from their names the only difference between these two controllers is in the type of fuzzy logic methods that each controller uses. In consequence of that, T1-ANFIS has one layer less than T2-ANFIS and this layer is for type reduction. In proposed T2-ANFIS, controller the
second layer is for type reduction. Similar to T2-ANFIS and fixed-fuzzy controllers, T1-ANFIS controller has two Gaussian membership functions for each input and the setting of the parameters of the membership functions are also similar. T1-ANFIS parameters are optimized with CS. T1-ANFIS uses similar setting for CS parameters and both controllers are the same in terms of size.

5.5 Experimental results and discussion

T2-ANFIS controller and all the benchmark controllers are implemented for a network of intersections. These controllers do not need any central controller for synchronization and cooperation. Each controller receives the traffic information of its immediate neighbors and adjusts the appropriate green time for the next phase of its own controller based on its current traffic condition and the information received from the neighbors.

We optimized the parameters of T2-ANFIS and T1-ANFIS controllers. The controllers are designed for a multi-intersection networks. To have an optimal ANFIS controller, it is perfect if we optimize both input membership functions parameters and the parameters of the consequent part of the rule base. It is necessary to have a synchronized optimization between all intersection in the network to obtain optimal controllers for the whole network. Therefore, the number of parameters will increased and computational cost increase as well.

To limit the number of parameters in ANFIS controller and increase the speed of convergence trend, we consider two membership functions for each input with fixed parameters, presented in Figs. 5.4 and 5.5. These membership functions are named small and large and demonstrate our perception of queue lengths. In ANFIS the number of rules directly depends on the number of inputs membership functions. Here, for both ANFIS controllers we have four inputs, each input is composed of the number of vehicles in the queue and the
number of vehicles that are arriving from neighbor intersection in each approaching link
to the intersection. There are two membership functions for each input, therefore, we will
have 16 rules with five parameters for each rule. The consequence part of each rule is like
Eq. 5.2.

\[\text{Rule} : aQ_1 + bQ_2 + cQ_3 + dQ_4 + e \tag{5.2}\]

where \(Q_1\ldots Q_4\) in Eq. 5.2 are the number of vehicles in each approaching link of the 4-way
intersection and \(a, b, c, d, e\) are the parameters to be optimized.

Each single controller in the network has \(16 \times 5 = 80\) parameters needing to be optimized. The traffic network is composed of nine intersections. In this situation, \(9 \times 80 = 720\)
parameters from consequent parts of rules need to be optimized. This is still a large number
reducing the chance of convergence to the optimal point. Wu and Mendel \[141\] present a
study about different setting for having the most effective design of interval T2-FLS. Based
on their work for the consequent part of rules we can set the parameters \(a, b, c,\) and \(d\) to
zero and just optimize the parameter \(e\) and still have a useful result from the controller.
Considering this fact, reducing the number of parameters from 720 to 144 greatly improves
the speed and performance of the optimization process. The convergence time for each
controller was approximately one week.

As mentioned before, both T2-ANFIS and T1-ANFIS controllers apply CS for optim-
izing their parameters. For both of these controllers, we consider fixed input membership
functions and there are 16 parameters in consequent parts of rules for each single controller.
The whole controlling system, in both T2-ANFIS and T1-ANFIS, have \(16 \times 9 = 144\) para-
eters in the traffic network. CS optimization is done for two aforementioned controllers
for 100 generations. The step size is set to 0.5 obtained base on some trials and errors. The
number of population for both is 20, and \(P_a\) is considered as 0.25. The outputs of these
controllers, which are the proposed green times for next phase, are set between 0 and 100 seconds.

Fixed-fuzzy controller has the same setting of input membership functions. This controller is a Mamdani type FLS and its output part contains two Gaussian membership functions with setting presented in Fig. 5.6. The output is set to be in range of 0-100 seconds.
A fixed-time controller with three different values is developed and considered as a basic benchmark during evaluations. The three different values for green phase time of the fixed-time controller are 20, 50, and 80 seconds respectively.

We consider two different seed numbers for testing and evaluating performances of controllers for three scenarios mentioned in previous section. Accordingly, each controller is evaluated in six different traffic simulations.

Table 5.1 shows the average delay time per vehicle in the traffic network for fifteen minute simulation. Obtained results demonstrate the superiority of ANFIS controllers over the three fixed-time controllers. T2-ANFIS has the best performance, however, the obtained delays for T1-ANFIS are close to the presented ones for T2-ANFIS. Values presented in
5.5 Experimental results and discussion

Table 5.1 are the average delay time per vehicle in traffic network for 15 minutes of simulation. It should be noticed that in 24 hours of a day and specially in busy hours, a few percentage higher performance of a controller has significant effect in reducing the traffic congestion.

In addition, the results show that fixed-time controllers with different green time phase values demonstrate different performances. For example, fixed-fuzzy controller has better performance compared to fixed-time controllers in green phase setting 50 and 80 seconds. However, fixed-time 20, except in scenario 3 seed 2, causes less delay for traffic system. Generally, fixed-time controller is not a robust controller for unpredictable traffic situations. It lacks the flexibility to adapt to traffic demand and its performance is variable based on traffic congestion and its predefined settings.
5.5 Experimental results and discussion

Table 5.1
Average delay time per vehicle in the network (times are presented in second). Fuzzy controllers are presented in the name of the methods they applied, T1 and T2 are for type-1 and type-2 respectively and FT is for fixed-time controller.

<table>
<thead>
<tr>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seed1 Seede</td>
<td>Seed1 Seede</td>
<td>Seed1 Seede</td>
</tr>
<tr>
<td>T2 – ANFIS</td>
<td>35.1</td>
<td>49.7</td>
</tr>
<tr>
<td></td>
<td>42.1</td>
<td>47.5</td>
</tr>
<tr>
<td>T1 – ANFIS</td>
<td>43.4</td>
<td>56.9</td>
</tr>
<tr>
<td></td>
<td>42.3</td>
<td>50.1</td>
</tr>
<tr>
<td>Fixed – fuzzy</td>
<td>61.7</td>
<td>73.0</td>
</tr>
<tr>
<td>FT20</td>
<td>55.5</td>
<td>69.5</td>
</tr>
<tr>
<td>FT50</td>
<td>69.7</td>
<td>78.6</td>
</tr>
<tr>
<td>FT80</td>
<td>84.7</td>
<td>90.6</td>
</tr>
</tbody>
</table>

Fig. 5.7 presents the performance of T2-ANFIS, T1-ANFIS, and fixed-fuzzy controllers against the average performance of the designed fixed-time controller with three different values in all the three scenarios. To reach these results, the percentage in reducing the average delay time per vehicle for both seed numbers for each controller against fixed-time controllers is calculated. The results show the superiority of the optimized T2-ANFIS, T1-ANFIS, and fixed-fuzzy controller respectively.

In addition, another calculation is done to show the average performance of each controller against average of all different values of fixed-time controller. Table 5.2 shows the results. According to these results, T2-ANFIS performs the best amongst other controllers as its performance improvement against the fixed controller is maximum (35.04%). T1-ANFIS got the second rank where its performance is almost 5% less than T2-ANFIS. Fixed-fuzzy has the lowest performance amongst them with an average 8.47% improvement over the fixed controllers.

Fixed-fuzzy controller has better performance than fixed-time controller. However, the results declare by replacing the fixed parameters with adaptive ones increases the performance up to 30%. In this work, we used CS for optimizing the parameters of the T2-ANFIS
5.5 Experimental results and discussion

Fig. 5.7. Results of experiments for three scenarios. The bars show the percentage of improvement of T2-ANFIS, T1-ANFIS, and fixed-fuzzy controller against average performance of fixed-time controllers.

<table>
<thead>
<tr>
<th>Method</th>
<th>Performance against fixed-time Controller</th>
</tr>
</thead>
<tbody>
<tr>
<td>T2-ANFIS</td>
<td>35.04 %</td>
</tr>
<tr>
<td>T1-ANFIS</td>
<td>30.82 %</td>
</tr>
<tr>
<td>Fixes-fuzzy</td>
<td>8.47 %</td>
</tr>
</tbody>
</table>

and T1-ANFIS controllers. This optimization methods is selected based on its demonstrated performance in different areas and its superiority to the other optimization methods such as genetic algorithm and PSO [132, 140]. The design of the both controllers are done in situation that both have similar amount of parameters for optimization process. With the same setting of CS parameters for both T2-ANFIS and T1-ANFIS controllers, the results show about 5% higher performance of T2-ANFIS. It can be concluded that T2-FLS is more suitable in design of the traffic signal controller.
5.6 Conclusions

In this chapter, different types of fuzzy controllers for traffic signal controlling in multi-intersection network are designed. Developing an adaptive fuzzy controller for a multi-intersection traffic network is the main aim of this chapter. To reach the optimal parameters for this controller, CS as a new developed nature-inspired optimization method, is used. In addition, using IT2-FLS and comparing its performance with T1-FLS show its superiority in timing the traffic signal lights. To evaluate the performance of the proposed controller, fixed-fuzzy controller and fixed-time controller with different values are also designed. Comparison revealed significantly higher performance of the adaptive proposed controller.

All types of designed controllers are distributed systems. Each single controller controls the traffic of its intersection through its own traffic congestion and traffic coming from its neighbors while aimed to reduce the vehicles delay in the whole traffic network. The controllers are also able to produce different green time values. This provides them with maximum flexibility to better respond to traffic demands. Producing the value of the next green phase instead of just terminating or extending a fixed amount of value to the current phase is the other benefit of the proposed controllers.
Chapter 6

Conclusions and future works

This chapter concludes the dissertation and provides recommendations for the future research that could enhance the functionality of the proposed intelligent controllers.

6.1 Conclusions

The main objective of this thesis is to design a distributed multi-agent base traffic signal controller through intelligent techniques. The choice of a distributed approach is motivated by the fact of high communication overhead and computational complexity. Regarding to applying intelligent methods, over the last few decades that AI methods have revealed their superiority in different fields and applications and also traffic signal timing against conventional systems. This research, focused on the capability of different intelligent techniques and uses a combination of them to create a powerful traffic signal controller with high accuracy.

At the first stage, a review of the previous studies in this field is presented. Different methods are presented in this regard and can broadly be divided into three groups. The first group is fixed-time controllers, which are not based on traffic demands and calculate signal times according to historical data. The next group are actuated controllers, where
signal controlling is performed by the help of sensors placed in points near stop lines and the current traffic demand determinants signal times. Adaptive signal timing is the final group with more flexibility to adapt signal times based on traffic.

Next stage is done in this thesis is developing a benchmark $Q$-learning controller based on [1]. In addition, the improved version of the $Q$-learning controller is designed and named SAQL. Both methods use $Q$-learning, which is a promising approach to control traffic lights and supports flexible rather than fixed cycle times. SAQL improvement is done by considering various state-space and reward functions. Evaluation of the performance of the controllers are done in a single intersection traffic model designed in PARAMICS. Results of experiments show higher performance of the proposed $Q$-learning controller methods compared to benchmark $Q$-learning controller and fixed-time controller.

Due to the review of the aforementioned stage and applying different techniques to traffic signal timing, it is decided to implement the most efficient methods for traffic signal timing in a unique testbed and compare their performance for the first time. In this regard, $Q$-learning, NN, FLS, and fixed-time are developed for an intersection with four approaching links and four different traffic signal phases. The designed NN and FLS controllers obtain their parameters during optimization by GA. These controllers are able to propose different range of number as traffic signal phase duration and they determine these numbers at the start of each cycle or phase. This characteristic provides the option of using timer signal controller, which is useful for drivers to know how long they have to stay in traffic.

The results of experiments for all designed controllers indicate FLS and NN controllers have the best performance. $Q$-learning controller also has a close performance to NN and FLS controllers. All intelligent controllers significantly outperform two fixed-time controllers. However, for having accurate conclusion about each of these controllers, there are some factors important to consider: Fixed-time controller may have a good performance in some cases, if the predefined time matches the traffic condition, but it is not useful for
6.1 Conclusions

unpredictable situations of urban traffic. In addition, it needs experts’ knowledge to set the appropriate time for each phase. Appropriately defining of states, actions, and reward function are issues for $Q$-learning controller. Furthermore, $Q$-learning is not useful to handle a huge amount of states and actions. It does not need supervision during learning period, which is beneficial for controlling unpredictable urban traffic. FLS and NN are suitable methods because of their speed and accuracy in learning and approximating hidden patterns in a huge amount of data. The performance of both methods is sensitive to their initialization and training process.

Recording the better performance of the FLS controller in designed scenarios, during next stage ANFIS controller is implemented and it’s performance is examined for controlling traffic signals for an isolated intersection. ANFIS gives the opportunity of using FLS in traffic signal controlling while there is no need to pre-defined rule base. Parameters of the ANFIS controller are optimally tuned using GA and ANFIS controller obtains its optimal rule base. Three other controllers are developed as benchmarks; GA-FLS, a fuzzy controller with fixed and predefined parameters, and a fixed-time controller with three different values. All the designed controllers are phase-based that proposed green time duration at the start of a phase and is pretty much accurate than cycle-based controllers. Trapezoidal and triangular membership functions are considered for queue lengths in fuzzy controllers. Results of the experiments for the simulation scenarios show the better performance of the ANFIS controller compared to two others fuzzy controllers and fixed-time method.

Next stage is developing T2-ANFIS traffic signal controller. In addition, three different optimization methods are used to find the optimal parameters for the T2-ANFIS controller. The results show the superiority of CS against GA and SA in almost similar testing conditions.

For both training and evaluating all designed controllers, multiple replications with
different seed numbers were considered for each scenario. Under all stochastic scenarios the algorithms were stable and converged.

6.2 Recommendations for future work

Although, the amount of research is done in the field of traffic signal timing is considerable, there is still significant time is spent in daily traffic congestion over the world. One of the reason should be the gap between the theoretical works and what is done in industry. The other reason could be related to the weakness of techniques which are currently applied. However, the role of increasing in the number of vehicles and travels in urban cities traffic are inevitable.

Here, some suggestions for the future work are presented.

- Most of the studies in controlling traffic signal timing is done for a single intersection or a few of them. Controlling traffic signal in a bigger traffic network and considering efficient coordination between intersections lead to better results in reducing the delay time of traveling.

- Most of the optimization methods usually have difficulty to convergence when encounter too many parameters. One of the future works is improving the quality of optimization methods and providing capabilities to handle huge number of parameters.

- Traffic signal timing and finding an optimal controller need enough data set for training. The controllers usually can propose effective values if they have been trained by similar data. To have accurate controllers it is helpful to consider all different factors affect in traffic condition, while providing all different cases for training is not practical and is very time consuming. Finding a solution to choose the useful factors and
6.2 Recommendations for future work

ignore the unnecessary ones during training is beneficial.

- Here, the number of membership functions for the ANFIS controller, which is the main developed controller, is fixed and set manually. Increasing the number of membership functions will lead to smoother and more powerful controller, but considering unnecessary membership functions can impose negative effects. Therefore, finding exactly the right numbers is beneficial in the performance of the controller. Providing intelligent technique for this purpose is useful.

- In this thesis, all the codes provided for traffic signal timing controller related to a specific intersection. Designing and developing a framework that can easily be adopted to all kind of intersections is beneficial.

- Regarding to the previous item, it should be added that the techniques used in this thesis and the developed systems could be suitable for different applications. A solid adaptable framework cover different application is useful.
References


REFERENCES


REFERENCES


[98] Y.-C. Chiou and Y.-F. Huang, “Stepwise genetic fuzzy logic signal control under


REFERENCES


